Metric Learning Tutorial

Parag Jain
cs13m1008@iith.ac.in
Indian Institute of Technology, Hyderabad

1 Metric Learning methods

Most popular machine learning algorithms like k-nearest neighbour, k-means, SVM uses a metric to identify the distance(or similarity) between data instances. It is clear that performances of these algorithm heavily depends on the metric being used. In absence of prior knowledge about data we can only use general purpose metrics like Euclidean distance, Cosine similarity or Manhattan distance etc, but these metric often fail to capture the correct behaviour of data which directly affects the performance of the learning algorithm. Solution to this problem is to tune the metric according to the data and the problem, manually deriving the metric for high dimensional data which is often difficult to even visualize is not only tedious but is extremely difficult. Which leads to put effort on *metric learning* which satisfies the data geometry.
Goal of metric learning algorithm is to learn a metric which assigns small distance to similar points and relatively large distance to dissimilar points.

Definition 1. *A metric on a set X is a function (called the distance function or simply distance).*
$d : X \times X \rightarrow R$,
where R is a set of real numbers, and for all x,y,z in X following condition are satisfied:
1.d(x, y) $\geq$ 0 (non-negativity)
2.d(x, y) = 0 if and only if x = y (coincidence axiom)
3.d(x, y) = d(y, x) (symmetry)
4.d(x, z) $\leq$ d(x, y) + d(y, z) (triangle inequality).

If a function does not satisfy the second property but satisfies other three then it is called a **pseudometric**. But since most of the metric learning methods learns a pseudometric instead of a metric for rest of the discussion we will refer pseudometric as metric. Most of the metric learning methods in literature learns the metric of form,

$$d_M(x, x') = \sqrt{(x - x')^T M (x - x')} \tag{1}$$

which is Mahalanobis distance,where, $M = (A^{1/2})^T (A^{1/2})$ is a positive semi-definite matrix.

July 2015

1.1 Supervised Metric Learning

Given a set of k dimensional data points $X \in \mathcal{R}^{N \times k}$, supervised metric learning methods learns a metric by using some similarity/dissimilarity information provided as a constraints. There are different formulations proposed for supervised metric learning accommodating different kinds of constraints. In a general supervised setting most popular form of constraints used in literature [6] are:

1. Similarity/dissimilarity constraints

$$\begin{aligned} d_A(x_i, x_j) \leq u \quad (i,j) &\in S \\ d_A(x_i, x_j) \geq l \quad (i,j) &\in D \end{aligned}$$

   where, $(i,j) \in S$ for objects that are similar, $(i,j) \in D$ for objects that are dissimilar.

2. Relative constraints
   $R = \ (x_i, x_j, x_k) : x_i$ should be more similar to x_j than to x_k .:

$$d_A(x_i, x_j) < d_A(x_i, x_k) - m$$

   Where m is margin, generally m is chosen to be 1.

Next section summarizes some of the widely used methods.

1.1.1 Large Margin Nearest Neighbor

Large Margin Nearest Neighbour(LMNN) [10] learns a metric of form 1 parameterized by matrix A for kNN classification setting. Intuition behind this method is to learn a metric so that the k-nearest-neighbours belongs to the same class while instances with difference class labels should be separated by a margin.

Let $X_{n \times d}$ is a set of data points in d dimensional space, and class labels $y_i \ : \ i = 1...n$

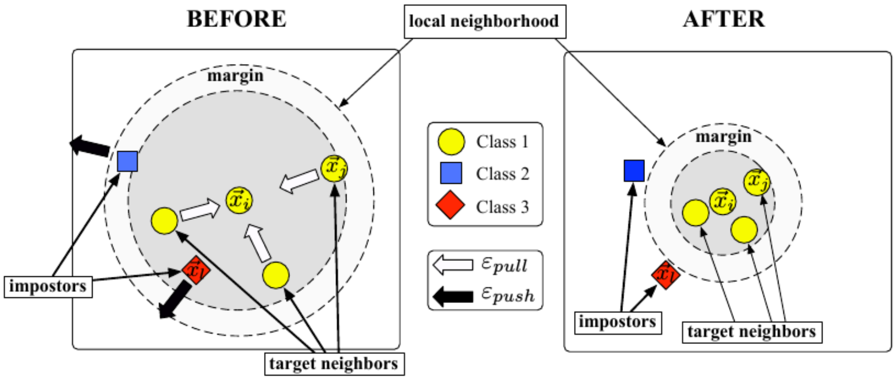

Figure 1: Schematic illustration of LMNN approach [10]

we define *target neighbours* for each point $x_i \in X$ as those points which are in k-nearest-neighbour of x_i and share the same label y_i and points which do not have same label as of x_i

we call them *impostors*. Formulation consist of two terms which compete with each other, first term is to penalizes the large distance between each point x_i and its target neighbors while second term penalizes small distance between x_i and impostors. Cost function is defined as:

$$\epsilon(L) = \sum_{ij} \eta_{ij} ||L(x_i - x_j)||^2 + c \sum_{ij} \eta_{ij}(1 - Y_{il}[1 + ||L(x_i - x_j)||^2 + ||L(x_i - x_l)||^2]_+) \tag{2}$$

