


# Generative Local Metric Learning for Nearest Neighbor Classification

Yung-Kyun Noh , Byoung-Tak Zhang, and Daniel D. Lee, *Fellow, IEEE*

**Abstract**—We consider the problem of learning a local metric in order to enhance the performance of nearest neighbor classification. Conventional metric learning methods attempt to separate data distributions in a purely discriminative manner; here we show how to take advantage of information from parametric generative models. We focus on the bias in the information-theoretic error arising from finite sampling effects, and find an appropriate local metric that maximally reduces the bias based upon knowledge from generative models. As a byproduct, the asymptotic theoretical analysis in this work relates metric learning to dimensionality reduction from a novel perspective, which was not understood from previous discriminative approaches. Empirical experiments show that this learned local metric enhances the discriminative nearest neighbor performance on various datasets using simple class conditional generative models such as a Gaussian.

**Index Terms**—Metric learning, nearest neighbor classification,  $f$ -divergence, generative-discriminative hybridization

## 1 INTRODUCTION

THE classic dichotomy between generative and discriminative methods for classification in machine learning can clearly be seen in two distinct performance regimes as the number of training examples is varied [1], [2]. Generative models—which employ models to find the underlying distribution  $p(\mathbf{x}|y)$  for discrete class label  $y$  and input data  $\mathbf{x} \in \mathbb{R}^D$ —typically outperform discriminative methods when the number of training examples is small, due to the smaller variance in the generative models which compensates for any possible bias in the models. On the other hand, more flexible discriminative methods—which are interested in a direct measure of  $p(y|\mathbf{x})$ —can accurately capture the true posterior structure  $p(y|\mathbf{x})$  when the number of training examples is large. Thus, given enough training examples, the best performing classification algorithms have typically employed purely discriminative methods.

However, due to the curse of dimensionality when  $D$  is large, the number of data examples may be insufficient for discriminative methods to approach their asymptotic performance limits. In this case, it may be possible to improve discriminative methods by exploiting knowledge from generative models. There has been recent work on hybrid

models showing some improvement [3], [4], [5], but the generative models have mainly been improved through the discriminative formulation. In this work, we consider a very simple discriminative classifier, the nearest neighbor classifier, where the class label of an unknown datum is chosen according to the class label of the *nearest* known datum. The choice of a metric to define *nearest* is then crucial, and we show how this metric can be locally defined based upon information garnered by generative models.

Previous work on metric learning for nearest neighbor classification has focused on a purely discriminative approach. The metric is parameterized by a single global form, which is then optimized on the training data to maximize pairwise separation between dissimilar points and to minimize the pairwise separation of similar points [6], [7], [8], [9], [10], where [10] has been extended to a faster online algorithm [11]. For  $k$ -nearest neighbors, a similar formulation is introduced which is tolerant to imposters making no errors by majority voting [12]. The discriminative learning can be used to find multiple metrics applied at different locations, where each metric is locally discriminative [13] or the metric in different locations is determined by the nonlinear transformation using the combination of regression trees [14]. In discriminative learning, the calculation of the distances for all similar and dissimilar pairs is unavoidable, and it is the main reason that discriminative metric learning does not scale well with many data. The optimization problem is simplified to an eigenvector problem in [15], but the optimization over the simplex of all dissimilar pairs remains a major drawback for large scale data.

Here, we introduce a scalable generative method by analyzing how the problem of learning a metric can be related to reducing the theoretical bias of the nearest neighbor classifier. Though the performance of the nearest neighbor classifier has good theoretical guarantees in the limit of infinite data, finite sampling effects can introduce a bias which can

- Y.-K. Noh is with the School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 08826, Korea.  
E-mail: nohyung@snu.ac.kr.
- B.-T. Zhang is with the School of Computer Science and Engineering, Seoul National University, Gwanak-gu, Seoul 08826, Korea.  
E-mail: btzhang@snu.ac.kr.
- D.D. Lee is with the Department of Electrical & Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104.  
E-mail: ddlee@seas.upenn.edu.

Manuscript received 15 Feb. 2016; revised 30 Dec. 2016; accepted 30 Jan. 2017. Date of publication 7 Feb. 2017; date of current version 12 Dec. 2017.  
Recommended for acceptance by G. Chechik.

For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.  
Digital Object Identifier no. 10.1109/TPAMI.2017.2666151

be minimized by the choice of an appropriate metric. By directly reducing this bias at each point, the classification error is reduced significantly compared to the global class-separating metric.

We show how to choose such a metric by analyzing the probability distribution on nearest neighbors, provided we know the underlying generative models. Analyses of nearest neighbor distributions have been discussed before [16], [17], [18], [19], but we take a simpler approach and derive the metric-dependent term in the bias directly. Minimizing this bias results in a semi-definite programming problem having an analytic solution of an optimal local-metric. In related work, Fukunaga et al. considered learning a metric from a generative setting [20], [21], providing a one-dimensional direction for data projection which is not related to reducing the bias [22]. Moreover, the direction is obtained from nearby data only, where the resulting direction is possibly vulnerable to high-dimensional noise [20]. To avoid the dependency on the local noise, their work was extended to use the averaged metric with the Monte Carlo average [21], but the bias reduction was still not considered. In a different line of research, Jaakkola et al. showed how a generative model can be used to derive a special kernel, called the Fisher kernel [1]. Unfortunately, the Fisher kernel is quite generic and does not necessarily improve nearest neighbor performance.

Our generative approach provides a theoretical relationship between metric learning and the dimensionality reduction problem. In order to find better projections for classification, research on dimensionality reduction has utilized information-theoretic measures such as the Bhattacharyya divergence [23] and mutual information [24], [25]. We argue that these problems can be connected with metric learning for nearest neighbor classification within the general framework of  $f$ -divergences. We will also explain how dimensionality reduction is entirely different from metric learning in the generative approach, whereas in the discriminative setting, it is simply a special case of metric learning where the distances along particular directions have been shrunk to zero.

The remainder of the paper is organized as follows. In Section 2, we begin by comparing the metric dependency of the discriminative and generative approaches for nearest neighbor classification. After we derive the bias due to finite sampling in Section 3, we show in Section 4 how minimizing this bias results in a local metric learning algorithm. In Section 5, we explain how metric learning should be understood in a generative perspective, in particular, its relationship with dimensionality reduction. Experiments on various datasets are presented in Section 6, and we derive an extended algorithm using  $k$ -nearest neighbors for  $k$  greater than one in Section 7. Finally, in Section 8, we conclude with a discussion of future work and possible extensions.

## 2 METRIC AND NEAREST NEIGHBOR CLASSIFICATION

In recent work, determining a good metric for nearest neighbor classification has been thought to be crucial. However, traditional generative analysis of this problem has simply ignored the metric issue with good reason. In this section, we explain why conventional metric learning

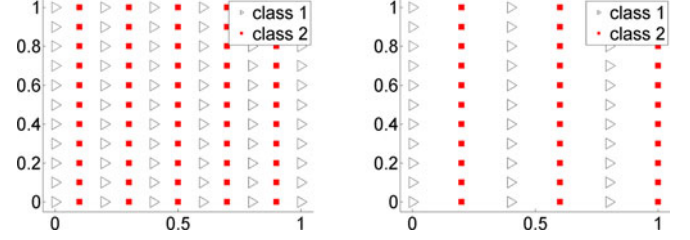


Fig. 1. Metric change for grid data. When the metric is changed by extending the horizontal direction, the separation between data is increased along the horizontal line.

cannot be explored in a traditional generative approach, and we briefly explain how this property can be leveraged for a novel metric learning algorithm.

### 2.1 Metric Learning for Nearest Neighbor Classification

A nearest neighbor classifier determines the label of an unknown datum according to the label of its nearest neighbor. In general, the meaning of the term *nearest* is defined along with the notion of distance in data space. One common choice for this distance is the Mahalanobis distance with a positive definite square matrix  $A \in \mathbb{R}^{D \times D}$  where  $D$  is the dimensionality of data space. In this case, the distance between two points  $\mathbf{x}_1$  and  $\mathbf{x}_2$  is defined as

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T A (\mathbf{x}_1 - \mathbf{x}_2)}, \quad (1)$$

and the nearest datum  $\mathbf{x}_{NN}$  is the one having minimal distance to the test point among labeled training data in  $\{\mathbf{x}_i\}_{i=1}^N$ .

In this classification task, the results are highly dependent on the choice of matrix  $A$ , and prior work [6], [7], [8], [9], [10] has attempted to improve the performance by a better choice of  $A$ . That work assumed the following common heuristic: The training data in different classes should be separated in a new metric space. Given training data, a global  $A$  is optimized such that directions separating different class data are extended, and directions binding together data from the same class are shrunk.

However, in terms of the test results, these conventional methods do not improve the performance dramatically, which will be shown in our later experiments on many datasets, and we show in our theoretical analysis why only small improvements arise or often no improvements at all.

### 2.2 Theoretical Performance of Nearest Neighbor Classifier

The primary motivation for conventional metric learning algorithms has been to seek a separation between sets of data belonging to different classes. For example, if Class 1 data and Class 2 data are distributed along grid lines as in Fig. 1, extending the metric along the horizontal direction obviously helps provide a smaller error for the nearest neighbor classification. Using this motivation, most recent metric learning algorithms have tried to maximize the separation between different classes.

Contrary to the recent metric learning approaches, however, a simple theoretical analysis using a generative model

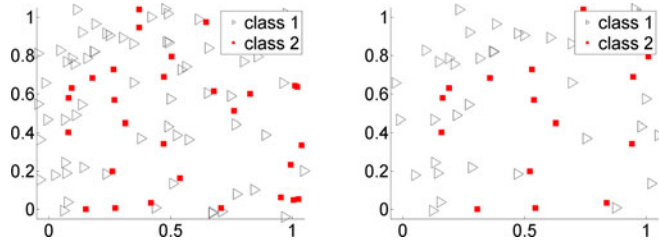


Fig. 2. Metric change for data from a probabilistic generative model. For random data from two different probability density functions, extending the horizontal line simply reduces the densities. There is no visible increase of separation along the horizontal direction between different data any greater than the increase of separation along the vertical direction.

displays no sensitivity to the choice of metric. In Fig. 2, with random data from two different probability densities  $p_1$  and  $p_2$  where  $p_1 > p_2$ , extending the horizontal direction does not increase the separation between data along the horizontal any more than the separation along the vertical direction. Instead, the extension of a particular direction only reduces the scalar densities of both classes together. In a situation where data are dense and the nearest neighbor appears within a small distance where the probability densities are considered to be uniform, the expected classification accuracies are unaffected by changing the metric. In this situation, no metric learning algorithms can improve the nearest neighbor classification. A more detailed analysis of this situation is as follows.

