METRIC LEARNING

Basic Overview of Metric Learning
Supervised Methods
Unsupervised Methods

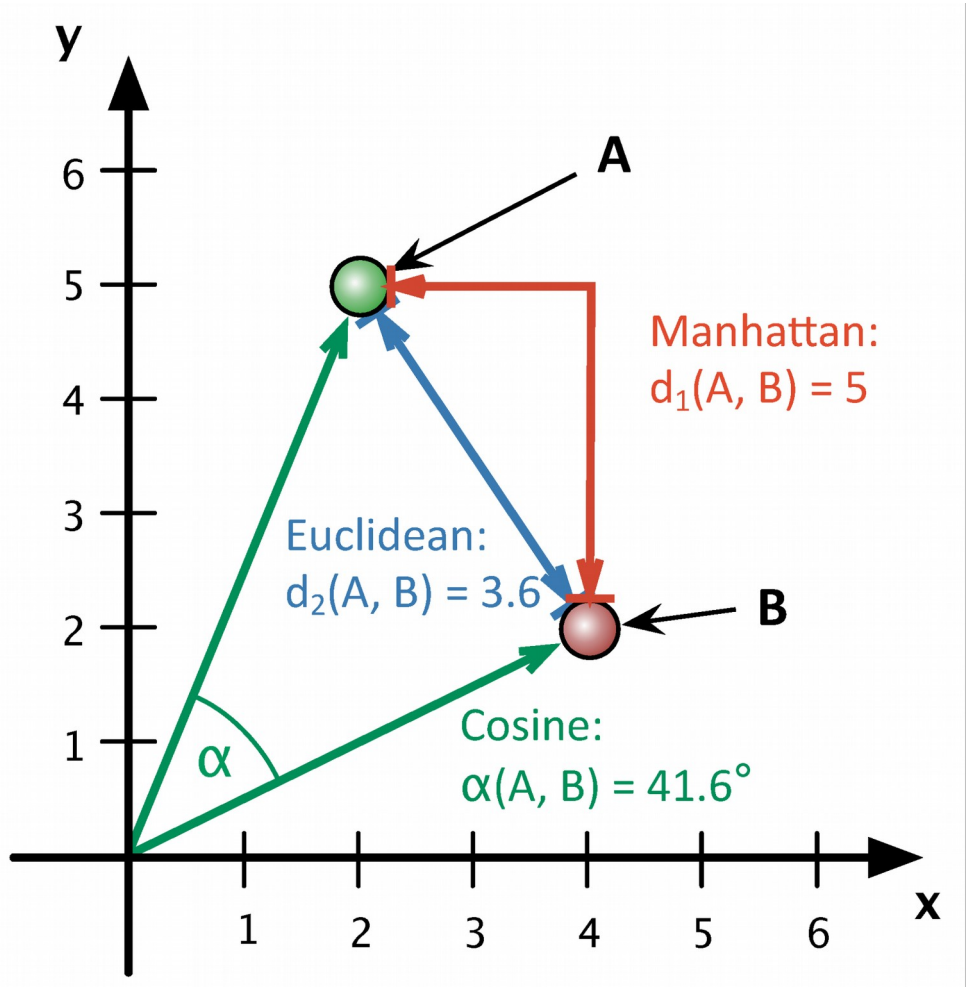
Why do we need Metric Learning?

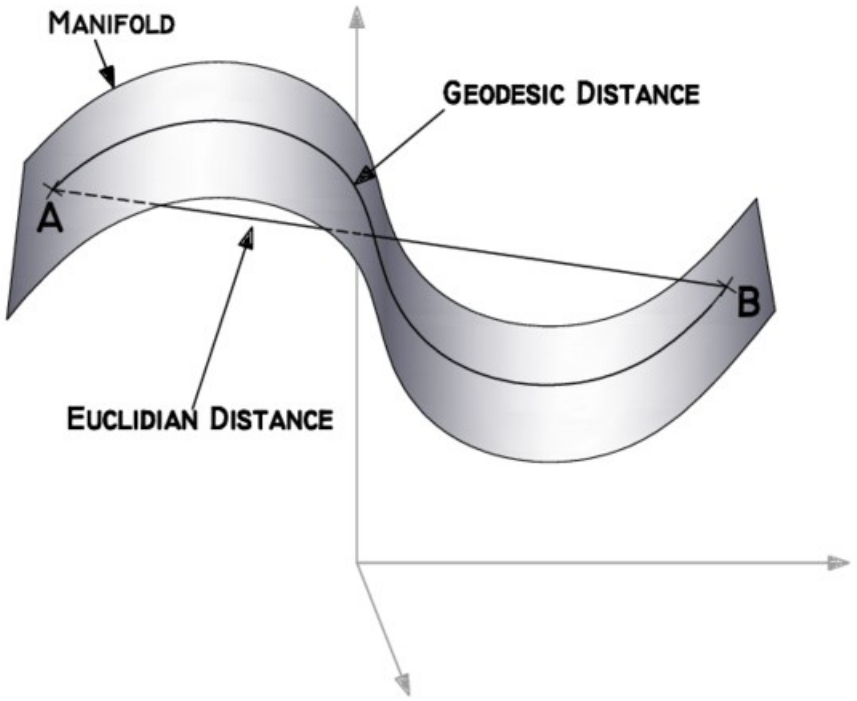
A good distance metric helps recognize similarity between sample points.

Similarity measures between points:

- Cosine Similarity
- Jaccard similarity
- Various distance functions
  - Euclidean Distance
  - Manhattan distance
  - Geodesic Distance

Various Similarity Measures Visualized





What is a Metric?

A metric or distance function is a function that defines a distance between each pair of elements of a set. We can use different distance measures to form a metric.

The following conditions need to be satisfied by a metric -

1. $D_{ij} \geq 0$
2. $D_{ii} = 0$
3. $D_{ik} + D_{kj} \geq D_{ij}$
4. $D_{ij} = D_{ji}$

Formally, it is a mapping over a vector space X

$$D : X \cdot X \rightarrow \mathbb{R}$$

Since, D defines the vector space itself in the Real plane

Why learn a Metric ?

