Data Dependent Distance Metric for Efficient Gaussian Processes Classification

Nayyar A. Zaidi and David McG. Squire

Clayton School of IT Monash University, Clayton VIC 3800, Australia
{nayyar.zaidi,david.squire}@infotech.monash.edu.au

Abstract. The contributions of this work are threefold. First, various metric learning techniques are analyzed and systematically studied under a unified framework to highlight the criticality of data-dependent distance metric in machine learning. The metric learning algorithms are categorized as naive, semi-naive, complete and high-level metric learning, under a common distance measurement framework. Secondly, the connection of feature selection, feature weighting, feature partitioning, kernel tuning, etc. with metric learning is discussed and it is shown that they are all in fact forms of metric learning. Thirdly, it has been shown that the realm of metric learning is not limited to k-nearest neighbor (k-NN) classification, and that a metric optimized in the k-nearest neighbor setting is likely to be effective and applicable in other kernel-based frameworks, for example Support Vector Machine (SVM) and Gaussian Processes (GP) classifiers. We support our hypotheses by tuning the length-scale parameters of GP with metric learning method proposed in k-NN framework. Our empirical results on a huge range of machine learning databases suggest that a metric optimized in the framework of one learning algorithm is likely to be effective in those of others.

1 Introduction

A typical machine learning algorithm takes advantage of training data to discover patterns among observed variables. For example, a learning algorithm can predict the label of a test data point by measuring its similarity with the training data points. Since the data is constituted of features which are not necessarily related by any evident relation, the notion of similarity is not trivially defined. When measuring similarity, one should not necessarily treat all features as being equally important. The problem of learning in high-dimensional machine learning data is actually the problem of estimating the relative importance of the features. Therefore, a lot of research in machine learning has focussed on estimating the relevance of individual and group of features. This relevance determination in fact tunes a data-dependent similarity measure. Most machine learning algorithms either implicitly or explicitly learn this similarity measure on which their performance is critically dependent.

The estimation of a similarity measure can be referred to as 'learning of a data-dependent distance metric' or 'metric learning'. Several techniques such as

feature selection, feature scaling, feature relevance, feature partitioning, kernel tuning, kernel scaling, kernel fusion, distance fusion, etc., that are used for making learning algorithms more efficient, are in fact metric learning methods. That is, they are doing nothing but estimating a measure of similarity that is suitable for the data.

The need to estimate a similarity metric arises because of the heterogeneous structure of the input space in which the data lies. Therefore, the estimation of a data-dependent similarity measure is in fact estimation of the properties of an input space. One can deduce that the performance of learning algorithms will depend on how well these properties are estimated. It is important to make a distinction between the properties of the learning algorithm and the properties of the input space. The properties of an input space depends on data distribution and are independent of any learning algorithm. Therefore, a similarity measure optimized in the framework of one learning algorithm is likely to be applicable and effective in those of other learning algorithms.

As mentioned above, several different terms are used for metric learning in the literature, for example feature selection, feature weighting, feature partitioning, scale estimation, kernel tuning, etc. In this work, we present a novel categorization scheme for scale estimation and metric learning approaches under a common distance framework. The algorithms are categorized as naive, semi-naive, complete and high-level metric learning. By analyzing and categorizing different methods under a common framework, it has been shown that all these methods are in fact special cases of metric learning. We also introduce kernel learning and integrate metric learning in kernel framework. By unveiling the obvious connection between the two frameworks i.e., kernel and metric learning, we propose to extend the realms of metric learning beyond k-NN classification. Therefore, we suggest to learn a distance metric in k-NN framework and use the metric as a kernel in GP framework which is inherently kernel based. GP classifiers typically estimate the kernel parameters using Automatic Relevance Determination (ARD) or cross-validation. On multiple UCIML and other databases, we compare the performance of our proposed method with those of ARD and other state-of-the-art methods. A similar scheme has been proposed for Support Vector Machine (SVM) classifiers in [1, 2].

2 Categorization of Metric Learning Methods

Metric learning is a method that is often applied to improve the performance of a k-NN classifier learns a metric in some Distance Measurement Framework (DMF). An example of a DMF is:

$$d^2(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T A(\boldsymbol{x}_1 - \boldsymbol{x}_2) \tag{1}$$

The Mahalanobis distance between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is a special case of DMF and can be written as:

$$d^2(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \Sigma^{-1}(\boldsymbol{x}_1 - \boldsymbol{x}_2) \tag{2}$$

where the matrix Σ is the covariance matrix of the data [3, 4]. The idea behind Mahalanobis distance formulation is to normalize the effect of different features when measuring distance between the two points. The matrix A in equation 1 can take any form and of course is not restricted to the inverse covariance matrix. We can modify equation 1 as:

$$\begin{aligned} d_A^2(\boldsymbol{x}_1, \boldsymbol{x}_2) &= \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_A^2 \\ &= (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T A(\boldsymbol{x}_1 - \boldsymbol{x}_2) \\ &= (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T L^T L(\boldsymbol{x}_1 - \boldsymbol{x}_2) \quad \text{where} \quad A = L^T L \\ &= (L\boldsymbol{x}_1 - L\boldsymbol{x}_2)^T I(L\boldsymbol{x}_1 - L\boldsymbol{x}_2) \quad \text{where} \quad I = \text{identity matrix} \\ &= d^2(L\boldsymbol{x}_1 - L\boldsymbol{x}_2) \\ &= \|L\boldsymbol{x}_1 - L\boldsymbol{x}_2\|_2^2 \end{aligned} \tag{3}$$