We consider i.i.d. samples generated from two different distributions  $p_1(\mathbf{x})$  and  $p_2(\mathbf{x})$  over the vector space  $\mathbf{x} \in \mathbb{R}^D$ . With infinite samples, the probability of misclassification using a nearest neighbor classifier can be obtained

$$E_{Asymp} = \int \frac{p_1(\mathbf{x})p_2(\mathbf{x})}{p_1(\mathbf{x}) + p_2(\mathbf{x})} d\mathbf{x}. \quad (2)$$

The equation comes from the probability that the labels of the query datum and its nearest neighbor are different. The probability that the query datum has Class 1 is  $p_1/(p_1 + p_2)$ , and the probability that the nearest neighbor has Class 2 is  $p_2/(p_1 + p_2)$  with many data. The multiplication of these two probabilities constitutes the probability of error. Along with the probability that the query has Class 2 and its nearest neighbor has Class 1 as well as the total density of data,  $\frac{p_1+p_2}{2}$  assuming equal priors, the integration of  $\frac{p_1+p_2}{2} \left( \frac{p_1}{p_1+p_2} \frac{p_2}{p_1+p_2} + \frac{p_2}{p_1+p_2} \frac{p_1}{p_1+p_2} \right) = \frac{p_1 p_2}{p_1 + p_2}$  gives the expectation of error with infinite data. The equation is better known by its relationship to an upper bound, twice the optimal Bayes error [20], [21], [26].

By looking at the asymptotic error in a linearly transformed  $\mathbf{z}$ -space, we can show that Eq. (2) is invariant to the change of metric. If we consider a linear transformation  $\mathbf{z} = L^T \mathbf{x}$  using a full rank matrix  $L$ , and the distribution  $q_c(\mathbf{z})$  for  $c \in \{1, 2\}$  in  $\mathbf{z}$ -space satisfying  $p_c(\mathbf{x})d\mathbf{x} = q_c(\mathbf{z})d\mathbf{z}$  and accompanying measure change  $d\mathbf{z} = |L|d\mathbf{x}$ , we see that  $E_{Asymp}$  in  $\mathbf{z}$ -space is unchanged. Since any positive definite  $A$  can be decomposed as  $A = LL^T$ , we can say the asymptotic error remains constant even as the metric shrinks or expands in any spatial directions in data space.

However, the metric independency only arises in the asymptotic limit. When we do not have infinite samples, the

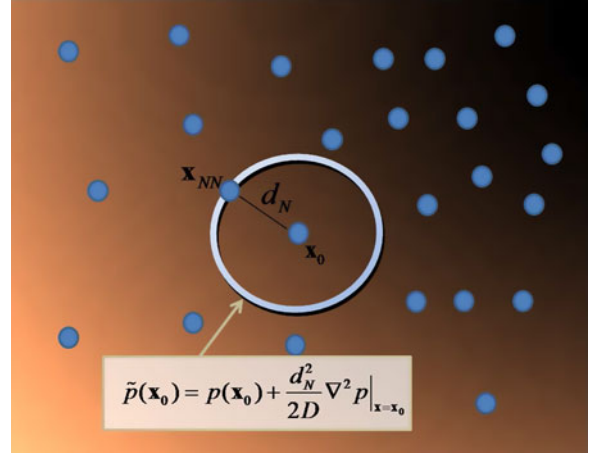


Fig. 3. The nearest neighbor  $\mathbf{x}_{NN}$  appears at a finite distance  $d_N$  from  $\mathbf{x}_0$  due to finite sampling. Given the data distribution  $p(\mathbf{x})$ , the average probability density function over the surface of a  $D$  dimensional hypersphere is  $\tilde{p}(\mathbf{x}_0) = p(\mathbf{x}_0) + \frac{d_N^2}{2D} \nabla^2 p|_{\mathbf{x}=\mathbf{x}_0}$  for small  $d_N$ .

expectation of error is biased in that it deviates from the asymptotic error, and the bias is dependent on the metric. From a theoretical perspective, the asymptotic error is the theoretical limit of expected error, and the bias reduces as the number of samples increases. Since these metric and sample dependencies have not been considered in previous research, the aforementioned class-separating metric will not exhibit performance improvements when the sample number is large.

In the next section, we look at the performance bias associated with finite sampling directly and find a metric that minimizes the bias from the theoretical asymptotic error.

### 3 PERFORMANCE BIAS DUE TO FINITE SAMPLING

Here, we obtain the expectation of nearest neighbor classification error from the distribution of nearest neighbors in different classes. As we consider a finite number of samples, the nearest neighbor from a point  $\mathbf{x}_0$  appears at a finite distance  $d_N > 0$ . This non-zero distance gives rise to the performance difference from its theoretical limit (2). A twice-differentiable probability density function  $p(\mathbf{x})$  is considered, and it is approximated to second order near a test point  $\mathbf{x}_0$

$$p(\mathbf{x}) \simeq p(\mathbf{x}_0) + \nabla p(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0}^T (\mathbf{x} - \mathbf{x}_0) \quad (3)$$

$$+ \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^T \nabla \nabla p(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0} (\mathbf{x} - \mathbf{x}_0) \quad (4)$$

with the gradient  $\nabla p(\mathbf{x})$  and Hessian matrix  $\nabla \nabla p(\mathbf{x})$  defined by taking derivatives with respect to  $\mathbf{x}$ .

Now, under the condition that the nearest neighbor appears at the distance  $d_N$  from the test point, the expectation of the probability  $p(\mathbf{x}_{NN})$  at a nearest neighbor point is derived by averaging the probability over the  $D$ -dimensional hypersphere of radius  $d_N$ , as in Fig. 3. Here, the point  $\mathbf{x}_{NN}$  is the location of the nearest neighbor for a given test point  $\mathbf{x}_0$ , and the set of  $\mathbf{x}_{NN}$  apart from  $\mathbf{x}_0$  by  $d_N$  is of interest.

After averaging  $p(\mathbf{x}_{NN})$ , the gradient term disappears, and the resulting expectation is obtained as the sum of the probability at  $\mathbf{x}_0$  and a residual term containing the



Laplacian of  $p$ , where the detailed derivation is shown in the Appendix. We replace this expected probability by  $\tilde{p}(\mathbf{x}_0)$

$$E_{\mathbf{x}_{NN}}[p(\mathbf{x}_{NN})|d_N, \mathbf{x}_0] \simeq p(\mathbf{x}_0) + \quad (5)$$

$$\frac{1}{2} E_{\mathbf{x}_{NN}}[(\mathbf{x} - \mathbf{x}_0)^\top \nabla \nabla p(\mathbf{x})(\mathbf{x} - \mathbf{x}_0) | \|\mathbf{x} - \mathbf{x}_0\|^2 = d_N^2] \\ = p(\mathbf{x}_0) + \frac{d_N^2}{2D} \cdot \nabla^2 p|_{\mathbf{x}=\mathbf{x}_0} \equiv \tilde{p}(\mathbf{x}_0), \quad (6)$$

where the scalar Laplacian  $\nabla^2 p(\mathbf{x})$  is given by the sum of the eigenvalues of the Hessian  $\nabla \nabla p(\mathbf{x})$ .

If we look at the expected error for two-class classification, it is the expectation of the probability that the test point and its neighbor are labeled differently. In other words, the expectation error  $E_{NN}$  is the expectation of  $e(\mathbf{x}, \mathbf{x}_{NN}) = p(C=1|\mathbf{x})p(C=2|\mathbf{x}_{NN}) + p(C=2|\mathbf{x})p(C=1|\mathbf{x}_{NN})$  over both the distribution of  $\mathbf{x}$  and the distribution of nearest neighbor  $\mathbf{x}_{NN}$  for a given  $\mathbf{x}$

$$E_{NN} = E_{\mathbf{x}} \left[ E_{\mathbf{x}_{NN}} [e(\mathbf{x}, \mathbf{x}_{NN}) | \mathbf{x}] \right]. \quad (7)$$

Using two class conditional density functions  $p_c(\mathbf{x})$  for  $c=1,2$ , we replace the posteriors  $p(C|\mathbf{x})$  and  $p(C|\mathbf{x}_{NN})$  as  $p_c(\mathbf{x})/(p_1(\mathbf{x}) + p_2(\mathbf{x}))$  and  $p_c(\mathbf{x}_{NN})/(p_1(\mathbf{x}_{NN}) + p_2(\mathbf{x}_{NN}))$ , respectively. We then approximate the expectation of the posterior  $E_{\mathbf{x}_{NN}}[p(C|\mathbf{x}_{NN})|d_N, \mathbf{x}]$  at a fixed distance  $d_N$  from test point  $\mathbf{x}$  using  $\tilde{p}_c(\mathbf{x})/(\tilde{p}_1(\mathbf{x}) + \tilde{p}_2(\mathbf{x}))$ . If we expand  $E_{NN}$  with respect to  $d_N$  and take the expectation using the decomposition,  $E_{\mathbf{x}_{NN}}[f] = E_{d_N}[E_{\mathbf{x}_{NN}}[f|d_N]]$ , then the expected error is given to the leading order by

$$E_{NN} \simeq \int \frac{p_1 p_2}{p_1 + p_2} d\mathbf{x} + \frac{E_{d_N}[d_N^2]}{4D} \int \frac{1}{(p_1 + p_2)^2} [p_1^2 \nabla^2 p_2 + p_2^2 \nabla^2 p_1 - p_1 p_2 (\nabla^2 p_1 + \nabla^2 p_2)] d\mathbf{x}, \quad (8)$$

where  $p_1$  and  $p_2$  are the abbreviations of the class conditional density functions  $p_1(\mathbf{x})$  and  $p_2(\mathbf{x})$ , respectively.