Given the distance function formula, eg: Euclidean distance formula, we can simply calculate the distance metric over a dataset. Why should we learn it?

$$|x_i - x_j|^2 = (x_i - x_j)^T (x_i - x_j)$$

We can also use the metric as a transformation metric to transform the data to a new vector space. This is what we do in Metric Learning.

Another View: Mahalanobis metric

There is a different way to express a distance metric, is using the definition of Mahalanobis distance.

$$|x_i - x_j|^2 = (x_i - x_j)^T \sum^{-1} (x_i - x_j)$$

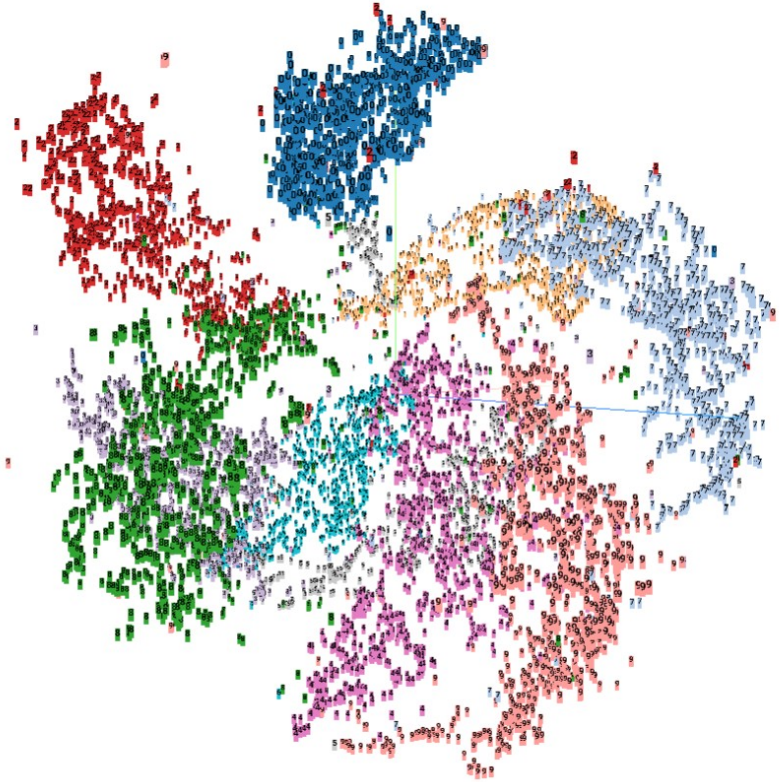
We can define the squared distance in terms of a squared metric,

$$\begin{aligned} D(x_i, x_j) &= (x_i - x_j)^T L^T L (x_i - x_j) \\ &= (x_i - x_j)^T M (x_i - x_j) \end{aligned}$$

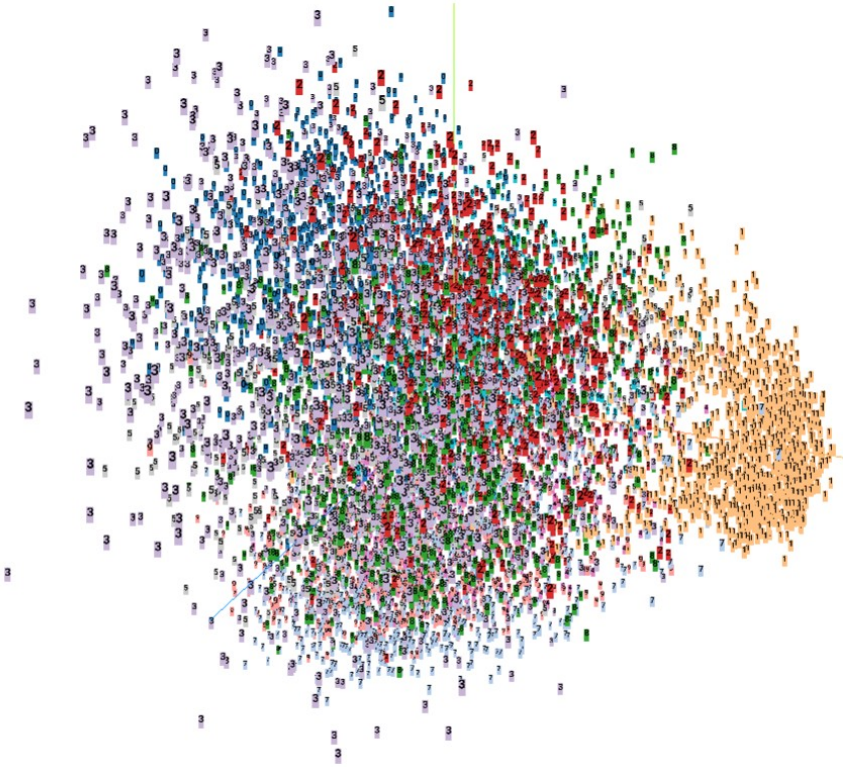
Setting $M = I$, we get back the Euclidean distance metric.

Data transformations on MNIST dataset

60,000 training samples, 10,000 test samples, 784 features, 10 classes



MNIST Dataset (tSNE)