Note, the decomposition of matrix A into $L^T L$ imposes a constraint that matrix A is symmetric positive semidefinite. It can be seen from equation 3 that the effect of matrix A in DMF is nothing but a linear transformation of the data by the matrix L. This is where metric learning algorithms come into play. Metric learning algorithms aim to learn a data-dependent distance metric (matrix A) such that in the transformed space induced by the matrix A, some desirable behavior is expected from the data. Some examples of desirable behavior in the transformed space are: points belonging to the same class tends to be close together, the predicted function changes isotropically, the transformed data lie in the Euclidean space[1], etc. Let us suppose that the matrix A is the inverse covariance matrix. It is desirable in some scenarios to transform the data such that the covariance matrix in the transformed space is the identity matrix. If φ is a matrix whose columns are the eigenvectors of A, and Λ is a diagonal matrix of the corresponding eigenvalues, then we can write matrix L as:

$$L = \varphi \Lambda^{-1/2}$$

The resulting transformation of data induced by the matrix L is typically known as the whitening transformation. The data in the transformed space is uncorrelated. Its covariance matrix is the identity matrix. Another motivation for metric learning is to transform data into a subspace such that the variance of the data is preserved in that subspace. This is again motivated from the dimensionality reduction point of view.

Using equation 1 the similarity between the two feature-vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ can be defined as kernel k:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left(\frac{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^T A(\boldsymbol{x}_1 - \boldsymbol{x}_2)}{\lambda^2} \right) \tag{4}$$

[1] The Euclidean space is equipped with a norm ($\|.\|$) which can be used to define a metric (Euclidean distance). The Euclidean distance between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is:

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| = \sqrt{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^T (\boldsymbol{x}_1 - \boldsymbol{x}_2)}.$$

Let us suppose that the matrix A is the inverse covariance matrix of the data and λ is the smoothing parameter. If the matrix A take the form:

$$A = \begin{pmatrix} \sigma_1^{-2} & 0 & \cdots & 0 \\ 0 & \sigma_2^{-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_P^{-2} \end{pmatrix} \tag{5}$$

we can write equation 4 as:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \begin{pmatrix} \sigma_1^{-2}\lambda^{-2} & 0 & \cdots & 0 \\ 0 & \sigma_2^{-2}\lambda^{-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_P^{-2}\lambda^{-2} \end{pmatrix} (\boldsymbol{x}_1 - \boldsymbol{x}_2) \tag{6}$$

There are two issues that needs to be discussed about equation 6. First, in equation 6, we have assumed the kernel to be isotropic, that is $\lambda = \lambda_1 = \lambda_2 = \cdots = \lambda_P$. This is not necessarily the case and the neighborhood can be adapted in any directions by choosing suitable λ_p. Multiple scaling parameters λ_p can be treated as a vector and equation 6 can be written as:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \begin{pmatrix} \sigma_1^{-2}\lambda_1^{-2} & 0 & \cdots & 0 \\ 0 & \sigma_2^{-2}\lambda_2^{-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_P^{-2}\lambda_P^{-2} \end{pmatrix} (\boldsymbol{x}_1 - \boldsymbol{x}_2) \tag{7}$$

Secondly, the σ_p and λ_p can be merged into a single value resulting in:

$$\begin{aligned} k(\boldsymbol{x}_1, \boldsymbol{x}_2) &= (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \begin{pmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_P \end{pmatrix} (\boldsymbol{x}_1 - \boldsymbol{x}_2) \\ &= (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T A (\boldsymbol{x}_1 - \boldsymbol{x}_2) \end{aligned} \tag{8}$$

So far we have assumed that the matrix A is diagonal. This does not have to be the case. We can learn diagonal and off-diagonal terms of the matrix. The metric learning algorithms studied in nearest neighbor framework [5–8] learn both the diagonal and off-diagonal terms of the matrix A. Therefore, we can rewrite equation 8 as:

$$\begin{aligned} k(\boldsymbol{x}_1, \boldsymbol{x}_2) &= (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1P} \\ a_{21} & a_{22} & \cdots & a_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ a_{P1} & a_{P2} & \cdots & a_{PP} \end{pmatrix} (\boldsymbol{x}_1 - \boldsymbol{x}_2) \\ &= (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T A (\boldsymbol{x}_1 - \boldsymbol{x}_2) \end{aligned} \tag{9}$$

The learning of matrix A in equations 8 and 9 is the goal of metric learning algorithms. As the kernels are used to transform data to a dot-product space $\mathcal{H}$ (using the kernel trick) where the similarity between the two points can be measured by the dot-product, the matrix A in equation 8 controls the transformation of the input space $\mathcal{X}$ into $\mathcal{H}$. A different value of matrix A will in fact transform data into a different space. Therefore, the performance of any classifier leveraging transformation of data by a kernel will depend critically on the choice of the kernel function and the distance matrix A.