Where Y_{ij} and η_{ij} are binary matrices such that Y_{ij} is 1 when labels y_i and y_j match and η_{ij} is 1 when x_j is in the target neighbours of x_i, in second term $[z]_+ = max(0, z)$ is a standard hinge loss function and c is some positive constant. Using cost function defined in 2 a convex optimization problem can be formulated as:

$$\begin{aligned} min \sum_{ij} \eta_{ij}(x_i - x_j)^T M(x_i - x_j) + c \sum_{ij} \eta_{ij}(1 - Y_{il}\xi_{ijl}) \\ \text{subject to} \quad (x_i - x_l)^T M(x_i - x_l) - (x_i - x_j)^T M(x_i - x_j) \geq 1 - \xi_{ijl} \\ \xi_{ijl} \geq 0 \\ M \succeq 0 \end{aligned} \tag{3}$$

where matrix $M = L^T L$ and η_{ijl} are slack variables.

1.1.2 Pseudo-Metric Online Learning Algorithm

Pseudo-Metric Online Learning algorithm (POLA) proposed by Shalev-Shwartz et al [8] updated/learns the mertic in an online manner. In this method, given that $\mathcal{X}$ denotes the feature space, POLA learns a metric of the form:

$$d_A(x, x') = \sqrt{(x - x')' A(x - x')}$$

where A is a positive semi-definite (PSD) matrix that defines the metric. The algorithm receives new samples as similarity and dissimilarity pairs in the form of $z = (x, x', y) \in (\mathcal{X} \times \mathcal{X} \times \{+1, -1\})$, where $y = +1$ if pair (x, x') are similar, otherwise $y = -1$. The loss function is defined as:

$$l_\tau(A, b) = \max\{0, y_\tau(d_A(x, x')^2 - b) + 1\} \tag{4}$$

where $b \in \mathbb{R}$ is a threshold. If $d_A(x, x') > b$, we predict pairs to be dissimilar otherwise similar. The goal is to learn a matrix-threshold pair (A_τ, b_τ) which minimizes the cumulative loss. At each step, the algorithm receives a pair (x, x', y) and updates the matrix threshold pair (A_τ, b_τ) in two steps:

1. Projecting the current solution (A_τ, b_τ) onto set C_τ where

$$C_\tau = \{(A, b) \in \mathbb{R}^{(n^2+1)} : l_\tau(A, b) = 0\}$$

i.e. C_τ is a set of all matrix-threshold pairs which gives zero loss on (x, x', y).

2. Projecting the new matrix-threshold pair to set of all admissible matrix-threshold pairs C_a,

$$C_a = \{(A, b) \in \mathbb{R}^{(n^2+1)} : A \succeq 0, b \geq 1\}$$

Projecting onto C_τ: We denote the matrix-threshold pair as a vector $w \in \mathbb{R}^{n^2+1}$, and $\mathcal{X}_\tau \in \mathbb{R}^{n^2+1}$ as a vector form of a matrix-scalar pair $(-y_\tau v_\tau v_\tau^t, y_\tau)$, where $v_\tau = x_\tau - x'_\tau$. Using this, we can rewrite set C_τ as,

$$C_\tau = \{w \in \mathcal{R}^{n^2+1} : w\mathcal{X}_\tau \geq 1\}$$

Now, we can write the projection of w_τ onto C_τ as:

$$\mathcal{P}_{C_\tau}(w_\tau) = w_\tau + \alpha_\tau \mathcal{X}_\tau \tag{5}$$

where $\alpha_\tau = 0$ if $w_\tau \mathcal{X}_\tau \geq 1$ otherwise $\alpha_\tau = (1 - w_\tau \mathcal{X}_\tau)/||\mathcal{X}_\tau||_2^2$, i.e.

$$\alpha_\tau = \frac{l_\tau(A_\tau, b_\tau)}{||\mathcal{X}_\tau||_2^2} = \frac{l_\tau(A_\tau, b_\tau)}{||v_\tau||_2^4 + 1}$$