When  $E_{d_N}[d_N^2] \rightarrow 0$  with an infinite number of samples, this error converges to the asymptotic limit in Eq. (2) as expected. The residual term can be considered as the finite sampling bias of the error discussed earlier. Under the coordinate transformation  $\mathbf{z} = L^\top \mathbf{x}$  and the distributions  $p(\mathbf{x})$  on  $\mathbf{x}$  and  $q(\mathbf{z})$  on  $\mathbf{z}$ , we see that this bias term is dependent upon the choice of metric  $A = LL^\top$

$$\frac{1}{(q_1 + q_2)^2} [q_1^2 \nabla^2 q_2 + q_2^2 \nabla^2 q_1 - q_1 q_2 (\nabla^2 q_1 + \nabla^2 q_2)] d\mathbf{z}, \quad (9)$$

$$= \frac{1}{(p_1 + p_2)^2} \text{tr} \left[ A^{-1} (p_1^2 \nabla \nabla p_2 + p_2^2 \nabla \nabla p_1 - p_1 p_2 (\nabla \nabla p_1 + \nabla \nabla p_2)) \right] d\mathbf{x}, \quad (10)$$

which is derived using  $p(\mathbf{x})d\mathbf{x} = q(\mathbf{z})d\mathbf{z}$  and  $|L|\nabla^2 q = \text{tr}[A^{-1}\nabla \nabla p]$  where the determinant of the second equation results from taking the derivative of  $p(\mathbf{x})$  with respect to  $\mathbf{z}$  using  $\mathbf{z} = L^\top \mathbf{x}$ . The expectation of squared distance  $E_{d_N}[d_N^2]$  in Eq. (8) is related to the determinant  $|A|$  by  $E_{d_N}[d_N^2] \propto |A|^{\frac{1}{D}}$ , because  $E_{d_N}[d_N^2]$  is proportional to  $(\frac{p_1+p_2}{2})^{-\frac{2}{D}}$  as in [22]. By using  $A$  with fixed determinant  $|A| = 1$ , we do not have to consider the change of  $E_{d_N}[d_N^2]$ . Therefore, finding the metric that minimizes the quantity given in Eq. (9) at each point is equivalent to minimizing the metric-dependent bias in Eq. (8).

#### 4 REDUCING DEVIATION FROM THE ASYMPTOTIC PERFORMANCE

Finding the local metric that minimizes the bias can be formulated as a semi-definite programming (SDP) problem of minimizing the squared residual with respect to a positive semi-definite metric  $A$

$$\min_A (\text{tr}[A^{-1}B])^2 \quad \text{s.t.} \quad |A| = 1, A \succeq 0, \quad (11)$$

where the matrix  $B$  at each point is

$$B = p_1^2 \nabla \nabla p_2 + p_2^2 \nabla \nabla p_1 - p_1 p_2 (\nabla \nabla p_1 + \nabla \nabla p_2). \quad (12)$$

Here  $B$  depends upon the given test point  $\mathbf{x}$ , and the corresponding optimal  $A$  is determined locally as well. However the  $B$ 's at different points come from the same generative model explaining  $p_1$  and  $p_2$  at different points. For a given generative model, the densities and Hessians  $p_1(\mathbf{x})$ ,  $p_2(\mathbf{x})$ ,  $\nabla \nabla p_1|_{\mathbf{x}}$ , and  $\nabla \nabla p_2|_{\mathbf{x}}$  are determined at each testing point, and the  $B$  at the same point is determined correspondingly.

The optimization problem in Eq. (11) is a simple SDP problem having an analytical solution. The solution shares the eigenvectors with  $B$ . Let  $\Lambda_+ \in \mathbb{R}^{d_+ \times d_+}$  and  $\Lambda_- \in \mathbb{R}^{d_- \times d_-}$  be the diagonal matrices containing the positive and negative eigenvalues of  $B$ , respectively. If  $U_+ \in \mathbb{R}^{D \times d_+}$  contains the eigenvectors corresponding to the eigenvalues in  $\Lambda_+$  and  $U_- \in \mathbb{R}^{D \times d_-}$  contains the eigenvectors corresponding to the eigenvalues in  $\Lambda_-$ , we use a scaled solution given by

$$A_{\text{opt}} = \beta [U_+ \ U_-] \begin{pmatrix} d_+ \Lambda_+ & 0 \\ 0 & -d_- \Lambda_- \end{pmatrix} [U_+ \ U_-]^\top, \quad (13)$$

with appropriate  $\beta$  satisfying  $|A_{\text{opt}}| = 1$ . Eq. (13) is optimal when either all the eigenvalues of the matrix  $B$  are positive ( $A_{\text{opt}} = \beta U_+ \Lambda_+ U_+^\top$ ) or all eigenvalues are negative ( $A_{\text{opt}} = -\beta U_- \Lambda_- U_-^\top$ ), which can be obtained by direct differentiation of the objective function. If the matrix  $B$  has both positive and negative eigenvalues, Eq. (13) produces the minimum zero objective function  $(\text{tr}[A_{\text{opt}}^{-1}B])^2 = 0$ . However in this case, note that Eq. (13) is not the unique solution with zero objective function. With this particular form, the solution Eq. (13) provides a continuous change at the boundary where the eigenvalues change signs.

The solution  $A_{\text{opt}}$  is a local metric since we assume that the nearest neighbor is close to the test point satisfying Eq. (4). In principle, distances should then be defined as geodesic distances using this local metric on a Riemannian manifold. However, this is computationally difficult, so we use the

surrogate distance  $A = \gamma I + A_{opt}$  and treat  $\gamma$  as a regularization parameter that is learned in addition to the local metric  $A_{opt}$ .

The multiway extension of this problem is straightforward. The asymptotic error with  $C$ -class distributions can be extended to  $\frac{1}{C} \sum_{c=1}^C \int (p_c \sum_{j \neq c} p_j) / (\sum_i p_i) d\mathbf{x}$  using the posteriors of each class, and it replaces  $B$  in Eq. (12) by the extended matrix

$$B = \sum_{i=1}^C \nabla \nabla p_i \left( \sum_{j \neq i} p_j^2 - p_i \sum_{j \neq i} p_j \right). \quad (14)$$

## 5 METRIC LEARNING IN GENERATIVE MODELS

Once the Hessian matrices  $\nabla \nabla p_c$  are given for all classes  $c = 1, \dots, C$ , the metric can be obtained from the eigenvalues and the eigenvectors of Eq. (14). Since  $p_c$  is the underlying density function, the Hessian cannot be obtained directly but has to be estimated. In this work, a simple standard Gaussian density function is used for  $p_c$  to learn the generative model with parameters  $\mu_c$  and  $\Sigma_c$ , the mean vector and the covariance matrix, respectively for each class  $c$ . The Hessian of the Gaussian density function is given by the expression

$$\nabla \nabla p_c(\mathbf{x}) = p_c(\mathbf{x}) \left[ \Sigma_c^{-1} (\mathbf{x} - \mu_c)(\mathbf{x} - \mu_c)^\top \Sigma_c^{-1} - \Sigma_c^{-1} \right]. \quad (15)$$

This expression is then used with estimated parameters to learn the local metric at each point  $\mathbf{x}$ . Because we use a generative model to obtain the metric, we call our learning method Generative Local Metric Learning (GLML). As a generative model, any twice-differentiable parametric or nonparametric density functions can be used, but empirically, a coarse Gaussian model for each class works well. For a Gaussian model, the computational complexity of calculating parameters is  $O(C(ND^2 + D^3))$  for  $C$ -class data with  $N$  number of  $D$ -dimensional training data, and the metric matrix is obtained with cost  $O(D^3)$  for solving the eigenvector problem for each testing datum. The algorithm can easily scale with the number of training data  $N$ . The algorithm using the Gaussian generative model is presented in Algorithm 1.

---

### Algorithm 1. Generative Local Metric Learning for Nearest Neighbor Classification

---

**Input:** data  $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$  and a point  $\mathbf{x}$

**Output:** classification output  $\hat{y}(\mathbf{x})$

**Procedure:**

- 1: Estimate mean vector  $\mu_c$  and covariance matrix  $\Sigma_c$  of each class  $c = \{1, \dots, C\}$  from  $\mathcal{D}$ .
  - 2: Use the estimated parameters  $\hat{\mu}_c, \hat{\Sigma}_c$  and obtain  $B$  matrix (Eq. (14)) at the point  $\mathbf{x}$ .
  - 3: Use the eigenvectors and the eigenvalues of  $B$  and obtain the metric matrix  $A$  in Eq. (13).
  - 4: Use metric  $A$ , perform the nearest neighbor classification, and obtain  $\hat{y}(\mathbf{x})$  with the new distance in Eq. (1).
- 

## 5.1 Dimensionality Reduction and Metric Learning

Traditional metric learning methods can be understood as being purely discriminative—in other words, they do not

consider the class-conditional density functions,  $p_1(\mathbf{x})$  or  $p_2(\mathbf{x})$ . Those methods are focused on maximizing the separation of data belonging to different classes. In general, their motivations are compared to the supervised dimensionality reduction methods, which try to find a low dimensional space where the separation between classes is maximized. The motivation of previous metric learning is not so different from this dimensionality reduction, thus dimensionality reduction has often been considered as a similar problem seeking a separation between classes but with a special constraint where the metric in particular directions is forced to be zero.

When a generative approach is considered, however, the relationship between dimensionality reduction and metric learning is different. As in the discriminative case, dimensionality reduction in generative models tries to obtain class separation in a transformed space. It assumes particular parametric distributions (typically Gaussians) and uses a criterion to maximize the separation [23], [24], [25], [27]. One general form of these criteria is the  $f$ -divergence (also known as Csiszer's general measure of divergence), which can be defined with respect to a convex function  $\phi(t)$  for  $t \in \mathbb{R}$  [28]

$$F(p_1(\mathbf{x}), p_2(\mathbf{x})) = \int p_1(\mathbf{x}) \phi \left( \frac{p_2(\mathbf{x})}{p_1(\mathbf{x})} \right) d\mathbf{x}. \quad (16)$$

Examples of using this divergence include the Bhattacharyya divergence  $\int \sqrt{p_1(\mathbf{x})p_2(\mathbf{x})} d\mathbf{x}$  when  $\phi(t) = \sqrt{t}$  and the KL-divergence  $-\int p_1(\mathbf{x}) \log \left( \frac{p_2(\mathbf{x})}{p_1(\mathbf{x})} \right) d\mathbf{x}$  when  $\phi(t) = -\log(t)$ . The use of mutual information between data and labels can be understood as an extension of KL-divergence. The well known Linear Discriminant Analysis is a special example of a Bhattacharyya criterion when we assume two-class Gaussians sharing the same covariance matrices.

Unlike dimensionality reduction, we cannot use the  $f$ -divergence as criteria for metric learning because any  $f$ -divergence is metric-invariant. Instead, we can find the relationship from the asymptotic error in Eq. (2). This asymptotic error is related to  $f$ -divergence by  $E_{Asymp} = 1 - F(p_1, p_2)$  where the  $f$ -divergence is associated with  $\phi(t) = 1/(1+t)$ . The  $f$ -divergence here can be called the asymptotic accuracy of nearest neighbor classification. While dimensionality reduction uses  $f$ -divergences as maximizing criteria with generative information, in metric learning in contrast, we view the problem as an estimation problem, and we optimize the metric for better estimation of the  $f$ -divergence. In other words, a better estimation can be interpreted as a matter of reconstructing the asymptotic nearest neighbor classification results—which are less than twice the Bayes error—with finite data, and this interpretation provides an understanding of how we can discriminate this novel idea from the concept underlying the dimensionality reduction problem.