Why Learn matrix A? Features in most machine learning data sets are not commensurate, coming from different sources and having different scales. The influence of each feature on the distance is proportional to the dispersion of its values over the training data. For example, if we change the scale of a feature by measuring it in a different unit, the contribution of this feature to distance measurement will change. This will in turn affect classification performance.

The relevance of each feature in predicting the class labels may differ. There is a need to give more weight to those features that are more important and less weight to unimportant features. It can be seen that the influence of an individual feature can by controlled through matrix A. For example, if A is modeled as a diagonal matrix, the influence of feature i can be increased or decreased by modifying a_i (equation 8).

Metric learning has the effect of neighborhood adaptation. Such an adaptive neighborhood is absolutely essential for reducing bias in higher dimensions and helps in alleviating the effects of the curse of dimensionality. A small value for a_i will result in a neighborhood that is elongated in the direction of the i'th feature. And similarly, a large value of a_i will result in neighborhood being constricted in the direction of the i'th feature. Making the value of a_i very small (note this will result in a_i^{-2} to be very large) will result in extending the neighborhood to the entire training data along the i'th feature. As a result of this, all the training data along i'th feature will be assigned equal weights in predicting the class label. This will result in removing the i'th feature from the consideration as the model has become global in i'th feature's direction [9].

2.1 Metric Learning Categorization

The following categorization has also been explored in some detail in [10].

Naive/Semi-naive Metric Learning Feature selection and feature relevance are well studied and effective techniques for ignoring or down-weighting irrelevant or redundant features and making relevant features more explicit. Feature selection can be viewed as a form of metric learning. For example learning a diagonal matrix in equation 2 with some diagonal elements as zero results in those features being ignored, i.e. feature selection. The case of learning a diagonal matrix in equation 2 is categorized as either naive metric learning or semi-naive metric learning. The naive metric learning case arises when we learn the diagonal

terms of matrix A in equation 2 and assume that features are independent from each other. The relevance of each feature is estimated separately. The problem with naive metric learning is that features that are irrelevant when analyzed separately may become relevant when analyzed together with other features. Therefore, there is a need to estimate the relevance of a feature in combination with the other features. This is the motivation behind semi-naive metric learning. Semi-naive metric learning is a modification of naive metric learning where a diagonal matrix is learned by learning the relevance of each feature in combination with the other features.

Naive and Semi-naive metric learning are actually forms of feature selection. Feature selection and weighting have been extensively studied in machine learning [11], but usually it is not explicitly specified that feature selection is in fact learning of a distance metric such that the measurement of distances across certain features is ignored. It should be noted that any feature selection technique can be viewed as a naive or semi-naive metric learning method and vice-versa. Though the two techniques (feature selection and metric learning) have been studied separately, the motivations behind them are exactly the same.

Complete Metric Learning As discussed, the matrix A learned in equation 2 does not have to be diagonal and a full distance matrix can be learned. Complete metric learning deals with the learning of both the diagonal and the off-diagonal terms of the matrix A in the form shown in equation 9. It should be noted that this form of matrix A is more reasonable, in that we can not simply assume that the function changes only along the directions of axes only. Typical metric learning algorithms studied in the k-NN classification framework generally estimate the full distance matrix A and are in fact complete metric learning methods [7, 6, 8].

High-level Metric Learning High-level metric learning deals with learning a data-dependent distance metric in cases where data is represented as feature-sets. Data is represented in the form of feature-sets in those problems when there is a natural partitioning among features, for example object recognition. High-level metric learning is concerned with how to deal with feature-vectors in the feature-sets. Let us suppose that m and n are two types of feature-vector used for object recognition problem (say m representing the 'shape' information and n representing the 'color' information). Let $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ be the feature-vector representing the m and n types of information about the data point $\boldsymbol{x}$ respectively. We can denote the feature-set representing the data point $\boldsymbol{x}$ as:

$$\mathcal{FS}(\boldsymbol{x}) = \{\boldsymbol{x}_m, \boldsymbol{x}_n\} \tag{10}$$

In the following, a brief introduction of the two schemes is given. In the first scheme, we can concatenate the two feature-vectors into one, learn a naive/semi-naive/complete metric with the resulting feature-vectors, and train a single classifier.

In the second scheme, we can treat $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ separately. A separate naive/semi-naive, complete distance metric is learned for each type of feature-vector. Using the learned distance matrices we can train a separate classifier for each type of feature-vector and combine the outputs of the classifiers using some weighting scheme to reach consensus about the label of the object (classifier fusion). Alternatively, we can combine the distances computed using the learned distance matrices for each type of feature-vector (distance fusion) and learn a single classifier. Note that this will have the same effect as measuring distances using a block-diagonal matrix A in equation 2 such that:

$$A = \begin{pmatrix} A_m & 0 \\ 0 & A_n \end{pmatrix} \tag{11}$$

where A_m and A_n are the distance matrices learned for feature-vector of type m and n.

