Joint Learning of Discriminative Prototypes and Large Margin Nearest Neighbor Classifiers

Martin Köstinger, Paul Wohlhart, Peter M. Roth, Horst Bischof
Institute for Computer Graphics and Vision, Graz University of Technology
{koestinger,wohlhart,pmroth,bischof}@icg.tugraz.at

Abstract

In this paper, we raise important issues concerning the evaluation complexity of existing Mahalanobis metric learning methods. The complexity scales linearly with the size of the dataset. This is especially cumbersome on large scale or for real-time applications with limited time budget. To alleviate this problem we propose to represent the dataset by a fixed number of discriminative prototypes. In particular, we introduce a new method that jointly chooses the positioning of prototypes and also optimizes the Mahalanobis distance metric with respect to these. We show that choosing the positioning of the prototypes and learning the metric in parallel leads to a drastically reduced evaluation effort while maintaining the discriminative essence of the original dataset. Moreover, for most problems our method performing k-nearest prototype (k-NP) classification on the condensed dataset leads to even better generalization compared to k-NN classification using all data. Results on a variety of challenging benchmarks demonstrate the power of our method. These include standard machine learning datasets as well as the challenging Public Figures Face Database. On the competitive machine learning benchmarks we are comparable to the state-of-the-art while being more efficient. On the face benchmark we clearly outperform the state-of-the-art in Mahalanobis metric learning with drastically reduced evaluation effort.

1. Introduction

Among the various different classification schemes k-nearest neighbor (k-NN) based approaches as Mahalanobis metric learning have recently attracted a lot of interest in computer vision. Several powerful metric learning frameworks (*e.g.* [28, 29], [8], or [11]) have been proposed that study different loss functions or regularizations. Conceptually, these methods take advantage of prior information in form of labels over simpler though more general similarity measures. Significant improvements have been

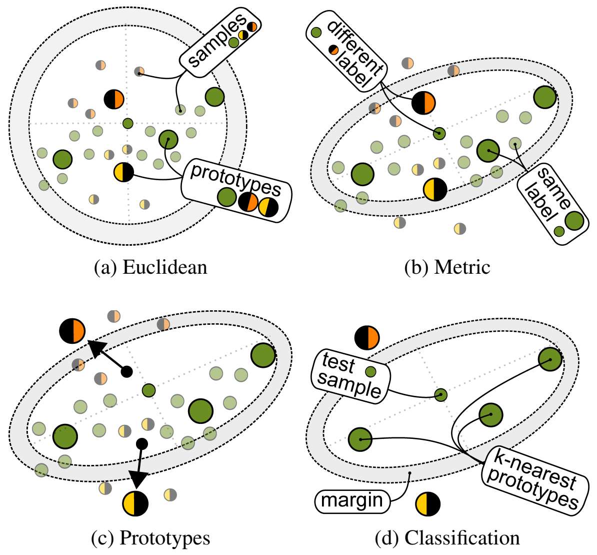


Figure 1: **Condensating a dataset by discriminative prototypes:** Learning the distance metric (b) and the positioning of the prototypes (c) in parallel allows to drastically reduce the evaluation effort while maintaining full discriminative power. With our method k-nearest prototype classification results improve even over k-NN classification for most problems (d).

observed for tracking [24], image retrieval [16], face identification [11], clustering [32], or person re-identification [14].

The large-scale nature of computer vision applications poses several challenges and opportunities to the class of Mahalanobis metric learning algorithms. For instance one can take the chance and learn a sophisticated distance metric that captures the structure of the dataset, or learn multiple local metrics that better adapt to the intrinsic characteristics of the feature space. On larger datasets this usually leads to lower error rates [29]. In contrast, this is challenged by the computational burden in training and the needed label effort. To reduce the required level of supervision, algorithms such as [8, 18] have been introduced that are able to learn from pairwise labels. Others tackle the problem of time complexity in learning by special optimization techniques [8, 29]. Ultimately, for many applications the time complexity in learning is not too critical. Nevertheless, one important aspect that is often neglected is the computational burden at test time.