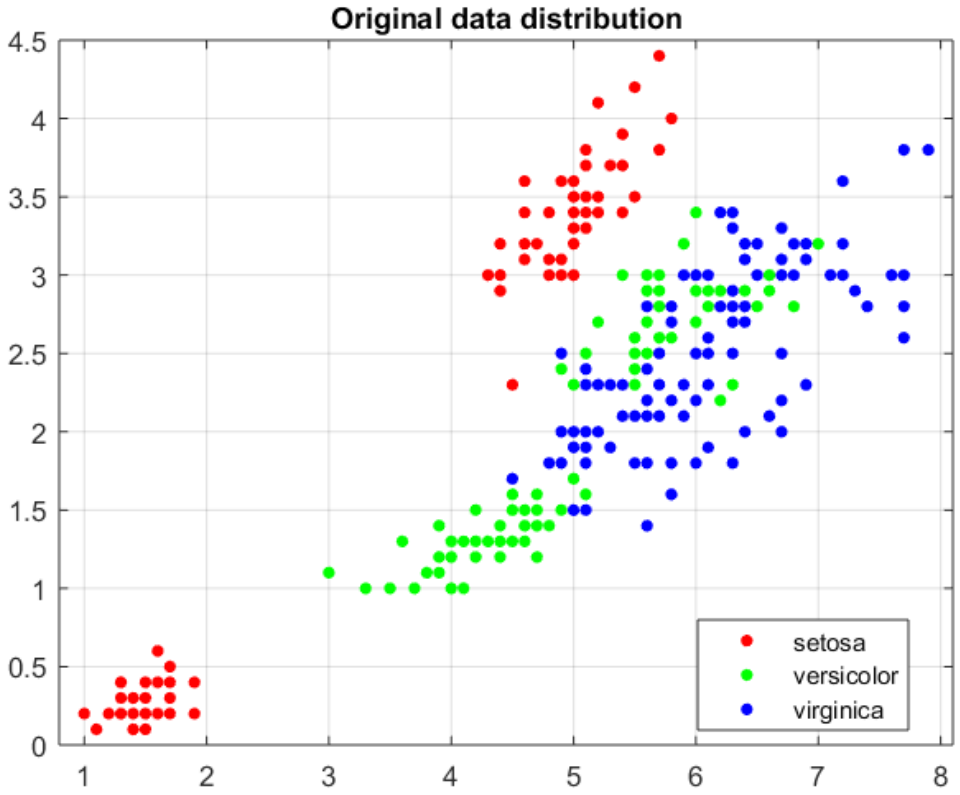


MNIST Dataset after PCA transformation

Iris Dataset

While MNIST is a more extensive and real-world dataset, we shall look at the use case of IRIS dataset since it is easier to visualize and understand.

Iris dataset has 3 classes of Iris flower species (Iris Setosa, Versicolour, Virginica) and 4 feature vectors (sepal length, sepal width, petal length, petal width in cm) for each of the 150 samples.



Metric Learning Algorithms

Supervised Algorithms

- Local Fisher Discriminant Analysis (LFDA)
- Large Margin Nearest Neighbor (LMNN)

Unsupervised Algorithms

- Mahalanobis Metric Learning for Clustering (MMC)
- Relative Components Analysis (RCA)
- Information Theoretic Metric Learning (ITML)

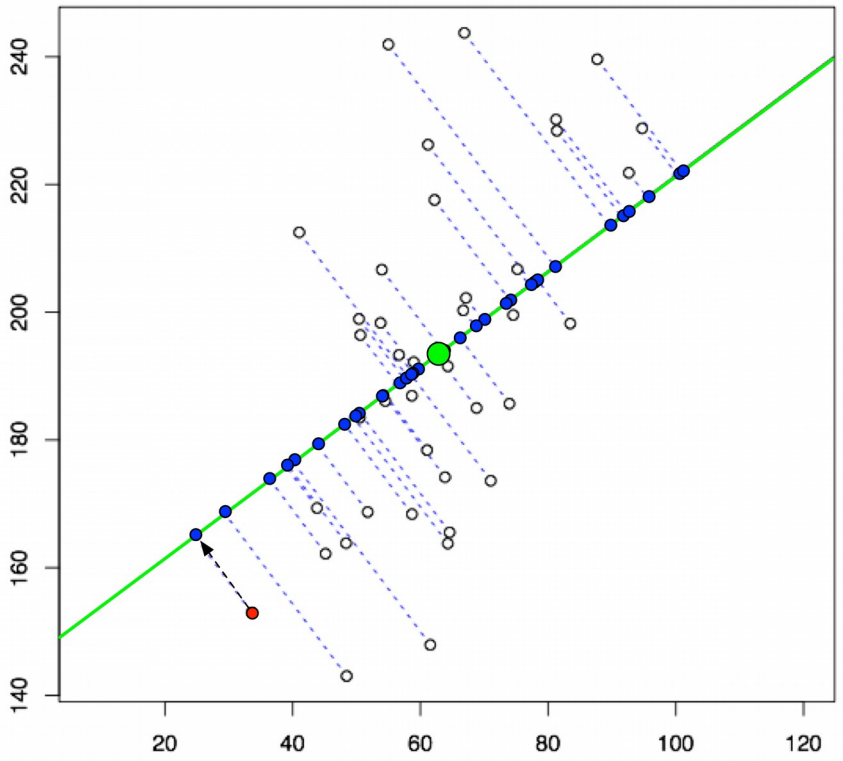
An overview of Principle Component Analysis

PCA finds the direction of maximum variance called the principle component.

$$\max_{U_1} \text{var}(U_1^T X)$$

$$\max_{U_1} (U_1^T S U_1) \text{ st } U_1^T U_1 = 1$$

Solved using Lagrange Multiplier or alternatively, perform singular value decomposition on co-variance matrix.