3 Gaussian Processes and Metric Learning

The Gaussian Processes (GP) is a non-linear nonparametric technique that has proven to be very effective for a wide range of classification and regression tasks. We can define GP as a collection of random variables, any finite number of which have a joint Gaussian distribution [12]. The GP is completely specified by its mean and covariance function. We can write GP as:

$$f(\boldsymbol{x}) \sim \mathrm{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')) \tag{12}$$

where $m(\boldsymbol{x})$ and $k(\boldsymbol{x}, \boldsymbol{x}')$ are the mean and covariance functions respectively. Usually, for simplicity, $m(\boldsymbol{x})$ is taken to be zero. The GP can thus be specified completely in terms of its covariance function. The problem of learning with GP is exactly the problem of finding a suitable covariance function (also known as the kernel).

One of the most widely applied covariance functions in GP setting is the isotropic Squared Exponential (SE) covariance function:

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sigma_f^2 \ \exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}{2\lambda^2}\right) + \sigma_n^2 \Delta_{ij} \tag{13}$$

σ_f^2 and σ_n^2 denotes the signal and noise variance in the data and λ is a parameter specifying the characteristic length scale. Informally, the characteristic length scale is the distance one can move in the input space before the function value changes significantly. Of the three parameters $\{\sigma_f^2, \sigma_n^2, \lambda\}$, λ is the most important, as the classification performance of GP classifier depends a great deal on the characteristic length scale parameters. During the course of this work, σ_f^2 and σ_n^2 will not be taken into account. Therefore, the kernel we are interested in has the following form:

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}{2\lambda^2}\right) \tag{14}$$

Using equation 9, the kernel in equation 14 can be generalized as:

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp\left(-(\boldsymbol{x_i} - \boldsymbol{x_j})^T A(\boldsymbol{x_i} - \boldsymbol{x_j})\right) \tag{15}$$

This suggests that, as for any other kernel-based learning algorithm, the problem of learning with GP is actually finding the right specification of matrix A in equation 15.

Algorithm 1 GPML1: GP classification using a data-dependent distance metric.

Require:
- $\boldsymbol{x}_0$: Testing data.
- $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$: Training data.

for $c = 1, 2, \ldots, C$ **do**
- Get a data-dependent distance metric (matrix A) using MEGM for category c such that $A = L^T L$. The kernel learned for category c is:

$$k_{Ac}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\|L\boldsymbol{x}_i - L\boldsymbol{x}_j\|^2\right)$$

end for
- Predict the label of the query point $\boldsymbol{x}_0$ using the pool of learned kernels $\{k_{Ac}\}_{c=1}^C$ in the GP formulation.

Algorithm 2 GPML2: GP classification using a data-dependent distance metric.

Require:
- Testing data: $\boldsymbol{x}_0$.
- Training data: $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$.

- Get a data-dependent distance metric (matrix A) for all C categories using megm such that $A = L^T L$. The kernel learned is:

$$k_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\|L\boldsymbol{x}_i - L\boldsymbol{x}_j\|^2\right)$$

- Predict the label of the query point $\boldsymbol{x}_0$ using the learned kernel k_A in the GP formulation.

The outline of the proposed algorithms 'Gaussian process metric learning' (GPML) is given in algorithms 1 and 2. For multi-class GP classification a one-versus-all strategy is used. GPML1 algorithm learns a different kernel using Mean-Square-Error-Gradient-Minimization (MEGM) metric learning algorithm for each category. MEGM [13] is a simple metric learning algorithm in k-NN

framework that has been shown to perform better than other metric learning algorithms. Whenever the classifier for category c is used to classify a test point $\boldsymbol{x}_0$, the kernel k_{Ac} is used to measure the similarity. GPML2 is a slight variant of GPML1. Instead of learning a separate kernel for each category, GPML2 learns only one kernel k_A for all categories. The learned kernel k_A is used by all c GP classifiers. The two algorithms will be denoted as GPML in the subsequent discussion.

4 Related Work

The unified view of metric and kernel learning has been hinted upon by Hastie et al. [14, section 6.4.1]. Similarly, the idea of defining various techniques such as feature selection, feature relevance, feature scaling, and kernel tuning in terms of metric learning and viewing as of learning the elements of the matrix A in equation 8 has also been mentioned by Chapelle et al. [15, page 133]. Within the GP framework, the representation of length scale parameters as the elements of a matrix has been mentioned by [12, sec. 5.1]. Also, [16–19] and others have also proposed methods for learning the elements of matrix A in GP setting.