One inherent drawback of Mahalanobis metric learning based methods is that the k-NN search in high-dimensional spaces is time-consuming, even on moderate sized datasets. For real-time applications with limited time budget this is even more critical. To alleviate this problem, different solutions have been proposed that focus on low dimensional embeddings. The resulting space should enable efficient retrieval and reflect the characteristics of the learned metric. For instance, one can accelerate nearest neighbor search by performing a low dimensional Hamming embedding. This can be done by applying locality sensitive hash functions directly [16] or on kernelized data [19]. Another strategy is to learn a low-rank Mahalanobis distance metric [29] that performs dimensionality reduction. Nevertheless, a too coarse approximation diminishes at least some of the benefits of learning a metric. Further, special data structures as metric ball trees have been introduced to speed up nearest neighbor search. Unfortunately, there is no significant time gain for high dimensional spaces.

Another technique is to reduce the number of training samples and introduce sparsity in the samples. Ideally, one maintains only a relatively small set of representative prototypes which capture the discriminative essence of the dataset. This condensation can be either seen as drawback, as it's likely to loose classification power, or taken as opportunity. In fact, the theoretical findings of Crammer *et al.* [6] provide even evidence that prototype-based methods can be more accurate than nearest neighbor classification. One reason might be that the condensation reduces overfitting. Choosing the positioning of the prototypes wisely can lead to a drastically reduced effort while maintaining the discriminative power of the original dataset.

Addressing challenges and opportunities of larger data sets and applications with limited time budget, this paper proposes to bridge the gap between Mahalanobis metric learning and discriminative prototype learning as illustrated in Figure 1. In particular, we are interested in joint optimization of the distance metric with respect to the discriminative prototypes and also of the positioning of the prototypes. This combination enables us to drastically reduce the computational effort while maintaining accuracy. Furthermore, we provide evidence that in most cases the proposed Discriminative Metric and Prototype Learning (DMPL) method generalizes even better to unseen data compared to recent Mahalanobis metric k-NN classifiers.

The rest of this paper is structured as follows: In Section 2 we give a brief overview of related work in the field of Mahalanobis metric and prototype learning. Succeeding, in Section 3 we describe our Discriminative Metric and Prototype Learning (DMPL) method as an alternating optimization problem. Detailed experiments on standard machine learning datasets and on the challenging PubFig [20] face recognition benchmark are provided in Section 4.

2. Related Work

Compared to other classification models Mahalanobis Metric Learning provides with k-NN search not only reasonable results but is also inherently multi-class and directly interpretable, based on the assigned neighbors. Several different methods (e.g., [28], [8], or [11]) have been proposed that show good results for many real world problems. A particular successful instance of this class of algorithms is the approach of Weinberger *et al.* [28, 29], which aims at improving k-NN classification by exploiting the local structure of the data. It mimics the non-continuous and non-differentiable classification error of the k-NN scheme by a convex loss function. The main idea bases on two simple intuitions. First, the k-NNs of each sample that share the class label (target neighbors) should move close to each other. Second, no differently labeled sample should invade this local k-NN perimeter plus a safety margin. This safety margin allows for focusing on samples near the local k-NN decision boundary and ensures that the model is robust to small amounts of noise.

Prototype methods such as learning vector quantization (LVQ) share some of the favorable characteristics of Mahalanobis metric learning. They deliver intuitive, interpretable classifiers based on the representation of classes by prototypes. The seminal work of Kohonen [17] updates prototypes iteratively based on a clever heuristic. A data point attracts the closest prototype in its direction if it matches the class label. Vice-versa it is repelled if it shows a different class label. Various extensions have been proposed that modify the original update heuristic. For instance, updating both the closest matching and non-matching prototype or restricting the updates close to the decision boundary. Otherwise LVQ can show divergent behavior.