Based on this, we can update matrix-threshold pair as:

$$A_{\hat{\tau}} = A_\tau - y_\tau \alpha_\tau v_\tau v_\tau^t, \quad b_{\hat{\tau}} = b_\tau + \alpha_\tau y_\tau \tag{6}$$

Projecting onto C_a: Projecting b_τ on set $\{b \in \mathbb{R} : b \geq 1\}$ is straightforward and can be achieved as $b_{\tau+1} = max\{1, b_\tau\}$. However, projecting $A_{\hat{\tau}}$ has two cases,

- $y_\tau = -1$: In this case, $A_{\hat{\tau}}$ becomes $A_{\hat{\tau}} = A_\tau + \alpha_\tau v_\tau v_\tau^t$ and $\alpha \geq 0$. Therefore, $A_{\hat{\tau}} \succeq 0$ and hence, $A_{\tau+1} = A_{\hat{\tau}}$.

- $y_\tau = +1$: In this case, we can write $A_{\hat{\tau}} = \sum_{i=1}^n \lambda_i u_i u_i^t$ where u_i is the i^{th} eigenvector of $A_{\hat{\tau}}$ and λ_i is corresponding eigenvalue. We can hence get $A_{\tau+1}$ by projecting $A_{\hat{\tau}}$ to the PSD cone as:
$$A_{\tau+1} = \sum_{i:\lambda_i > 0}^{n} \lambda_i u_i u_i^t$$

For every new sample pair, the update is done by successively projecting (A_τ, b_τ) to C_τ and C_a.

1.1.3 Information Theoretic Metric Learning

Information Theoretic Metric Learning(ITML)[2] learns a mahalanobis distance metric that satisfy some given similarity and dissimilarity constraints on input data. Goal of ITML algorithm is to learn a metric of form $d_A = (x_i - x_j)' A (x_i - x_j)$ according to which similar data point is close relative to dissimilar points.
ITML starts with an initial matrix d_{A_0} where A_0 can be set to identity matrix(I) or inverse of covariance of the data and eventually learns a metric d_A which is close to starting metric d_{A_0} and satisfies the the defined constraints. To measure distance between metrics it exploits the bijection between Gaussian distribution with fixed mean μ and Mahalanobis distance,

$$\mathcal{N}(x|\mu, A) = \frac{1}{Z} exp\Big(-\frac{1}{2} d_A(x, \mu)\Big)$$

Using the above connection, the problem is formulated as:

$$\begin{aligned} &\underset{A}{min} \mathcal{N}(x, \mu, A_0) log\left(\frac{\mathcal{N}(x, \mu, A_0)}{\mathcal{N}(x, \mu, A)}\right) dx \\ \text{subject to} \quad & d_A(x_i, x_j) \leq u \ \ \forall (x_i, x_j) \in S, \\ & d_A(x_i, x_j) \geq l \ \ \forall (x_i, x_j) \in D, \\ & A \succeq 0 \end{aligned} \tag{7}$$

Above formulation can be simplified by utilizing the connection between KL-divergence and LogDet divergence which is given as,

$$\begin{aligned} &\mathcal{N}(x, \mu, A_0) log\left(\frac{\mathcal{N}(x, \mu, A_0)}{\mathcal{N}(x, \mu, A)}\right) dx = \frac{1}{2} D_{ld}(A, A_0) \\ \text{where,} \quad & D_{ld}(A, A_0) = tr(AA_0^{-1}) - logdet(AA_0^{-1}) - d \end{aligned} \tag{8}$$

Using 8 and 7 problem can be reformulated as:

$$\begin{aligned} &\underset{A}{min} \quad D_{ld}(A, A_0) \\ \text{subject to} \quad & d_A(x_i, x_j) \leq u \ \ \forall (x_i, x_j) \in S \\ & d_A(x_i, x_j) \geq l \ \ \forall (x_i, x_j) \in D \\ & A \succeq 0 \end{aligned} \tag{9}$$

Above formulation can be solved efficiently using bregman projection method as described in [2].

1.1.4 Mirror Descent for Metric Learning

Mirror Descent for Metric Learning, by [7], is online metric learning approach which learns a pseudo-metric of form,

$$d_M(x, z)^2 = (x - z)^T M(x - z)$$

given a pair of labeled points,$(x_t, z_t, y_t)^T$, where y_t denotes similarity/dissimilarity. Taking μ as a margin, constraints can be written as,

$$y(\mu - d_M(x, z)^2) \geq 1$$
$$l(M, \mu) = \max\left\{0, 1 - y(\mu - d_M(x, z)^2)\right\}$$

Where $l(M, \mu)$ is hinge loss.To learn pseudo-metric incrementally from triplets, updates can be computed as,

$$M_{t+1} = \underset{M \succ 0}{argmin} B_\psi(M, M_t) + \eta \left\langle \Delta_M l_t(M_t, \mu_t), M - M_t \right\rangle + \eta\rho|||M|||$$
$$\mu_{t+1} = \underset{\mu \geq 1}{argmin} B_\psi(\mu, \mu_t) + \eta \Delta_\mu l_t(M_t, \mu_t)'(\mu - \mu_t).$$

Where $B_\psi(M, M_t)$ is bregman divergence, with $\psi(x)$ was taken as either squared-Frobenius distance and von Neumann divergence.