The problem of optimizing the length-scale parameters of anisotropic Gaussian kernel in a GP framework was investigated by Neal [16]. The log-likelihood function of a GP model can be given as:

$$\log\ p(\boldsymbol{y}|\theta) = -\frac{1}{2}\boldsymbol{y}^T K^{-1}\boldsymbol{y} - \frac{1}{2}\log\ |K| - \frac{n}{2}\log\ 2\pi \tag{16}$$

where $\boldsymbol{\theta} = (L, \sigma_f^2, \sigma_n^2)$, $\boldsymbol{y}$ is the prediction of GP, and K is the covariance matrix of the input data. L can be optimized by getting the partial derivatives of log likelihood with respect to L and maximize it using gradient based methods. Note, L is a diagonal matrix. It is also straight forward to introduce priors over θ and maximize the log posterior. Maximizing the log posterior to determine length-scale parameters allows the relative importance of different dimensions to be inferred from the data. This represents an example of Automatic Relevance Determination (ARD) in the GP.

Zhou et al. in [17] have used spectral techniques to determine the relevance of each feature. They have proposed a measure called 'Major Bandwidth' to measure the spectral properties of each feature and tuned a metric based on this measure. By learning a metric, data is transformed into a space where isotropic behavior is expected. The authors have reported good results in [17] for human action recognition. A major drawback of their method is the assumption that features are uncorrelated. Similarly, Snelson et al in [18] and Schmidt et al. in [19] have proposed a transformation of data so that it is well modeled as a Gaussian process. The technique proposed in [18] learns a transformation as a part of probabilistic modeling rather that as a pre-processing stage.

The methods proposed in [17, 16, 19] have been the major motivation for our work. Our work is different from all these in the sense that we pre-process data by learning a full distance metric whereas previous work in GP framework has only considered optimizing diagonal terms.

Database	#Data	#Features	#Classes	#Eigen-faces	#(Train,Test)/Class
yalefaces	165	77760	15	50	(4, 7)
yalefacesB	5850	307200	10	20	(10, 20)
caltechfaces	435	47500	29	30	(5, 10)
caltechfacesB	435	47500	2	30	(50, 100)
AT&Tfaces	400	10304	40	150	(5, 5)
USPS	9298	256	10	50	(100, 30)

Table 1. Details of Face and USPS Databases used for classification

5 Experimental Results

In this section, the performance of the proposed GPML algorithms is compared with other standard GP methods, the standard SVM classifier, and the k-NN classifier on various UCIML, faces and digit databases. Faces and USPS data-sets are pre-processed for efficiency. Pre-processing images using PCA is a common approach in object recognition research to reduce dimensionality. This results in vastly reduced computational cost. In our experiments, the results are obtained by reducing the dimensionality of data-set by projecting data on first few eigenfaces. Number of eigenfaces used for each database is given in table 1. No pre-processing is done for UCIML databases. The following methods are compared:

- **KNN:** Simple 1-nearest neighbor classification with the Euclidean distance..
- **SVM:** Standard SVM formulation that is, a multi-class SVM with a Gaussian kernel is used. The $\mathcal{C}$ and σ parameters for the SVM are tuned through cross-validation, that is they are selected from the sets: $\mathcal{C} = \{1, 10, 100, 1000\}$ and $\sigma = \{0.1, 0.5, 1, 2, 3, 5\}$ respectively. A one-versus-all strategy is employed for multi-class classification.
- **GP:** Standard GP classifier with an isotropic Gaussian kernel. The value of σ is optimized through cross-validation. σ is selected from the following values: $\{0.1, 0.5, 1.3, 2.0, 2.0\}$.
- **ISO:** Standard GP classifier with an isotropic Gaussian kernel whose length scale value is optimized through the automatic relevance determination procedure.
- **ARD:** Standard GP classifier with an anisotropic Gaussian kernel. The values of the length scale parameters are tuned through automatic relevance determination procedure.
- **GPML1:** GP with metric learning (algorithm 1). Apart from the distance matrix, no other parameter is learned.
- **GPML2:** GP with metric learning (algorithm 2). Apart from the distance matrix, no other parameter is learned.

For faces and USPS, each experiment is repeated 10 times, mean and standard deviation results are reported. Note that no optimization is done for GPML1 and GPML2. Rasmussen et al. implementation of GP from [20] is used to train GP, ISO and ARD classifiers.

5.1 UCIML Repository Databases

The number of data, features and classes for each UCIML database used is reported in the title of each graph portraying the results. The correctness rate of each method is obtained using 40 rounds of 2-fold cross-validation. Prior to training, features in all the databases were normalized to have zero mean and unit variance.

The comparative performance (correctness rate) of the different methods for various UCIML databases is shown in figure 1. As can be seen from figure 1, GPML2 performed better than GPML1. It appears that training one metric for all classes is more effective than training a separate metric for each class. Out of the 12 databases, GPML2 performed best on nine whereas GPML1 performed best on only three (figures 1(i), 1(j), 1(k)). There are two possible reasons: first, due to MEGM, since the MEGM algorithm suffers from local minima problems. There are more chances that local minima will affect the results in the case of GPML1, as the algorithm has to be run C times. Secondly, possibly training a separate metric for each class leads to over-fitting and hence affects GPML1 performance.

5.2 Face Databases

This section deals with GPML performance evaluation on large databases. Five face (yalefaces, AT&T, yalefacesB, caltechfaces and caltechfacesB) and one (USPS) digit database are used. The details of databases used in this section are given in table 1.