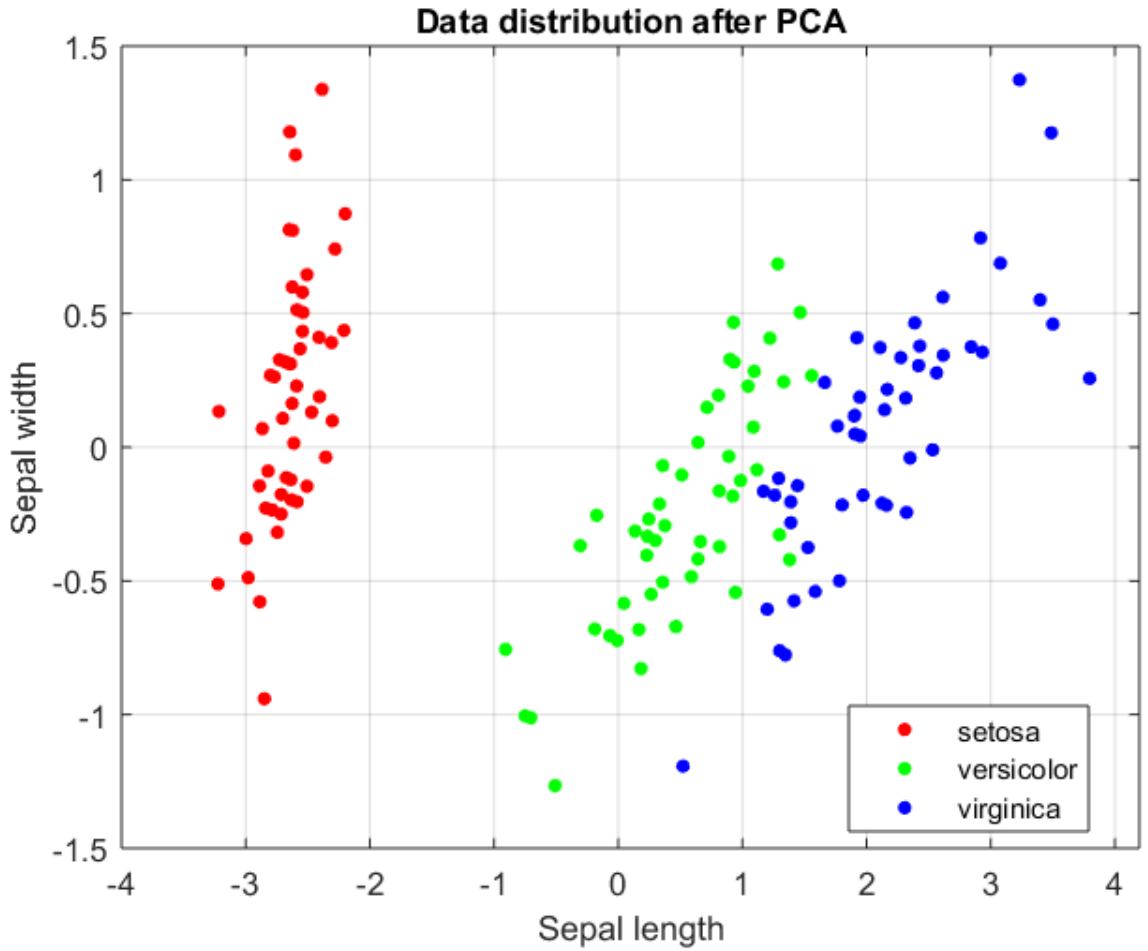


PCA on Iris Dataset

```
% pca
sigma = (1/m).*(meas')*meas; % compute covariance matrix
[U,S,V] = svd(sigma); % perform singular value decomposition
Ureduce = U(:,1:2);
Z = -1*meas*Ureduce;
% -1 because the eigen vectors over here come to be negative which makes the
% plot a mirror of the 'correct' plot
% depending on the eigen solver, the eigen vectors can have negative or
% positive results
```

1. Mean Normalization: sensitive regarding the variances of the initial variables.
2. Compute the Co-variance Matrix
3. Perform Singular Value Decomposition on Covariance Matrix
4. Choose the k^{th} top eigen-vectors as principle components.

PCA transformation of Iris Dataset



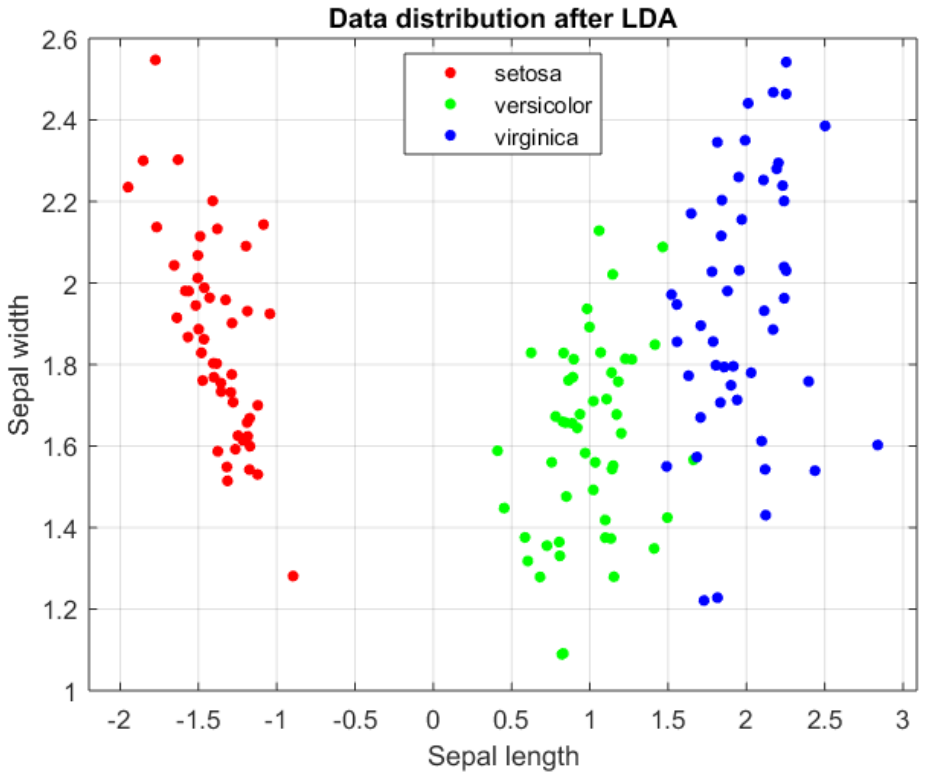
Supervised Metric Learning: LDA overview

LDA calculates the between-class and within-class scatter matrix and performs singular value decomposition on $S_b S_w^{-1}$

$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{n_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