## 5.2 Generative Information for Classification

Classification using generative models typically tries to make a model for class-conditional probability densities  $p_1(\mathbf{x})$  and  $p_2(\mathbf{x})$  and use the criterion  $p_1(\mathbf{x})/p_2(\mathbf{x}) \geq 1$  for classification. Nearest neighbors have the probability

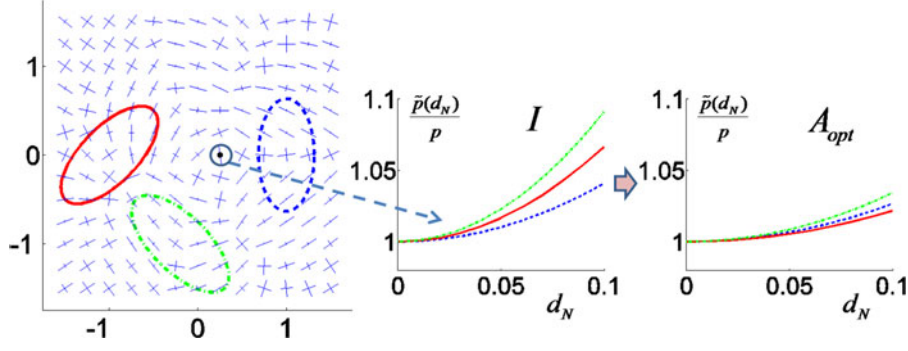


Fig. 4. Optimal local metrics are shown on the left for three example Gaussian distributions in a 5-dimensional space. The projected 2-dimensional distributions are represented as ellipses (one standard deviation from the mean), while the remaining 3 dimensions have isotropic distributions. The local  $\tilde{p}/p$  of the three classes is plotted on the right using a euclidean metric  $I$  and for the optimal metric  $A_{opt}$ . The solution  $A_{opt}$  tries to keep the ratio  $\tilde{p}/p$  over the different classes as similar as possible for different distance  $d_N$ .

density information where the higher density tends to give the shorter nearest neighbor distance. However, the classification using nearest neighbors from finite samples deviates from that of using infinite samples because the probability density itself deviates at the location of nearest neighbors.

If we can keep the probability densities intact at the location of nearest neighbors  $\mathbf{x}_{NN}$ , the expected error of a point  $\mathbf{x}$  is still  $\frac{p_1}{p_1+p_2} \left| \frac{p_2}{p_1+p_2} \right|_{\mathbf{x}_{NN}} + \frac{p_2}{p_1+p_2} \left| \frac{p_1}{p_1+p_2} \right|_{\mathbf{x}_{NN}} = \frac{p_1(\mathbf{x})p_2(\mathbf{x})}{(p_1(\mathbf{x})+p_2(\mathbf{x}))^2}$ , where  $p_1(\mathbf{x})$  and  $p_2(\mathbf{x})$  are the probability densities at the point of interest  $\mathbf{x}$ , and the nearest neighbor classification will achieve the asymptotic classification error.

Our motivation for reducing Eq. (9) provides an alternative way of reconstructing such asymptotic errors. By reformulating Eq. (12) into  $(p_2 - p_1)(p_2 \nabla^2 p_1 - p_1 \nabla^2 p_2)$ , we can see that the optimal metric tries to minimize the difference between  $\frac{\nabla^2 p_1}{p_1}$  and  $\frac{\nabla^2 p_2}{p_2}$ . If  $\frac{\nabla^2 p_1}{p_1} \approx \frac{\nabla^2 p_2}{p_2}$ , this also implies

$$\frac{\tilde{p}_1}{p_1} \approx \frac{\tilde{p}_2}{p_2}, \quad (17)$$

for  $\tilde{p} = p + \frac{d_N^2}{2D} \nabla^2 p$ , the average probability at a distance  $d_N$  in (6). Thus, the algorithm tries to keep the ratio of the average probabilities  $\tilde{p}_1/\tilde{p}_2$  at a distance  $d_N$  as similar as possible to the ratio of probabilities  $p_1/p_2$  at the test point. Although we cannot keep the probability density constant at a distance, our metric can stabilize the ratio between the average of the probability densities at a distance. This means that the expected nearest neighbor classification at a distance  $d_N$  will be least biased again even though the density information is changed. Fig. 4 shows how the learned local metric  $A_{opt}$  varies at a point  $\mathbf{x}$  for a 3-class Gaussian example and, at a specific point, how the ratio of  $\tilde{p}/p$  is kept as similar as possible between different classes.

In the following experiments, we show how the actual nearest neighbor classification can benefit from the analysis. At issue is how reliably we can estimate the Hessian from data. It turns out that by applying a simple model, such as a single Gaussian for each class, we can see a significant increase in performance for many datasets. A coarse model captures the overall curvature structure of the underlying probability densities, providing guidelines to the nearest neighbor classifier on which directions to focus for reconstructing asymptotic results.

## 6 EXPERIMENTS

We apply our algorithm for learning a local metric to synthetic and various real datasets and see how well it improves nearest neighbor classification performance. We compare the performance of our method—GLML using class-conditional Gaussian model in Algorithm 1—with recent metric learning discriminative methods which report state-of-the-art performance on a number of datasets. These include Information-Theoretic Metric Learning (ITML)<sup>1</sup> [6], Boost Metric<sup>2</sup> (BM) [9], and Largest Margin Nearest Neighbor (LMNN)<sup>3</sup> [10]. As recently proposed methods, we also present results using Parametric Local Metric Learning (PLML) [13] and Distance Metric Learning with Eigenvalue Optimization (DML-eig) [15]. We use the implementations downloaded from the corresponding authors' websites. We also compare with a local metric given by the Fisher kernel [1] assuming a single Gaussian for the generative model and using the location parameter to derive the Fisher information matrix. The metric from the Fisher kernel was not originally intended for nearest neighbor classification, but it is the only other reported algorithm that learns a local metric from generative models.

### 6.1 Experiments with Gaussian Data

For the synthetic dataset, we generated data from two-class random Gaussian distributions having two fixed means. The covariance matrices are generated from random orthogonal eigenvectors and random eigenvalues. Experiments were performed varying the input dimensionality, and the classification accuracies are shown in Fig. 5a along with the results of the other algorithms. We used 500 test points and an equal number of training examples. The experiments were performed with 20 different realizations, and the results were averaged. As the dimensionality grows, the original nearest neighbor performance degrades because of the high dimensionality. However, we see that the proposed local metric substantially outperforms the discriminative nearest neighbor performance in a high dimensional space. We note that this example is ideal for GLML, and it shows considerable improvement compared to the other methods.

1. <http://userweb.cs.utexas.edu/~pjain/itml/>

2. <http://code.google.com/p/boosting/>

3. <http://www.cse.wustl.edu/~kilian/Downloads/LMNN.html>



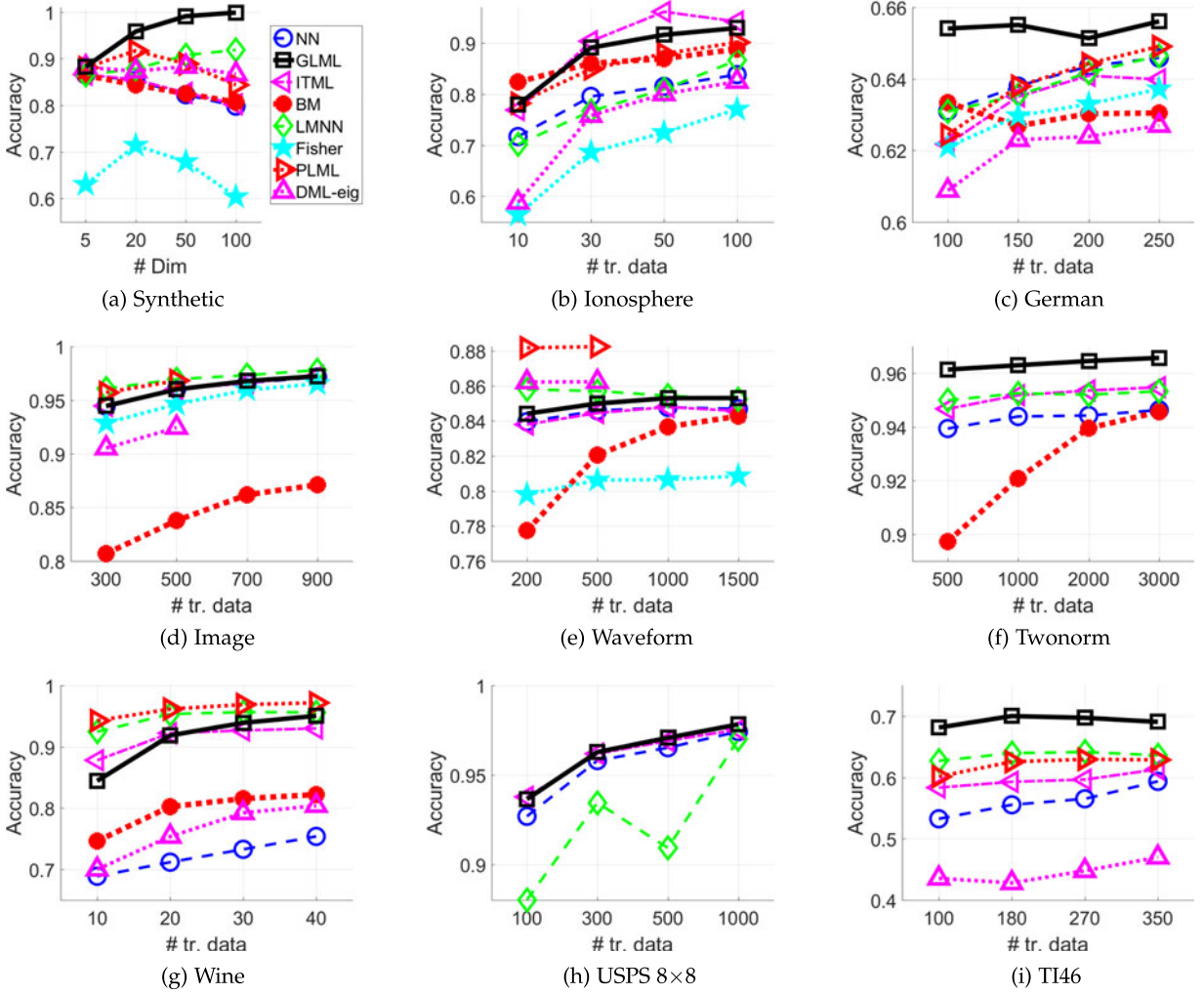


Fig. 5. (a) Gaussian synthetic data with varying dimensionalities. As the number of dimensions grows, accuracies of many methods degrade. On the other hand, GLML continues to improve its performance vastly over other methods. Experiments (b)~(h) on benchmark datasets vary the number of training data per class. The speech dataset (i) TI46 is pronounced by eight men and eight women. The Fisher kernel and BM are omitted for (f)~(i) and (h),(i) respectively, since their performances are much worse than the nearest neighbor classifier with euclidean and other metrics. DML-eig and PLML are (partly) missing in (d)~(f),(h),(i) and (d)~(f),(h) respectively, due to the complexity of the algorithms.