The comparative results (correctness rate) are shown in the figure 2. GPML2 performed best on yalefaces, yalefacesB, caltechfaces and caltechfacesB, whereas GPML1 performed best on AT&Tfaces and USPS digit database. On all six databases, the GP classifier trained with metric learning algorithm performed not only better than the standard GP, but also better than ISO and ARD formulations of GP, where parameters are tuned through the computationally expensive automatic relevance determination procedure. As mentioned, there is no tuning of parameters in the GPML algorithms. It only requires a data-dependent distance metric. This highlights the efficacy of the proposed approach, as the results are comparable with, and in most cases better than, the ISO and ARD formulations. GPML performance is also comparable with SVM performance. It should be noted, however, that SVM is optimized by grid searching over $\mathcal{C}$ and σ values.

5.3 Comparison with Metric Learning in k-NN Framework

In this section, GPML results are compared with the following two metric learning methods in k-NN settings:

– **MEGM-KNN:** MEGM complete metric learning based on the minimization of Mean-Square-Error in k-NN framework as proposed in [13].

– **NCA:** Neighborhood Component Analysis complete metric learning algorithm based on the maximization of margin in k-NN framework [6].

The comparison of the results of GPML algorithms with MEGM and NCA on the face and digit databases is given in figure 3. The GPML and MEGM-KNN methods each performed best on three databases. The results on the UCIML databases are given in figure 4. GPML methods results in significant improvement over the classification accuracy of both MEGM-KNN and NCA. It performed better than all other methods on all but the *tictactoe* data set.

6 Conclusion

In this paper we proposed a unified framework for categorizing metric learning techniques and examined feature selection, feature weighting, kernel tuning, feature fusion under the proposed framework. Even though these techniques are introduced in different scenarios, most of them can be viewed as doing some sort of i.e. naive, semi-naive, complete or high-level metric learning. Categorizing these methods as metric learning provides a unified analysis of these techniques and help the understanding of the learning algorithm. For example, metric is the property of the data and if a metric is optimized for one learning algorithm, it is likely to be effective and applicable in those of others. We investigated the connection between kernel and metric learning and proposed algorithms to train a kernel function for GP by learning a data-dependent distance metric in k-NN framework. Rather than learning length-scale parameters in GP settings like ARD, we learned the metric in k-NN framework. Our empirical results on various UCIML and face data-sets showed that learning a linear transformation of data as a pre-processing step for GP can improve GP performance. And in most cases results are better than the competing method like ARD. As metric learning approaches have only been studied in the Nearest Neighbor perspectives, there is a need to study the impact of learning a linear transformation on GP and SVM classification performance.

References

1. N. Zaidi and D. M. Squire, "Svms and data dependent distance metric," in *Proceedings of the 25th International Conference of Image and Vision Computing New Zealand*, 2010.
2. ——, "Local adaptive svm for object recognition," in *2010 International Conference on Digital Image Computing: Techniques and Applications*, 2010, pp. 196–201.
3. P. Mahalanobis, "On tests and measures of group divergences," *Journal of the Asiatic Society of Bengal*, pp. 541–588, 1930.
4. ——, "On the generalised distance in statistics," in *Proceedings of the National Institute of Sciences of India*, 1936, pp. 49–55.
5. E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," in *Advances in Neural Information and Processing Systems*, 2002, pp. 505–512.

6. J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighborhood component analysis," in *Advances in Neural Information and Processing Systems*, 2005.
7. K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.
8. A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning distance functions using equivalence relation," in *Proceedings of the Twentieth International Conference on Machine Learning*, 2003, pp. 11–18.
9. J. Friedman, "Flexible metric nearest neighbor classification," Department of Statistics, Stanford University, Tech. Rep., 1994.
10. N. Zaidi, "Metric learning and scale estimation in high dimensional machine learning problems with an application to generic object recognition," Ph.D. dissertation, Faculty of Information Technology, Monash University, Clayton, 2011.
11. I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, *Feature Extraction, Foundation and Applications*. Springer, 2004.
12. C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
13. N. Zaidi, D. M. Squire, and D. Suter, "A gradient-based metric learning algorithm for k-nn classifiers," in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*, 2010, pp. 194–203.
14. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, 2001.
15. O. Chapelle, V. Vladimir, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, pp. 131–159, 2002.
16. R. Neal, *Bayesian Learning for Neural Networks*. Springer-Verlag New York, 1996.
17. H. Zhou and D. Suter, "Improved building detection by Gaussian processes classification via feature space rescale and spectral kernel selection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–6.
18. E. Snelson, C. E. Rasmussen, and Z. Ghahrammani, "Warped gaussian processes," in *Advances in Neural Information and Processing Systems*, 2003.
19. A. M. Schmidt and A. O'Hagan, "Bayesian inference for non-stationary spatial covariance structure via spatial deformations," *Journal of the Royal Statistical Society*, vol. 65, no. 3, pp. 743–758, 2003. [Online]. Available: http://www.jstor.org/stable/3647549
20. C. E. Rasmussen and C. K. I. Williams, "Documentation for gpml matlab code," 2007. [Online]. Available: http://www.gaussianprocess.org/gpml/code/matlab/doc/