$$S_b = \sum_{i=1}^{C} n_i (\mu_i - \mu)(\mu_i - \mu)^T$$

$$\max_T \; (T^T S_w T)^{-1} (T^T S_b T)$$



LDA code

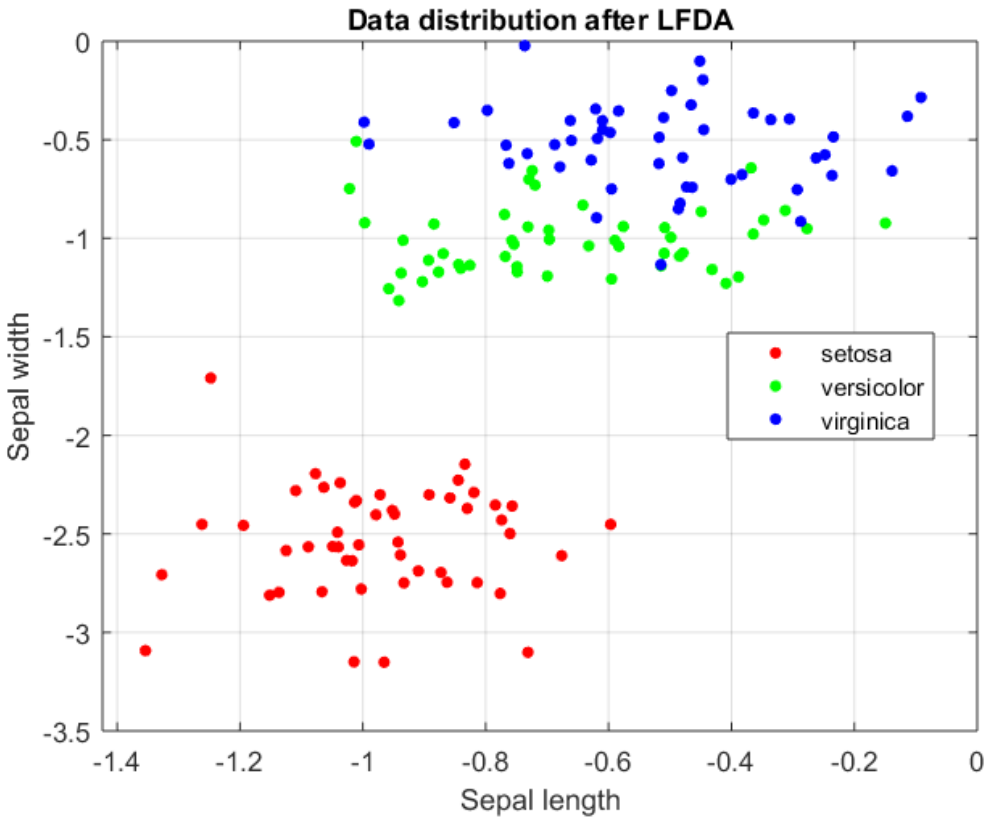
```
% between-class scatter matrix
Sb = zeros(n);
for i=1:num_classes
    temp = mean_vecs(i,:) - overall_mean;
    Sb = Sb + (samples_class(i)*temp'*temp);
end

% within-class scatter matrix
Sw = zeros(n);
i = 1;
while i<m
    temp = meas(i:i+50-1,:) - mean_vecs(M(char(species(i))),:);
    Sw = Sw + (temp'*temp);
    i = i+50;
end

W = inv(Sw)*Sb;
[V D] = eig(W);
```

LFDA

LFDA uses LDA and an affinity matrix from LPP (Locality Preserving Projections) to reformulate the objective function as,

$$\max_{T} (T^T S_w T)^{-1} (T^T S_b T)$$

which can be solved using SVD of $S_w^{-1} S_b$



LFDA

$$\overline{\boldsymbol{S}}^{(w)} = \frac{1}{2}\sum_{i,j=1}^{n} \overline{\boldsymbol{A}}_{i,j}^{(w)} (\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top,$$

$$\overline{\boldsymbol{S}}^{(b)} = \frac{1}{2}\sum_{i,j=1}^{n} \overline{\boldsymbol{A}}_{i,j}^{(b)} (\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top,$$

where

$$\overline{\boldsymbol{A}}_{i,j}^{(w)} = \begin{cases} \boldsymbol{A}_{i,j}/n_c & \text{if } y_i = y_j = c, \\ 0 & \text{if } y_i \neq y_j, \end{cases}$$

$$\overline{\boldsymbol{A}}_{i,j}^{(b)} = \begin{cases} \boldsymbol{A}_{i,j}(1/n - 1/n_c) & \text{if } y_i = y_j = c, \\ 1/n & \text{if } y_i \neq y_j. \end{cases}$$

LFDA

$$\boldsymbol{A}_{i,j}^{(w)} = \begin{cases} 1/n_c & \text{if } y_i = y_j = c, \\ 0 & \text{if } y_i \neq y_j, \end{cases}$$

$$\boldsymbol{A}_{i,j}^{(b)} = \begin{cases} 1/n - 1/n_c & \text{if } y_i = y_j = c, \\ 1/n & \text{if } y_i \neq y_j, \end{cases}$$

Here, the within-class and between-class affinity matrices are used from LPP and are used as to capture pairwise similarity. Thus, the scatter matrices obtained from these embed *local* information of the data and preserves *multimodal* structure of the data (structure within the class clusters)

LFDA Code

```
% calulate the scatter matrixes
Sw = zeros(n);
Sb = zeros(n);
for i=1:m
    for j=1:m
        temp = meas(i,:) - meas(j,:);
        tSb = tAb(i,j)*(temp'*temp);
        tSw = tAw(i,j)*(temp'*temp);
        Sb = Sb + tSb;
        Sw = Sw + tSw;
    end
end
Sb = 0.5*Sb;
Sw = 0.5*Sw;

W = inv(Sw)*Sb;
[V D] = eig(W);
```

Large Margin Nearest Neighbor (LMNN)

LMNN uses a binary matrix y_{ij} to indicate if labels y_i and y_j match or not and another binary matrix n_{ij} to indicate if the samples x_i and x_j are neighbors are not. This is determined using Euclidean distance for k Nearest Neighbors. These two matrices remain unchanged during computation and they help in reserving the intrinsic geometry (local information) of the data distribution.