## 6.2 Experiments with Real Data

The other experiments consist of the following benchmark datasets: UCI machine learning repository<sup>4</sup> datasets (Ionosphere, Wine), and the IDA benchmark repository<sup>5</sup> (German, Image, Waveform, Twonorm). We also used the USPS handwritten digits and the TI46 speech dataset. For the USPS data, we resized the images to  $8 \times 8$  pixels and trained on the 64-dimensional pixel vector data. For the TI46 dataset, the examples consist of spoken sounds pronounced by eight different men and eight different women. We chose the pronunciation of 10 digits (“zero” to “nine”), and performed a 10 class digit classification task. Ten different filters in the Fourier domain were used as features to preprocess the acoustic data. The experiments were done on 20 data sampling realizations for Twonorm and TI46, 10 for USPS, 200 for Wine, and 100 for the others. For DML-eig and PLML, the algorithms are not scalable with the number of data, so the results are not fully obtained for datasets with many data.

Our algorithm has two hyperparameters. The metric matrix is regularized as  $A = A_{opt} + \gamma I$  with the identity matrix  $I$ . We also used a regularized estimated covariance matrix  $\Sigma = \hat{\Sigma} + \alpha I$  using  $\alpha$  with the maximum-likelihood estimated  $\hat{\Sigma}$ . Both  $\gamma$  and  $\alpha$  are chosen by cross-validation on a subset of the training data, then fixed for testing. The covariance regularization  $\alpha$  is fixed for all classes in each experiment.

In Fig. 5a, we show the results for the synthetic data with varying dimensionalities. Many algorithms show decreasing accuracies for higher dimensionalities, where this decrease is due to the curse of dimensionality. In high dimensional space, distance becomes non-informative, and many algorithms do not succeed at extracting information living in the high dimensional space. On the other hand, two Gaussians in a high dimensional space become more separable than the Gaussians in a lower dimensional space. Our algorithm captures the information effectively and increases the accuracy substantially over other algorithms.

From the results shown in Fig. 5b, 5c, 5d, 5e, 5f, 5g, 5h, 5i, our local metric algorithm generally outperforms most of

4. <http://archive.ics.uci.edu/ml/>

5. <http://www.fml.tuebingen.mpg.de/Members/raetsch/benchmark>

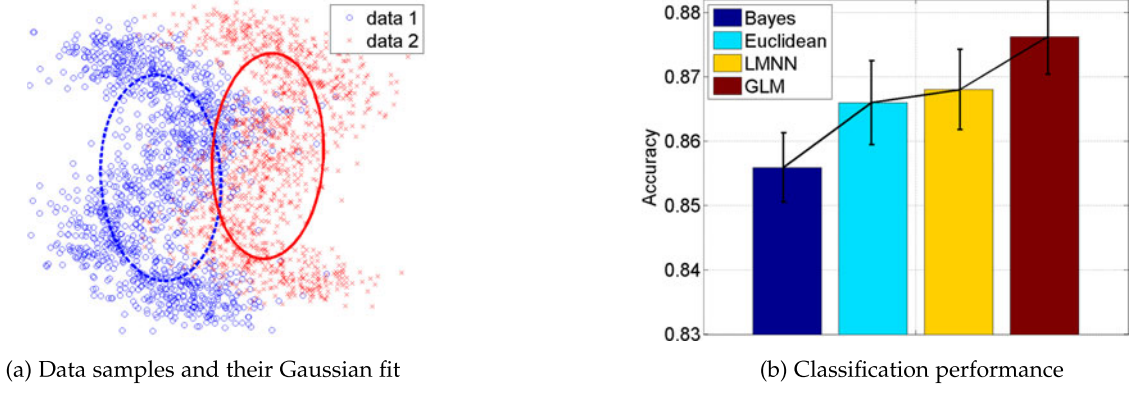


Fig. 6. (a) Non-Gaussian data for nearest neighbor classification. (b) Classification performance of using Gaussian generative models and nearest neighbor performance with euclidean, LMNN, and GLML metrics.

the other metrics across most of the datasets. On quite a number of datasets, many of the other methods do not outperform the original euclidean nearest neighbor classifier. This is because these methods (all except PLML and Fisher metric) use a global metric upon which performance can be improved only little. On the other hand, the local metric derived from simple Gaussian distributions always shows a performance gain over the naive nearest neighbor classifier. In contrast, using the Bayes rule with these simple Gaussian generative models often results in a very poor performance. The computational time using a local metric is also very competitive, since the underlying SDP optimization has a simple spectral solution. This is in contrast to other methods which numerically solve for a global metric using an SDP over the data points.

For some algorithms in our experiments, we did not report the results because those algorithms do not scale with the amount of data. DML-eig obtains the metric matrix using an iterative iteration of solving two problems: an eigenvector problem of  $D \times D$  matrix with data dimensionality  $D$  and an optimization problem over the set of dissimilar pairs where the number of pairs scales with  $O(N^2)$  with the number of data  $N$ . We used two hyperparameters—the smoothing parameter in the algorithm and the number of nearest neighbors for the dissimilar pair-selection, and they are selected using cross-validation with training data. We used  $10^{-3}$  as a tolerance value. PLML tries to solve two objective functions at the same time: one is the local smoothness objective function of the metric matrices at different points, and the other is the discriminative objective function of the metric matrices at anchor points. The amount of computation scales with  $O(NMD + N^2M)$  for smoothness optimization with  $M$  number of anchor points and  $O(MD^3 + MND^2 + TD)$  with  $T$  number of similar and dissimilar triplets. The number of triplets can scale with  $O(N^3)$ .

Our algorithm can easily scale with the number of training data. Once the parameter of the generative model is estimated, the calculation cost of the metric is  $O(D^3)$  for solving the eigenvector problem for each testing datum. In addition, considering that the Hessian matrix in Eq. (15) divided by  $p(\mathbf{x})$ ,  $\frac{\nabla \nabla p(\mathbf{x})}{p(\mathbf{x})} = [\Sigma^{-1}(\mathbf{x} - \mu)(\mathbf{x} - \mu)^\top \Sigma^{-1} - \Sigma^{-1}]$ , is the summation of the rank-one matrix  $[\Sigma^{-1}(\mathbf{x} - \mu)(\mathbf{x} - \mu)^\top \Sigma^{-1}]$  and the negative-inverse-covariance matrix  $[-\Sigma^{-1}]$ , the eigenvalues

and the eigenvectors can be obtained using the rank-one modification method having complexity  $O(D^2)$  [29], [30]. For example, the two-class problem uses the eigenvalues and the eigenvectors of  $p_1^2 \nabla \nabla p_2 + p_2^2 \nabla \nabla p_1 - p_1 p_2 (\nabla \nabla p_1 + \nabla \nabla p_2) = (p_2 - p_1) p_1 p_2 (\frac{\nabla \nabla p_1}{p_1} - \frac{\nabla \nabla p_2}{p_2})$ , and because

$$\frac{\nabla \nabla p_1}{p_1} - \frac{\nabla \nabla p_2}{p_2} = [\Sigma_1^{-1}(\mathbf{x} - \mu_1)(\mathbf{x} - \mu_1)^\top \Sigma_1^{-1} - \Sigma_1^{-1}] \quad (18)$$

$$- [\Sigma_2^{-1}(\mathbf{x} - \mu_2)(\mathbf{x} - \mu_2)^\top \Sigma_2^{-1} - \Sigma_2^{-1}], \quad (19)$$

once the eigenvalues and the eigenvectors of  $[\Sigma_2^{-1} - \Sigma_1^{-1}]$  are obtained, they can be rank-one modified by the two rank one matrices:  $[\Sigma_1^{-1}(\mathbf{x} - \mu_1)(\mathbf{x} - \mu_1)^\top \Sigma_1^{-1}]$  and  $[\Sigma_2^{-1}(\mathbf{x} - \mu_2)(\mathbf{x} - \mu_2)^\top \Sigma_2^{-1}]$  at each point of interest  $\mathbf{x}$  where a test datum is located.

### 6.3 Experiments with an Inconsistent Generative Model

In the experiments with synthetic and real data, we use a generative model to obtain the curvature information for the metric. In particular, experiments with Gaussian data in Fig. 5a show that the curvature information from the consistent model obviously improves the performance. Other experiments in this section show whether the accuracy is improved when the generative model does not contain the true underlying probability density but captures only the rough configuration of data.

Fig. 6a shows a two-dimensional projection of the two-class five-dimensional data generated from two Gaussian mixtures where each class cannot be modeled using only a single Gaussian. The remaining three dimensionalities have an isotropic Gaussian, where any useful information for discrimination is in the presented two-dimensional space. The ellipsoids for two classes represent the Gaussian density functions which were fit to data using maximum likelihood estimation.