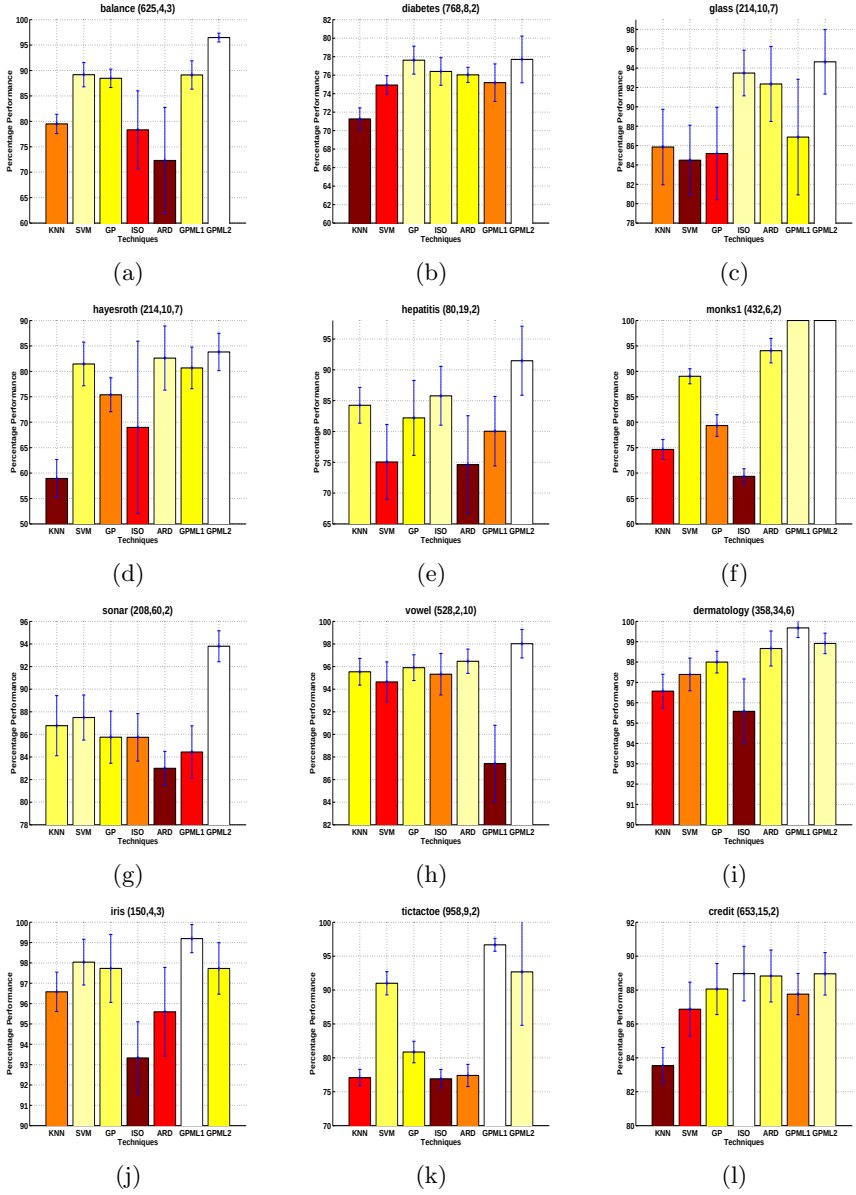

Fig. 1. Comparison of the correctness rate of GPML1 and GPML2 with the standard GP, the ISO and ARD formulation of GP, KNN and standard SVM on various UCIML databases. The mean and standard deviation of the correctness rate over ten runs (each run with different training data) is reported.

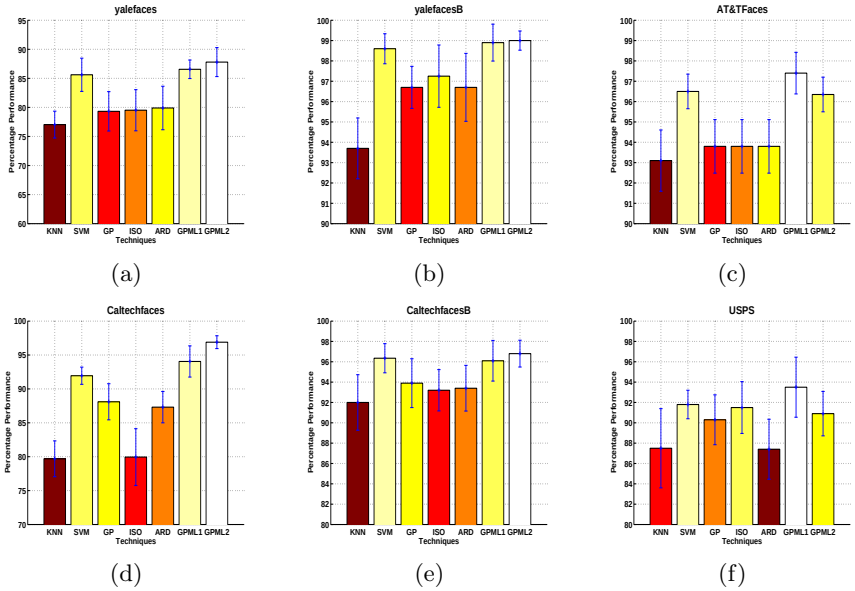

Fig. 2. Comparison of the correctness rate of GPML1 and GPML2 with the standard GP, the ISO and ARD formulation of GP, KNN, and standard SVM on various faces and USPS databases.