Minimize $\sum_{ij} \eta_{ij}(\vec{x}_i - \vec{x}_j)^\top \mathbf{M}(\vec{x}_i - \vec{x}_j) + c\sum_{ij} \eta_{ij}(1 - y_{il})\xi_{ijl}$ **subject to:**

(1) $(\vec{x}_i - \vec{x}_l)^\top \mathbf{M}(\vec{x}_i - \vec{x}_l) - (\vec{x}_i - \vec{x}_j)^\top \mathbf{M}(\vec{x}_i - \vec{x}_j) \geq 1 - \xi_{ijl}$

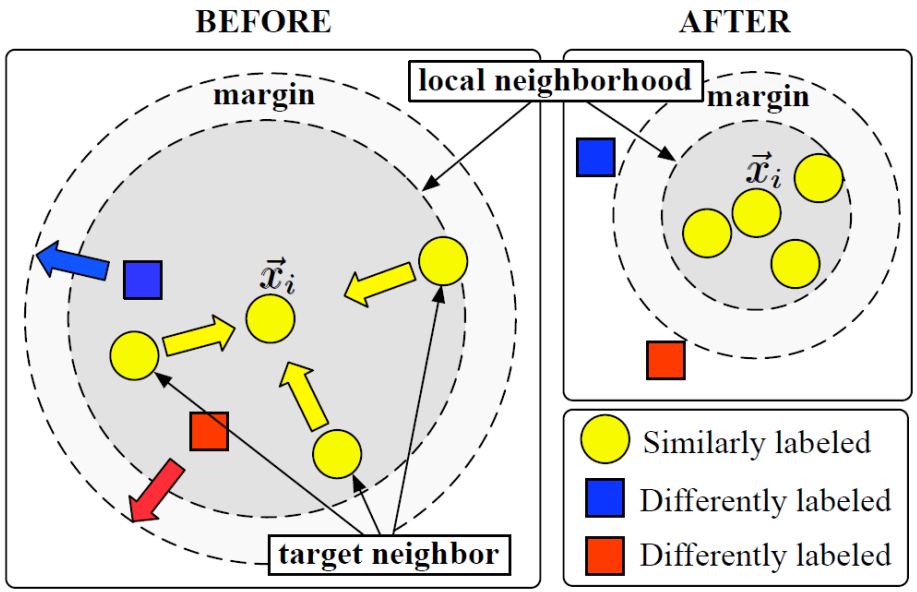
(2) $\xi_{ijl} \geq 0$

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

LMNN

This objective function can be formulated as a problem of instance of semidefinite programming (SDP). The matrix defined in the constraint is required to be positive semidefinite (PSD).

SDPs are convex and thus, there should not be any problem of getting stuck in local minimum.

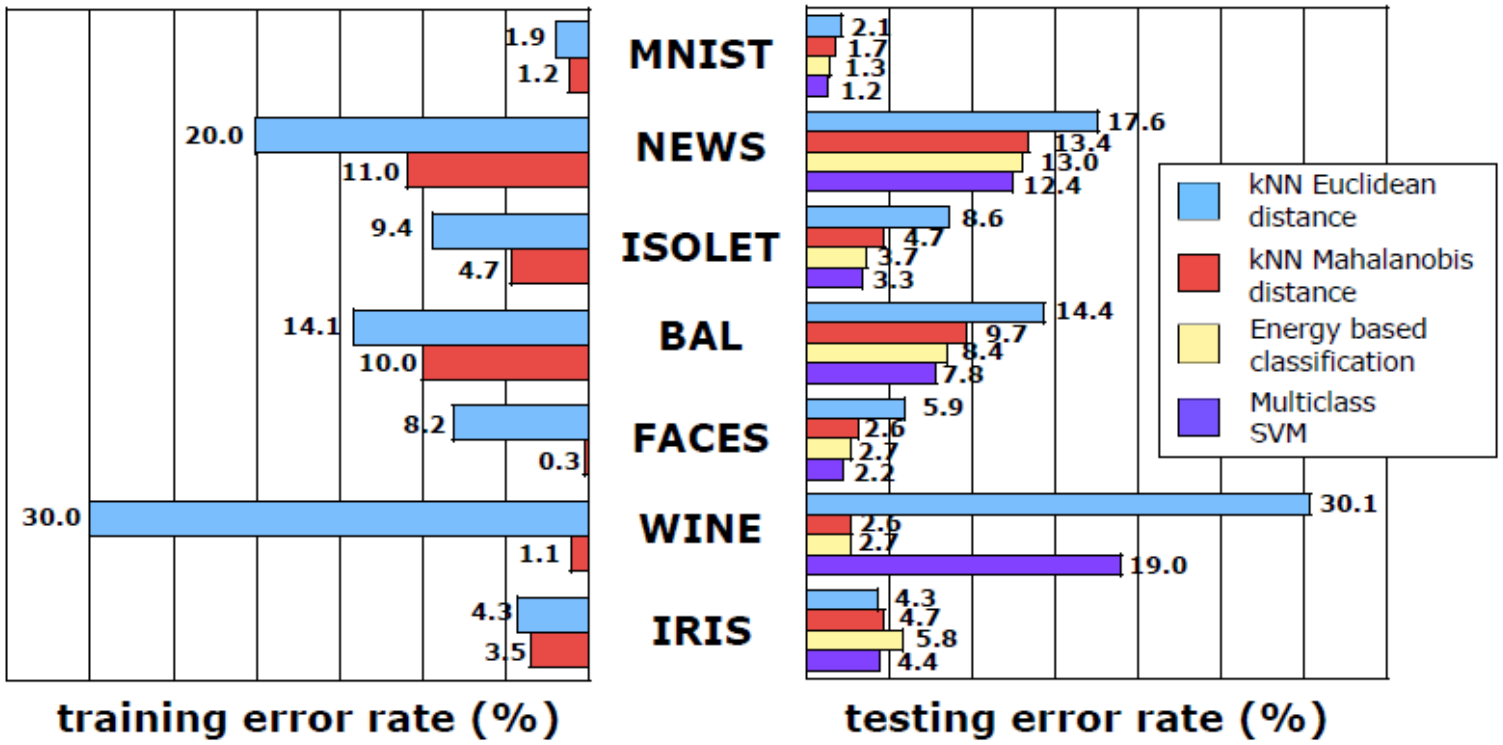


LMNN code

The SPD problem can be written in MATLAB 2017b onwards using the Optimization Toolbox function optimproblem which allows you to define the equality, inequality and bounded constraints for the objective function.

```
obj_fun = sum(sum(((x(i)-x(j))'M(x(i)-x(j))))+c*sum(sum(n_ij(1-y_ij)E_ijl));

linprob = optimproblem('Objective',obj_fun);

linprob.Constraints.cons1 = (x(i)-x(l))'M(x(i)-x(l))-(x(i)-x(j))'M(x(i)-x(j)) >= 1 - E_ijl;

linprob.Constraints.cons1 = E_ijl >= 0;

linprob.Constraints.cons1 = M >= 0; %SPD

linsol = solve(linprob);
```

LMNN on various datasets



The wine, iris, and balance data sets are small data sets and hence show comparable results when the other distance functions are used but on larger datasets show much less training error rate.