As a test using the Gaussian generative model  $\hat{p}_1(\mathbf{x})$  and  $\hat{p}_2(\mathbf{x})$  with estimated parameters, the classification can be performed by comparing  $\hat{p}_1(\mathbf{x}) \gtrless \hat{p}_2(\mathbf{x})$ . In Fig. 6b ‘Bayes’ shows the accuracy of this classification which is outperformed by the nearest neighbor classification, ‘euclidean’, due to the



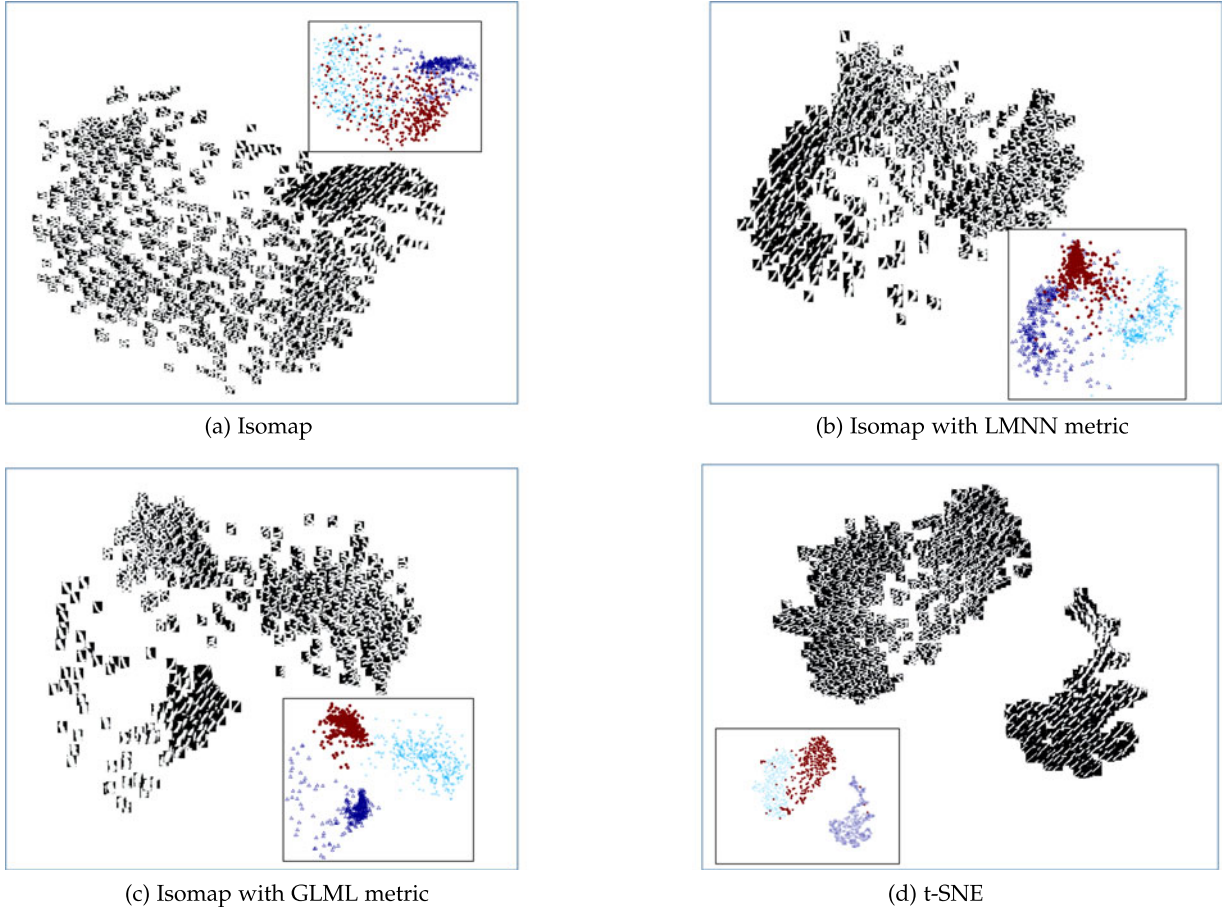


Fig. 7. Isomap embedding with (a) euclidean metric, (b) Large Margin Nearest Neighbor (LMNN) metric, (c) GLML metric, and (d) t-Distributed Stochastic Neighbor Embedding (t-SNE) embedding. The inset figures represent the label distribution.

inconsistency of the model. Compared with the LMNN result, GLML greatly improves the accuracy although GLML uses the information from the inconsistent generative model.

The result shows that even though the generative model is not flexible enough for capturing the detailed structure of the true densities, the rough information of the configuration helps improve the accuracy of the nearest neighbor classification. In our experiments, the generative model using one Gaussian captures the global structure of data although the data themselves are not exactly from a single Gaussian. This compensates for the fundamental weakness of the nearest neighbor methods which use only local data. In particular, in high dimensional space, the distance information is easily corrupted by a small amount of ambient noise but by using the metric from the global structure, the noise is efficiently reduced. Initially, nearest neighbor classification is designed to capture the detailed boundaries of different classes by using local information, and the information necessary for this method is a rough guideline of directions which should have more focus. We understand that the cascade of this generative and discriminative information is cooperative because each model captures different information, the global rough structure for generative and the local detailed structure for discriminative.

#### 6.4 Embedding Experiments

In this section, we show the Isomap embeddings when different metrics are used. Isomap captures local distance

information and finds the global manifold structure preserving the local distances [31]. In our experiment, we used Isomap for the embedding of the three handwritten digits “1,” “3,” and “8” from MNIST [32]. When the euclidean distance is used in Fig. 7a, the Isomap captures the slope structure of digits along the horizontal axis. The slope of writing on the right-hand side tends to be rotated more clockwise than the slope on the left-hand side. In terms of the separation between classes, three different digits are roughly separated in this embedding, but there is an overlap throughout the embedding as well.

LMNN aims at finding the metric maximally separating different classes. The LMNN embedding in Fig. 7b shows a clearer separation between different digits than the euclidean embedding in Fig. 7a, but it still shows overlap around the boundaries.

If we consider the motivation of GLML, the method does not intentionally separate different classes. Instead, it finds the metric that reconstructs the asymptotic classification results. However, in Fig. 7c, different classes are more clearly separated for GLML than LMNN embedding, and the slope structure is shown along the horizontal axis as well. In fact, it shows an unfolding of the manifold where the slope of characters is rotated clockwise as we move along the horizontal axis from left to center, and it is rotated back counterclockwise when we move from center to the right. This unfolding is due to the metric that separates different classes while

simultaneously preserving the slope structure. Here we used a single Gaussian generative model for each class as before.

A result of the conventional embedding method, t-Distributed Stochastic Neighbor Embedding (t-SNE), is also presented in Fig. 7d for comparison. The embedding using t-SNE is known to find clustering structure without actual labels [33]. According to the figure, t-SNE finds three big clusters, each of which corresponds to a single digit. However, in the clearly separated clusters, digits belonging to different clusters are found in many points, which can be compared with our Gaussian GLML results in which different classes are clearly separated.

## 7 EXTENSION TO $k$ -NEAREST NEIGHBOR CLASSIFICATION

In this section, we derive the GLML metric for  $k$ -nearest neighbor classification. Compared to nearest neighbor classification,  $k$ -nearest neighbor classification needs to combine the expectations of the combinatorial cases based on the majority voting strategy. Using the expected error of these combinatorial cases and their deviations from the asymptotic expectations, we derive the metric for the  $k$ -nearest neighbor classification.

First, the asymptotic expected error for classifying a datum at  $\mathbf{x}$  is as follows where the class-posteriors are  $\mu_1 = \frac{p_1(\mathbf{x})}{p_1(\mathbf{x})+p_2(\mathbf{x})}$  and  $\mu_2 = \frac{p_2(\mathbf{x})}{p_1(\mathbf{x})+p_2(\mathbf{x})}$

$$\begin{aligned} \epsilon_k(\mathbf{x}) &= \mu_1 \cdot \sum_{\{c_1, \dots, c_k | \sum_{i=1}^k \delta_{c_i=1} < \sum_{i=1}^k \delta_{c_i=2}\}} \mu_1^{\sum_{i=1}^k \delta_{c_i=1}} \mu_2^{\sum_{i=1}^k \delta_{c_i=2}} \\ &\quad + \mu_2 \cdot \sum_{\{c_1, \dots, c_k | \sum_{i=1}^k \delta_{c_i=1} > \sum_{i=1}^k \delta_{c_i=2}\}} \mu_1^{\sum_{i=1}^k \delta_{c_i=1}} \mu_2^{\sum_{i=1}^k \delta_{c_i=2}} \\ &= \mu_1 \sum_{l=0}^{(k-1)/2} \binom{k}{l} \mu_1^l \mu_2^{k-l} \\ &\quad + \mu_2 \sum_{l=0}^{(k-1)/2} \binom{k}{l} \mu_1^{k-l} \mu_2^l, \end{aligned} \quad (20)$$

where  $\mu_1$  and  $\mu_2$  are the posteriors that  $\mathbf{x}$  has Class 1 and Class 2 labels, respectively,  $c_j$  is the class of  $j$ th nearest neighbor, and  $\{c_1, \dots, c_k | \sum_{i=1}^k \delta_{c_i=1} \geq \sum_{i=1}^k \delta_{c_i=2}\}$  is the set of all  $k$  number of labels where the number of Class 1 is greater (or less) than the number of Class 2 labels. Error occurs when  $\mathbf{x}$  has Class 1 and the majority voting classifies  $\mathbf{x}$  as Class 2 with the condition  $\sum_{i=1}^k \delta_{c_i=1} < \sum_{i=1}^k \delta_{c_i=2}$ . The expectation of this error is the first term of Eq. (20). Another error occurs when  $\mathbf{x}$  has Class 2 and the majority voting classifies  $\mathbf{x}$  as Class 1 which gives the second term.

We consider Eq. (20) and the perturbed probability,  $\tilde{p}_c(\mathbf{x}) = p_c(\mathbf{x}) + \alpha \nabla^2 p_c(\mathbf{x})$ , the expected probability at the location of nearest neighbors. Here,  $\alpha$  is a constant to represent the deviation of the probability density at the point of nearest neighbors. To find the deviation of the posteriors  $\mu_1$  and  $\mu_2$ , we can use a constant  $\alpha$  regardless of Class  $c$  because we consider the same nearest neighbor position for both classes.

For each posterior  $\tilde{\mu}_1$  and  $\tilde{\mu}_2$  at nearest neighbor point, we use  $\tilde{p}_1$  and  $\tilde{p}_2$  to calculate the deviation

$$\begin{aligned} \frac{\partial \tilde{\mu}_1}{\partial \alpha} &= \frac{\partial}{\partial \alpha} \frac{\tilde{p}_1}{\tilde{p}_1 + \tilde{p}_2} \\ &= \frac{1}{(p_1 + p_2)^2} [\nabla^2 p_1 (p_1 + p_2) - p_1 (\nabla^2 p_1 + \nabla^2 p_2)] \quad (21) \\ &= \frac{p_1 p_2}{(p_1 + p_2)^2} \left( \frac{\nabla^2 p_1}{p_1} - \frac{\nabla^2 p_2}{p_2} \right), \end{aligned}$$

$$\begin{aligned} \frac{\partial \tilde{\mu}_2}{\partial \alpha} &= \frac{\partial}{\partial \alpha} \frac{\tilde{p}_2}{\tilde{p}_1 + \tilde{p}_2} \\ &= -\frac{p_1 p_2}{(p_1 + p_2)^2} \left( \frac{\nabla^2 p_1}{p_1} - \frac{\nabla^2 p_2}{p_2} \right), \end{aligned} \quad (22)$$

for each of the nearest neighbors. Now we consider the modified posteriors at  $i$ th nearest neighbor  $\tilde{\mu}_{c_i}^{(i)}$  of Class  $c_i$ , then the product of probabilities in Eq. (20) can be modified as