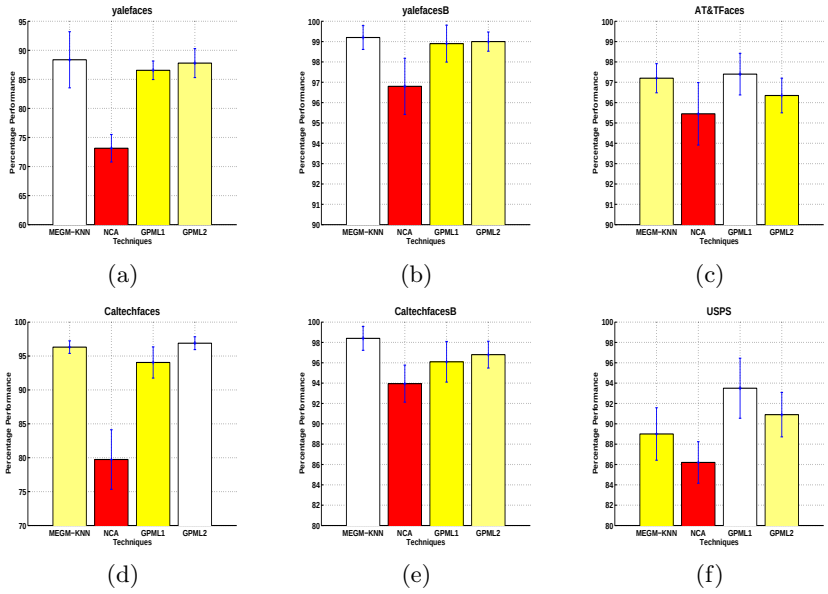

Fig. 3. Comparison of the correctness rate of MEGM-KNN, NCA, GPML1 and GPML2 on faces and USPS databases.

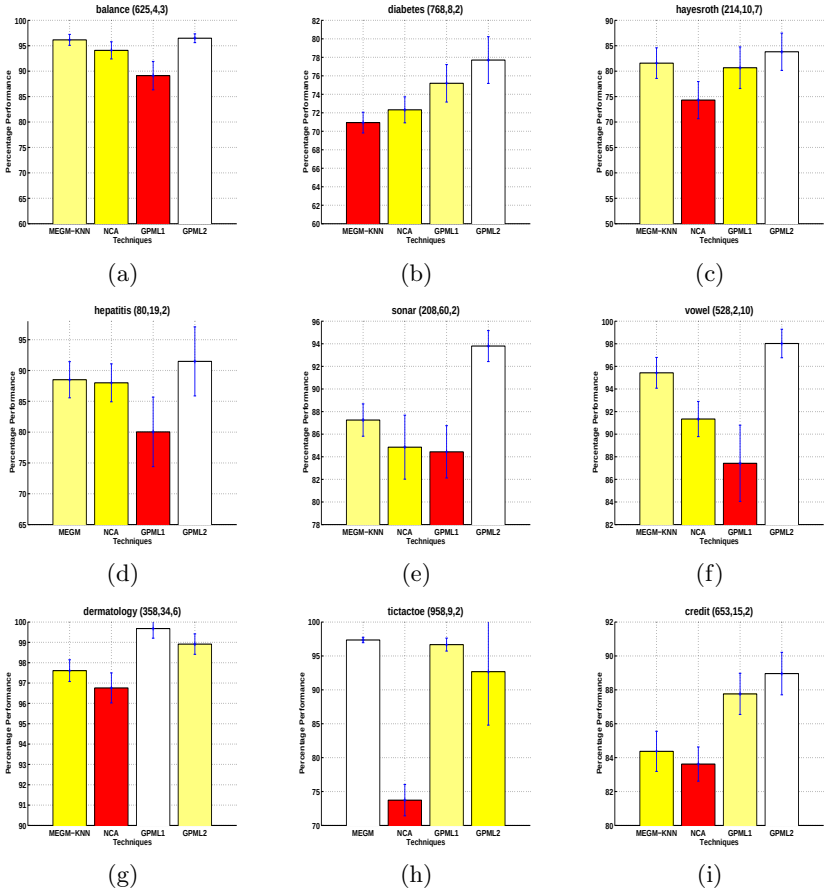

Fig. 4. Comparison of the correctness rate of MEGM-KNN, NCA, GPML1 and GPML2 on UCIML databases.