Mahalanobis Metric Learning for Clustering (MMC)

MMC minimizes the sum of squared distances between similar examples, while enforcing the sum of distances between dissimilar examples to be greater than a certain margin.

This leads to a convex and, thus, local-minima-free optimization problem that can be solved efficiently.

However, the algorithm involves the computation of eigenvalues, which is the main speed-bottleneck.

MMC

1. Learning a Mahalanobis Distance Metric

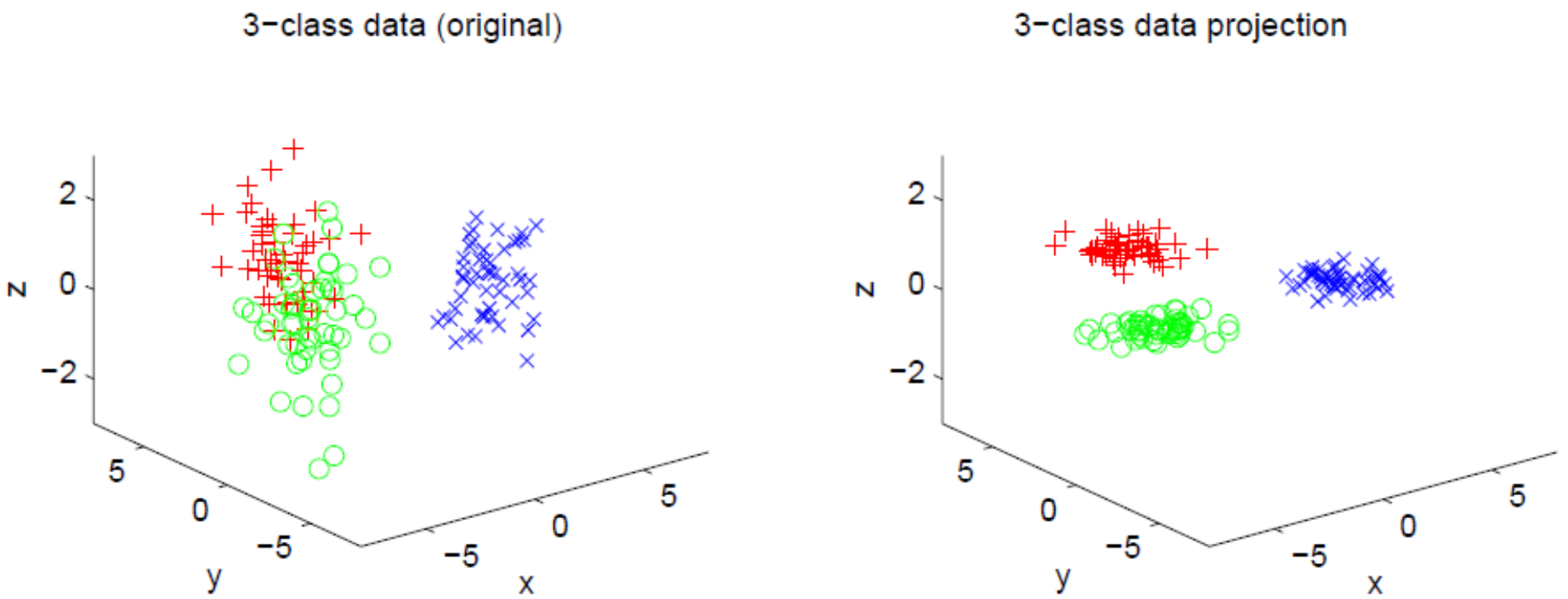
$$d(x, y) = d_A(x, y) = ||x - y||_A = \sqrt{(x - y)^T A(x - y)}.$$

2. Optimization problem with class constraints

$$\begin{aligned} \min_A \quad & \sum_{(x_i, x_j) \in \mathcal{S}} ||x_i - x_j||_A^2 \\ \text{s.t.} \quad & \sum_{(x_i, x_j) \in \mathcal{D}} ||x_i - x_j||_A \geq 1, \\ & A \succeq 0. \end{aligned}$$

3. Convex Optimization Problem: use SVD. There is a global minimum that the metric will converge to.

MMC


3−class data (original)
2
0
−2
z
5
0
−5
y
−5
0
5
x

3−class data projection
2
0
−2
z
5
0
−5
y
−5
0
5
x

Relative Components Analysis (RCA)

RCA does Adjustment learning which has the following features,

- Classication - the generation of equivalence classes at the output that would be relevant to the task.
- Reduce irrelevant variability, without modeling its exact sources
- Unsupervised learning where no labels are given and chunklets can be identified in the data.

RCA is defined as compression of the data along the dimensions of highest irrelevant variability, thus preserving the relevant variability and minimizing the irrelevant variability by using a weighted transformation matrix.

RCA

We use scatter matrices again but instead of defining within and between class scatter matrices we define a chunklet scatter matrix S_{ch} to provide an estimate of S_w (within-class scatter matrix). We perform SVD on the total co-variance matrix S_T and obtain the *r* largest dimensions the data is to be projected to. We then use a whitening re-scaling transformation to project S_{ch} to the lower dimensional space.

$$W = V\Lambda^{-\frac{1}{2}}$$

$$S_W = V\Lambda V^T$$

RCA

$$S_{ch} = \frac{1}{|\Omega|} \sum_{n=1}^{N} |H_n| Cov(H_n) = \frac{1}{|\Omega|} \sum_{n=1}^{N} \sum_{j=1}^{|H_n|} \left(\boldsymbol{x}_n^j - \hat{\mu}_n\right) \left(\boldsymbol{x}_n^j - \hat{\mu}_n\right)^T$$

Ω is the data distribution, H_n is the sample from the nth chunklet, $Cov(H_n)$ approaches co-variance matrix of the class when added with the co-variance matrix of the contribution from sensors and the global environment.

Sch can be understood as shifting each chunklet so that its mean conincides with the origin. The shifted chunklets are then superimposed which gives a good approximation of the within class scatter matrix since all the global data distribution information is being ignored.