1.2 Unsupervised Metric Learning

Unsupervised metric learning is generally seen as a byproduct of manifold learning or dimensionality reduction algorithms, although metric learning has a direct connection between linear manifold learning techniques as it finally learns a projective mapping but for non linear techniques, which are more useful, connection is not exact and can only be seen with some approximations. Because of these limitations of manifold techniques unsupervised metric learning has its own importance. Unsupervised metric learning aims to learn a metric without any supervision, most of the method proposed in this area either solve this problem in a domain specific way like clustering [4] or by understanding the geometric properties of data.

1.2.1 Diffusion Maps

Diffusion maps [1] is a non-linear dimensionality reduction technique. Consider a graph $G = (\Omega, W)$ where $\Omega = \{x_i\}_{i=1}^{N}$ are data samples and W is a similarity matrix with $W(i,j) \in [0,1]$. W is obtained by applying Gaussian kernel on distances,

$$W(i,j) = exp\left\{\frac{-d^2(i,j)}{\sigma^2}\right\} \tag{10}$$

Using W we can obtain a transition matrix by row wise normalizing the similarity matrix:

$$P(i,j) = \frac{W(i,j)}{d_i} \text{ where, } d_i = \sum_{j=1}^{N} W_{ij} \tag{11}$$

Diffusion map introduce diffusion distance based on transition probabilities P of data, given as:

$$d_t^2 = ||P_t(i,:) - P_t(j,:)||_{1/\phi}^2 \tag{12}$$

where, $P_t = P^t$.

1.2.2 Unsupervised metric learning using self-smoothing operator

Unsupervised metric learning using self-smoothing operator [5] proposed a diffusion based approach to improve input similarity between data points. It uses similar framework as diffusion maps but instead of using the notion of diffusion distance it uses a Self Smoothing Operator(SSO) which preserves the structure of weight matrix W described in equation 10. Main steps of SSO algorithm are summarized below:

1. Compute smoothing kernel: $P = D^{-1}W$, where D is a diagonal matrix such that $D(i,i) = \sum_{k=1}^{n} = W(i,k)$
2. Perform smoothing for t steps: $W_t = WP^t$
3. Self-normalization: $W^* = \Gamma^{-1}W_t$ where Γ is a diagonal matrix such that $\Gamma(i,i) = W_t(i,i)$
4. Project W^* to psd cone $\hat{W}^* = psd(W^*)$

1.2.3 Unsupervised Distance Metric Learning using Predictability

Unsupervised distance metric learning using predictability [4] learns a transformation of data which give well separated clusters by minimizing the *blur ratio*. This work proposes a two step algorithm to achive this task which alternates between predicting cluster membership by using linear regression model and again cluster these predictions. Given input data matrix $X_{N \times p}$ with N number of points in p dimensional space goal is to find learn a mahalanobis distance metric $d(x, y) = \sqrt{(x - y)A(x - y)^T}$ which minimizes the blur ration defined as:

$$\underset{A,c}{min} BR(A, c) \equiv \frac{SSC}{SST}$$

where SSC and SST are within cluster variance and total variance respectively.

1.3 Laplacian eigenmaps

Laplacian eigenmaps learns a low dimensional representation of the data such that the local geometry is optimally preserved, this low-dimensional manifold approximate the geometric structure of data. Steps below describes the methods in detail.
Consider set of data points $X \in \mathcal{R}^N$, goal of laplacian eigenmaps is to find an embedding in m dimensional space where $m < N$ preserving the local properties of data.

1. Construct a graph $G(V, E)$ where E is set of edges and V is a set of vertices. Each node in the graph G corresponds to a point in X, we connect any two vertices v_i and v_j by an edge if they are close, closeness can be defined in 2 ways:

   (a) $||x_i - x_j||^2 < \epsilon$, $||.||$ is euclidean norm in $\mathcal{R}^N$ or,

   (b) x_i is in k nearest neighbour of x_j

   here ϵ & k are user defined parameters.