$$\begin{aligned} \text{Eq. (20)} &= \mu_1 \sum_{l=0}^{(k-1)/2} \binom{k}{l} \mu_1^l \mu_2^{k-l} + \mu_2 \sum_{l=0}^{(k-1)/2} \binom{k}{l} \mu_1^{k-l} \mu_2^l \\ &\rightarrow \mu_1 \sum_{l=0}^{(k-1)/2} \sum_{\{c_1, \dots, c_k | \sum_{i=1}^k \delta_{c_i=1} = l\}} \prod_{i=1}^k \tilde{\mu}_{c_i}^{(i)} \\ &\quad + \mu_2 \sum_{l=0}^{(k-1)/2} \sum_{\{c_1, \dots, c_k | \sum_{i=1}^k \delta_{c_i=2} = l\}} \prod_{i=1}^k \tilde{\mu}_{c_i}^{(i)} \\ &= \text{Eq. (20)} \\ &\quad + \mu_1 \sum_{j=1}^k \alpha^{(j)} \sum_{l=0}^{(k-1)/2} \sum_{\{c_1, \dots, c_k | \sum_{i=1}^k \delta_{c_i=1} = l\}} \prod_{i=1}^k \frac{\mu_{c_i}^{(i)}}{\mu_{c_j}^{(j)}} \frac{\partial \mu_{c_j}^{(j)}}{\partial \alpha^{(j)}} \\ &\quad + \mu_2 \sum_{j=1}^k \alpha^{(j)} \sum_{l=0}^{(k-1)/2} \sum_{\{c_1, \dots, c_k | \sum_{i=1}^k \delta_{c_i=2} = l\}} \prod_{i=1}^k \frac{\mu_{c_i}^{(i)}}{\mu_{c_j}^{(j)}} \frac{\partial \mu_{c_j}^{(j)}}{\partial \alpha^{(j)}} \end{aligned} \quad (24)$$

with constant  $\alpha^{(j)}$  for  $j$ th nearest neighbor. Here,  $\alpha^{(j)}$  is the constant that varies according to the location of each nearest neighbor for  $j = 1, \dots, k$ . All terms in the deviation contain one differentiation  $\frac{\partial \mu_{c_j}^{(j)}}{\partial \alpha^{(j)}}$  in each term, and according to the Eq. (21) and the Eq. (22), all terms share a single metric dependent equation  $\left( \frac{\nabla^2 p_1}{p_1} - \frac{\nabla^2 p_2}{p_2} \right)$ . In order to minimize the amount of deviation for  $k$ -nearest neighbor classification, a metric has to be chosen to minimize  $\left( \frac{\nabla^2 p_1}{p_1} - \frac{\nabla^2 p_2}{p_2} \right)^2$ . Thus, the metric with the least deviation of the nearest neighbor classification also minimizes the deviation for the  $k$ -nearest neighbor classification as well.

### 7.1 Experiments with Caltech 101 and Caltech 256

In this section, we present experimental results with Caltech 101 and Caltech 256 [34]. The dataset contains images of 101 and 256 different data objects, respectively. For Caltech 101, we first use the bag of words of the Speeded Up Robust Feature (SURF) [35] and show in Fig. 8(a) how our algorithm

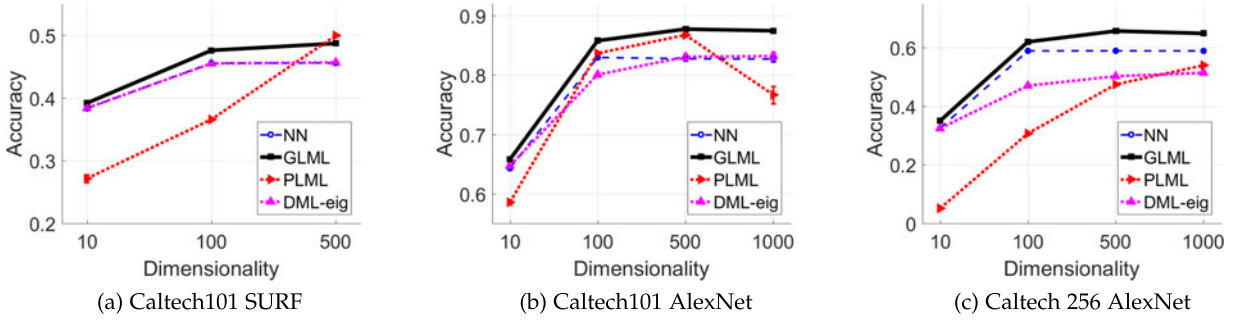


Fig. 8. Nearest neighbor classification results with various metrics and 5 nearest neighbors. Nearest neighbor search uses GLML, PLML, and DML-eig metric for (a) Caltech 101 dataset with SURF feature, (b) Caltech 101 dataset with trained AlexNet hidden layer, and (c) Caltech 256 dataset with trained AlexNet hidden layer.

increases the classification accuracy of the nearest neighbor classification. In this figure, we report the average and standard error of the 5-fold experiments with  $k = 5$  for the 70 percent randomly selected training data and the remaining 30 percent testing data for each class. The SURF features are obtained from training data, and they are clustered to produce a bag of words. Using the centers of the 500 clusters, we changed all images into 500 dimensional vectors, and from the 500 dimensional data, we extracted 10 and 100 dimensional data using principal component analysis (PCA). We show the results of the original nearest neighbor classification and the nearest neighbor classification using GLML, PLML, and DML-eig metrics with data of 101 labels. According to Fig. 8a, DML-eig shows very similar accuracy to the original 5-nearest neighbor classification, and PLML shows some increase with 500 dimensionalities but with the cost of calculation time which was 33 hours compared with 20 minutes of the total GLML experiment time.

In Fig. 8b, we used the features from the AlexNet with trained weights using ImageNet [36]. The 4096 features of the second fully-connected layer are used, and the dimensionalities are reduced to 10, 100, 500, and 1000 using PCA. For the same Caltech 101 data used in Fig. 8a, this setting showed accuracy above 80 percent by using simple 5-nearest neighbor classification, and with the GLML metric, the accuracy is increased with the largest margin among the results from various metrics. In Fig. 8c, we applied the same setting to Caltech 256 data with more data as well as more labels. We again used the average of 5-fold experiments, but with Caltech 256, we used 90 percent for train and 10 percent for test because the 10 percent test set included enough data. In all experiments, GLML increases the original 5-nearest neighbor classification accuracy, while other recent metric learning methods fail to increase the accuracy with large numbers of data and labels.

## 8 CONCLUSIONS

In our study, we show how a local metric for nearest neighbor classification can be learned using generative models. Our experiments show improvement over competitive methods on a number of experimental datasets. The learning algorithm is derived from an analysis of the asymptotic performance of the nearest neighbor classifier, such that the optimal metric minimizes the bias of the expected performance of the classifier. This connection to generative models is very powerful, as it can easily be extended to include missing data—one of the

major advantages of generative models in machine learning. The use of kernel methods in conjunction with our method is also straightforward. We use simple Gaussians for the generative models but could easily extend them to include other possibilities as well, such as mixture models, hidden Markov models, or other dynamic generative models.

## APPENDIX A AVERAGE PROBABILITY DENSITY AT THE NEAREST NEIGHBOR POINT

In this appendix, we derive the average probability density on the surface of hypersphere where a nearest neighbor is found, as in Fig. 3. Due to the local curvature structure of the density function, the expectation of the probability density  $p(\mathbf{x}_{NN})$  at a nearest neighbor point is perturbed, and here, we derive the deviation in Eq. (6) from the density at the point of interest  $\mathbf{x}_0$ ,  $p(\mathbf{x}_0)$ .

First, we consider the volume of a  $D$ -dimensional hypersphere with radius  $r$

$$V = \frac{\pi^{\frac{D}{2}} r^D}{\Gamma(\frac{D}{2} + 1)}, \quad (25)$$

and the volume of the surface with an infinitesimal thickness  $dr$

$$\left( \frac{d}{dr} \frac{\pi^{\frac{D}{2}} r^D}{\Gamma(\frac{D}{2} + 1)} \right) dr = \frac{D \pi^{\frac{D}{2}} r^{D-1}}{\Gamma(\frac{D}{2} + 1)} dr. \quad (26)$$

Finally, we consider the integration of  $(\mathbf{x}' - \mathbf{x})^\top \nabla \nabla p(\mathbf{x})(\mathbf{x}' - \mathbf{x})$  over the space with Hessian matrix,  $\nabla \nabla p(\mathbf{x})$ . We use the eigenvectors of Hessian,  $\mathbf{u}_1, \dots, \mathbf{u}_D$  as bases with the coefficients  $\beta_1, \dots, \beta_D$  satisfying  $\mathbf{x}' - \mathbf{x} = \beta_1 \mathbf{u}_1 + \dots + \beta_D \mathbf{u}_D$ , to represent the integration as

$$\int (\mathbf{x}' - \mathbf{x})^\top \nabla \nabla p(\mathbf{x})(\mathbf{x}' - \mathbf{x}) d\mathbf{x}' \quad (27)$$

$$= \int \dots \int_{\sqrt{\beta_1^2 + \dots + \beta_D^2} \leq d} (\beta_1 \mathbf{u}_1 + \dots + \beta_D \mathbf{u}_D)^\top \nabla \nabla p(\mathbf{x}) (\beta_1 \mathbf{u}_1 + \dots + \beta_D \mathbf{u}_D) d\beta_1 \dots d\beta_D \quad (28)$$

$$= \int \dots \int_{\sqrt{\beta_1^2 + \dots + \beta_D^2} \leq d} \sum_{i=1}^D \beta_i^2 \lambda_i d\beta_1 \dots d\beta_D, \quad (29)$$

with the eigenvalues of Hessian,  $\lambda_1, \dots, \lambda_D$ .



With the integration of Eq. (29) and the derivative with respect to the radius, we obtain the average Hessian on the surface

$$\frac{\pi^{\frac{D}{2}} r^{D+1} \sum \lambda_i}{\Gamma(\frac{D}{2} + 1)} dr / \text{Volume of the surface} \quad (30)$$

$$= \frac{\pi^{\frac{D}{2}} r^{D+1} \sum \lambda_i}{\Gamma(\frac{D}{2} + 1)} dr / \frac{D \pi^{\frac{D}{2}} r^{D-1}}{\Gamma(\frac{D}{2} + 1)} dr \quad (31)$$

$$= \frac{r^2}{D} \nabla^2 p|_{x_0}. \quad (32)$$

Here, we used the Laplacian as the sum of Hessian eigenvalues. In other words,  $\nabla^2 p = \text{tr}[\nabla \nabla p] = \sum \lambda_i$ .