Information Theoretic Metric Learning (ITML)

Instead of learning an optimal distance metric, an optimal Gaussian is learnt with respect to an entropic objective. The objective can be viewed as: maximize the differential entropy of a multivariate Gaussian subject to constraints on the associated Mahalanobis distance. Mahalanobis distance generalises the standard Euclidean distance by admitting linear scalings and rotations of the feature space.

$$d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T A(\boldsymbol{x}_i - \boldsymbol{x}_j).$$

$d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq u$ similar points

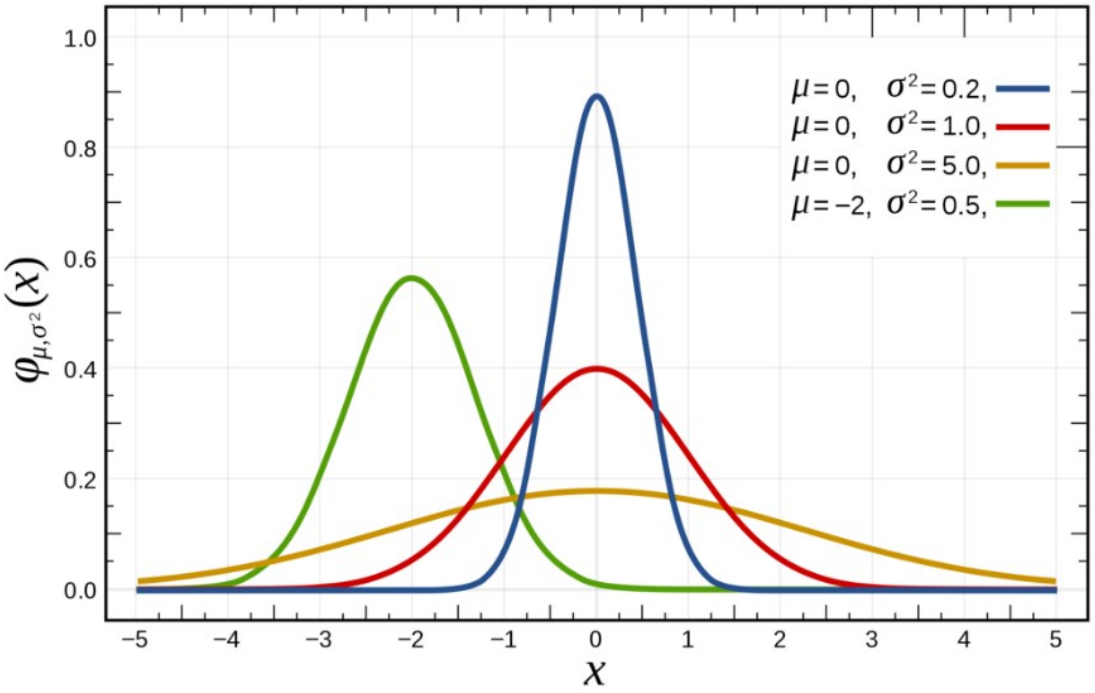
$d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \ell$ dissimilar points

Overview on Gaussian Distributions

(also called Normal Distributions)

- Very commonly seen in natural phenomena
- Central limit theorem
- Entire distribution can be specified by mean and variance

Hence, commonly used in statistics and Machine Learning.



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\mu - x)^2}{2\sigma^2}}$$

ITML

We define a Mahalanobis matrix A to be as close as possible to a Mahalanobis distance function A_0 We can then define the relative entropy between their corresponding multivariate Gaussians.

$$\mathrm{KL}(p(\boldsymbol{x}; A_o) \| p(\boldsymbol{x}; A)) = \int p(\boldsymbol{x}; A_0) \log \frac{p(\boldsymbol{x}; A_0)}{p(\boldsymbol{x}; A)} \, d\boldsymbol{x}.$$

$$\begin{aligned} \min_{A} \quad & \mathrm{KL}(p(\boldsymbol{x}; A_0) \| p(\boldsymbol{x}; A)) \\ \text{subject to} \quad & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq u \qquad (i, j) \in S, \\ & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \ell \qquad (i, j) \in D. \end{aligned}$$

ITML Problem Optimization

Use LogDet (logarithm determinant) divergence which is a Bregman matrix divergence generated over a convex function. The multivariate Gaussians can be expressed as the convex combination of a Mahalanobis distance between mean vectors and LogDet divergence between the co-variance matrices.

$$\begin{aligned} \mathrm{KL}(p(\boldsymbol{x}; A_0) \| p(\boldsymbol{x}; A)) &= \frac{1}{2} D_{\ell \mathrm{d}}(A_0^{-1}, A^{-1}) \\ &= \frac{1}{2} D_{\ell \mathrm{d}}(A, A_0), \end{aligned}$$

The optimization method which forms the basis for the algorithm repeatedly computes Bregman projections—that is, projections of the current solution onto a single constraint.

References

- Wold, S., Esbensen, K. and Geladi, P., 1987. Principal component analysis. Chemometrics and intelligent laboratory systems, 2(1-3), pp.37-52.
- Mika, S., Ratsch, G., Weston, J., Scholkopf, B. and Mullers, K.R., 1999, August. Fisher discriminant analysis with kernels. In Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468) (pp. 41-48). Ieee.
- Sugiyama, M., 2007. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. Journal of machine learning research, 8(May), pp.1027-1061.
- Weinberger, K.Q., Blitzer, J. and Saul, L.K., 2006. Distance metric learning for large margin nearest neighbor classification. In Advances in neural information processing systems (pp. 1473-1480).
- Xing, E.P., Jordan, M.I., Russell, S.J. and Ng, A.Y., 2003. Distance metric learning with application to clustering with side-information. In Advances in neural information processing systems (pp. 521-528).
- Shental, N., Hertz, T., Weinshall, D. and Pavel, M., 2002, May. Adjustment learning and relevant component analysis. In European conference on computer vision (pp. 776-790). Springer, Berlin, Heidelberg.
- Davis, J.V., Kulis, B., Jain, P., Sra, S. and Dhillon, I.S., 2007, June. Information-theoretic metric learning. In Proceedings of the 24th international conference on Machine learning (pp. 209-216). ACM.