2. We construct a weight matrix $W(i, j)$ which assigns weights between each edge in the graph G, weights can be assigned in two ways:

   (a) Simple minded approach is to assign $W(i, j) = 1$ if vertices v_i and v_j are connected otherwise 0.

   (b) Heat kernel based, we assign weight $W(i, j)$ such that:
   $$W(i,j) = \begin{cases} exp(\frac{||x_i - x_j||^2}{t}) & \text{if } v_i \text{ and } v_j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

3. Construct laplacian matrix $L = D - W$ of the graph G, where D is a diagonal matrix with $D_{ii} = \Sigma_j W(i, j)$. Final low dimensional embedding can be computes by solving generalized eigen decomposition
   $$Lv = \lambda Dv$$

Let $0 = \lambda_0 \leq \lambda_1 ... \leq \lambda_m$ be the first smallest $m + 1$ eigenvalues, choose corresponding eigenvectors $v_1, v_2 ... v_m$ ignoring eigenvector corresponding to $\lambda_0 = 0$. Embedding coordinates can be calculates as mapping:

$$x_i \in \mathcal{R}^N \mapsto y_i \in \mathcal{R}^m$$
$$where \quad y_i^T = [v_1(i), v_2(i), ... v_m(i)]$$

$y_i, \ i = 1, 2 ... n$ is the coordinates in m dimensional embedded space.

1.4 Active Metric Learning

Active learning is a form of semi-supervised learning, difference is that in an active learning setup algorithm itself chooses what data it wants to learn. Aim is to select data instances which is most effective in training the model this saves significant cost to the end user end by asking less queries.

1.4.1 Active Metric Learning for Object Recognition

Active metric learning for object recognition by [3] propose to combine metric learning with active sample selection strategy for classification. This work explores to exploitation(entropy based and margin based) and two exploration(kernel farthest first and graph density) based strategy for active sample selection. To learn a metric Information theoretic metric learning is used, which is combined with active sample selection is two different modes,

1. Batch active metric learning: In this mode metric is learned only once, it starts with querying the desired number of labeled data points according to the chosen sample selection strategy and learns a metric based on this labeled data.

2. Interleaved active metric learning: This approach alternates between active sample selection and metric learning.

1.4.2 Metric+Active Learning and Its Applications for IT Service Classification

Metric+Active learning [9] learns a metric for ticket classification which are used by IT service providers. This work proposed two methods to solve this problem:

1. Discriminative Neighborhood Metric Learning (DNML): DNML aims to minimize the local discriminability of data which is same as maximize the local scatterness and to minimize the local compactness simultaneously.

$$J = \frac{\sum_{j:x_j \in \mathcal{N}_i^o} (x_i - x_j)^T C (x_i - x_j)}{\sum_{k:x_k \in \mathcal{N}_i^e} (x_i - x_k)^T C (x_i - x_k)}$$

Where $\mathcal{N}_i^o$ is nearest points from x_i with same labels as of x_i, $\mathcal{N}_i^e$ are nearest points from x_i which have different labels than of x_i.

2. Active Learning with Median Selection(ALMS): ALMS improves Transductive Experimental Design (TED) by using available labelled information.

doi:10.20944/preprints201809.0131.v1

References

[1] Ronald R Coifman and Stéphane Lafon. "Diffusion maps". In: *Applied and computational harmonic analysis* 21.1 (2006), pp. 5–30.

[2] Jason V Davis et al. "Information-theoretic metric learning". In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 209–216.

[3] Sandra Ebert, Mario Fritz, and Bernt Schiele. *Active metric learning for object recognition*. Springer, 2012.

[4] Ungar Lyle H. Gupta Abhishek A. Foster Dean P. "Unsupervised distance metric learning using predictability". In: *ScholarlyCommons Technical Reports (CIS)* (2008).

[5] Jiayan Jiang, Bo Wang, and Zhuowen Tu. "Unsupervised metric learning by self-smoothing operator". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 794–801.

[6] Brian Kulis. "Metric learning: A survey". In: *Foundations & Trends in Machine Learning* 5.4 (2012), pp. 287–364.

[7] Gautam Kunapuli and Jude Shavlik. "Mirror descent for metric learning: a unified approach". In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 859–874.

[8] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y Ng. "Online and batch learning of pseudo-metrics". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 94.

[9] Fei Wang et al. "Two heads better than one: Metric+ active learning and its applications for it service classification". In: *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE. 2009, pp. 1022–1027.

[10] Kilian Q Weinberger and Lawrence K Saul. "Distance metric learning for large margin nearest neighbor classification". In: *The Journal of Machine Learning Research* 10 (2009), pp. 207–244.