Now, the average of the probability density on the surface of a hypersphere can be derived as follows: We consider a  $D$ -dimensional hypersphere with radius  $d$  with the center of the hypersphere  $\mathbf{x}$ , and the point on the surface  $\mathbf{x}'$  which is close to  $\mathbf{x}$ . The probability density  $p(\mathbf{x}')$  can be approximated using the second order Taylor series

$$p(\mathbf{x}') \simeq p(\mathbf{x}) + \nabla p(\mathbf{x})^\top (\mathbf{x}' - \mathbf{x}) + \frac{1}{2} (\mathbf{x}' - \mathbf{x})^\top \nabla \nabla p(\mathbf{x}) (\mathbf{x}' - \mathbf{x}). \quad (33)$$

We consider the average of  $p(\mathbf{x}')$  on the surface of the hypersphere

$$\begin{aligned} \langle p(\mathbf{x}') \rangle_{S(D,r)} &= \langle p(\mathbf{x}) + \nabla p|_{\mathbf{x}}^\top (\mathbf{x}' - \mathbf{x}) \\ &\quad + \frac{1}{2} (\mathbf{x}' - \mathbf{x})^\top \nabla \nabla p|_{\mathbf{x}} (\mathbf{x}' - \mathbf{x}) \rangle_{S(D,r)}, \end{aligned} \quad (34)$$

where  $S(D, r)$  is the surface of the  $D$ -dimensional hypersphere with radius  $r$ .

The average of  $\langle p(\mathbf{x}) \rangle_{S(D,r)} = p(\mathbf{x})$  because  $p(\mathbf{x})$  is constant, and the second term  $\langle \nabla p|_{\mathbf{x}}^\top (\mathbf{x}' - \mathbf{x}) \rangle_{S(D,r)} = 0$  due to symmetry. The average of the third term is  $\langle \frac{1}{2} (\mathbf{x}' - \mathbf{x})^\top \nabla \nabla p|_{\mathbf{x}} (\mathbf{x}' - \mathbf{x}) \rangle_{S(D,r)} = \frac{r^2}{2D} \nabla^2 p|_{x_0}$  yielding the expectation of the probability density on the surface

$$\langle p(\mathbf{x}') \rangle_{S(D,r)} = p(\mathbf{x}) + \frac{r^2}{2D} \nabla^2 p|_{\mathbf{x}}. \quad (35)$$

## ACKNOWLEDGMENTS

Y.-K. Noh is supported by grants from the U.S. Air Force Office of Scientific Research, the National Security Research Institute in Korea, and the SNU-MAE BK21+ program. B.-T. Zhang is supported by grants from the U.S. Air Force Office of Scientific Research, and Korea government (IITP-R0126-16-1072, KEIT-10044009, KEIT-10060086). D. D. Lee is supported by grants from the U.S. National Science Foundation, the U.S. Department of Transportation, the U.S. Air Force Office of Scientific Research, the U.S. Army Research Laboratory, and the U.S. Office of Naval Research.

## REFERENCES

[1] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Proc. Advances Neural Inf. Process. Syst.*, 1998, pp. 487–493.

[2] A. Ng and M. Jordan, "On discriminative versus generative classifiers: A comparison of logistic regression and naive Bayes," in *Proc. Advances Neural Inf. Process. Syst.*, 2001, pp. 841–848.

[3] S. Lacoste-Julien, F. Sha, and M. Jordan, "DiscLDA: Discriminative learning for dimensionality reduction and classification," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 897–904.

[4] J. Lasserre, C. Bishop, and T. Minka, "Principled hybrids of generative and discriminative models," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2006, pp. 87–94.

[5] R. Raina, Y. Shen, A. Ng, and A. McCallum, "Classification with hybrid generative/discriminative models," in *Proc. Advances Neural Inf. Process. Syst.*, 2004, pp. 545–552.

[6] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon, "Information-theoretic metric learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 209–216.

[7] A. Globerson and S. Roweis, "Metric learning by collapsing classes," in *Proc. Advances Neural Inf. Process. Syst.*, 2006, pp. 451–458.

[8] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Proc. Advances Neural Inf. Process. Syst.*, 2005, pp. 513–520.

[9] C. Shen, J. Kim, L. Wang, and A. van den Hengel, "Positive semi-definite metric learning with boosting," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 1651–1659.

[10] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proc. Advances Neural Inf. Process. Syst.*, 2006, pp. 1473–1480.

[11] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," *J. Mach. Learn. Res.*, vol. 11, pp. 1109–1135, 2010.

[12] S. Trivedi, D. McAllester, and G. Shakhnarovich, "Discriminative metric learning by neighborhood gerrymandering," in *Proc. Advances Neural Inf. Process. Syst.*, 2014, pp. 3392–3400.

[13] J. Wang, A. Kalousis, and A. Woznica, "Parametric local metric learning for nearest neighbor classification," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 1601–1609.

[14] D. Kiedem, S. Tyree, F. Sha, G. R. Lanckriet, and K. Q. Weinberger, "Non-linear metric learning," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 2573–2581.

[15] Y. Ying and P. Li, "Distance metric learning with eigenvalue optimization," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 1–26, Jan. 2012.

[16] M. N. Goria, N. N. Leonenko, V. V. Mergel, and P. Inverardi, "A new class of random vector entropy estimators and its applications in testing statistical hypotheses," *J. Nonparametric Statist.*, vol. 17, no. 3, pp. 277–297, 2005.

[17] F. Perez-Cruz, "Estimation of information theoretic measures for continuous random variables," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 1257–1264.

[18] Q. Wang, S. R. Kulkarni, and S. Verdú, "A nearest-neighbor approach to estimating divergence between continuous random vectors," in *Proc. IEEE Int. Symp. Inf. Theory*, 2006, pp. 242–246.

[19] Q. Wang, S. R. Kulkarni, and S. Verdú, "Divergence estimation for multidimensional densities via k-nearest-neighbor distances," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2392–2405, May 2009.

[20] K. Fukunaga and R. Short, "The optimal distance measure for nearest neighbour classification," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 622–627, Sep. 1981.

[21] K. Fukunaga and T. E. Flick, "An optimal global nearest neighbour measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 6, no. 3, pp. 314–318, May 1984.

[22] R. R. Snapp and S. S. Venkatesh, "Asymptotic expansions of the  $k$  nearest neighbor risk," *Ann. Statistics*, vol. 26, no. 3, pp. 850–878, 1998.

[23] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. San Diego, CA, USA: Academic, 1990.

[24] K. Das and Z. Nenadic, "Approximate information discriminant analysis: A computationally simple heteroscedastic feature extraction technique," *Pattern Recog.*, vol. 41, no. 5, pp. 1548–1557, 2008.

[25] Z. Nenadic, "Information discriminant analysis: Feature extraction with an information-theoretic objective," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 8, pp. 1394–1407, Aug. 2007.

[26] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. Hoboken, NJ, USA: Wiley-Interscience, 2000.

[27] M. Loog and R. Duin, "Linear dimensionality reduction via a heteroscedastic extension of LDA: The Chernoff criterion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 732–739, Jun. 2004.

[28] J. Kapur, *Measures of Information and Their Applications*. New York, NY, USA: Wiley, 1994.

- [29] R. Fletcher and M. Powell, "On the modification of  $LDL^T$  factorizations," *Math. Comput.*, vol. 28, no. 128, pp. 1067–1087, 1974.
- [30] J. R. Bunch, C. P. Nielsen, and D. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, pp. 31–48, 1978.
- [31] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Sci.*, vol. 290, no. 5500, pp. 2319–2323, 2000.
- [32] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits," [Online]. Available: <http://yann.lecun.com/exdb/mnist/>
- [33] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.
- [34] L. Fei-fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, Apr. 2006.
- [35] T. Tuytelaars, L. Van Gool, H. Bay, and A. Ess, "SURF: Speeded up robust features," *Comput. Vis. Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.
- [36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.



**Yung-Kyun Noh** received the BS degree in physics from POSTECH, in 1998 and the PhD degree in computer science from Seoul National University, in 2011. He is currently a BK assistant professor in the School of Mechanical and Aerospace Engineering, Seoul National University. His research interests include metric learning and dimensionality reduction in machine learning, and he is especially interested in applying statistical theory of nearest neighbors to real, large datasets. From 1998 to 2003, he was a member of

the Marketing Research Group, MPC Ltd., Korea, where he worked as a researcher and a system engineer. He worked in the GRASP Robotics Laboratory, University of Pennsylvania from 2007 to 2012 as a visiting researcher.



**Byoung-Tak Zhang** (M95) received the BS and MS degrees in computer science and engineering from Seoul National University (SNU), Seoul, Korea, in 1986 and 1988, respectively, and the PhD degree in computer science from the University of Bonn, Bonn, Germany, in 1992. He is the POSCO chair professor in the School of computer science and engineering and adjunct professor of the Graduate Programs in Cognitive Science and Brain Science, Seoul National University (SNU), and directs the Biointelligence

Laboratory and the Center for Cognitive Robotics and Artificial Intelligence (CRAIC). Prior to joining SNU, he was a research fellow in the German National Research Center for Information Technology (GMD), from 1992 to 1995. He was a visiting professor in the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), Cambridge, Massachusetts (2003-2004) and the Cognitive Interaction Technology Excellence Center (CITEC), Bielefeld, and the Cognition for Technical Systems (CoTeSys) Cluster of Excellence, Munich, Germany (2010-2011). His research focuses on understanding the computational principles of human brain and mind and on building autonomous intelligent robots that perceive, act, think, and learn like humans. He serves on the editorial board of *Genetic Programming and Evolvable Machines*, *Applied Intelligence*, and *Journal of Cognitive Science* and as an associate editor for *BioSystems*, *Advances in Natural Computation*, and the *IEEE Transactions on Evolutionary Computation* (1997-2010).



**Daniel D. Lee** received the BA (summa cum laude) degree in physics from Harvard University, in 1990 and the PhD in condensed matter physics from the Massachusetts Institute of Technology, in 1995. He is the UPS Foundation chair professor in the School of Engineering and Applied Science, University of Pennsylvania. Before coming to Penn, he was a researcher with AT&T and Lucent Bell Laboratories in the Theoretical Physics and Biological Computation departments. He has received the National Science Foundation

CAREER award and the University of Pennsylvania Lindback award for distinguished teaching. He was also a fellow of the Hebrew University Institute of Advanced Studies in Jerusalem, an affiliate of the Korea Advanced Institute of Science and Technology, and organized the US-Japan National Academy of Engineering Frontiers of Engineering symposium. As director of the GRASP Laboratory and co-director of the CMU-Penn University Transportation Center, his group focuses on understanding general computational principles in biological systems, and on applying that knowledge to build autonomous systems. He is a fellow of the IEEE.

► **For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).**