Seo and Obermayer [23] explicitly avoid the divergent behavior by an underlying optimization problem. The main idea is to treat the prototypes as unit size, isotropic Gaussians and maximize the likelihood ratio of the probability of correct assignment versus the total probability in the Gaussian mixture model. The resulting robust learning scheme updates only prototypes close to the decision boundary by incorrectly classified samples. Also, the work of Crammer *et al.* [6] derives a loss-based algorithm for prototype positioning based on the maximal margin principle. LVQ arises as special case of this algorithm. Remarkably, the authors provide evidence that prototype methods follow max-margin principles. However, for this class of algorithms classification is solely based on a predefined metric.

Therefore, to alleviate this issue variants have been proposed that learn some parameters of the distance function. For instance, Parametric Nearest Neighbor (P-NN) and its ensemble extension (EP-NN) [36] learn weights on the Euclidean distance function. Bonilla and Robles-Kelly [1] propose a probabilistic discriminative generalization of vector quantization. They jointly learn discriminative weights on soft-assignments to prototypes and further the prototype positions. Nevertheless, as these approaches learn only restricted parameters of the distance function these may miss different scalings or correlations of the features.

In contrast to these previous works we want to exploit a more general metric structure. In particular, we are interested in improving runtime and classification power by combining the favorable characteristics of Mahalanobis metric learning and prototype methods. Our method integrates a large margin formulation with focus on samples close to the decision boundary. Further, it naturally integrates with k-NN, which may be in some situations the favorable choice over nearest neighbor assignment.

3. Discriminative Mahalanobis Metric and Prototype Learning

In the following, we derive a new formulation that jointly chooses the positioning of prototypes and also optimizes the distance metric with respect to these. This allows us to exploit the global structure of the data (via metric) and to drastically reduce the computational effort during evaluation (via prototypes). Finally, this reduces evaluation time and improves k-NP classification.

3.1. Problem Formulation

For the following discussion let us introduce a training set $\mathcal{X} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, with N samples $\mathbf{x}_i \in \mathbb{R}^D$ and corresponding labels $y_i \in \{1, 2, \ldots, C\}$. Let $\mathcal{Z} = \{(\mathbf{z}_1, y_1), \ldots, (\mathbf{z}_K, y_K)\}$ correspond to a set of K prototypes. Then, the squared Mahalanobis distance between a data sample $\mathbf{x}_i$ and a prototype $\mathbf{z}_k$ is defined as

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{z}_k) = (\mathbf{x}_i - \mathbf{z}_k)^\top \mathbf{M} (\mathbf{x}_i - \mathbf{z}_k), \tag{1}$$

where $\mathbf{M} \succeq 0$ is a symmetric positive semidefinite matrix. Our goal is to estimate the metric matrix $\mathbf{M}$ and the prototypes $\{\mathbf{z}_k\}_1^K$ in parallel. The idea to fuse metric and prototype learning is general and can be adapted to various Mahalanobis metric learning methods. In particular, we adopt ideas of LMNN [28] and locally establish a perimeter surrounding each data sample. Prototypes with different class label should not invade this perimeter plus a safety margin. This behavior can be realized by minimizing the following energy:

$$\begin{aligned}\epsilon^i(\mathbf{M}, \{\mathbf{z}_k\}_{k=1}^K) = (1-\mu) \sum_{j \rightsquigarrow i} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{z}_j) \\ + \mu \sum_{j \rightsquigarrow i} \sum_{l} (1 - y_{il}) \xi_{ijl}(\mathbf{M}),\end{aligned} \tag{2}$$

where $j \rightsquigarrow i$ indicates that $\mathbf{z}_j$ is a target prototype of sample $\mathbf{x}_i$ and $\mu \in [0, 1]$ is a weighting factor. The first term attracts target prototypes $\mathbf{z}_j$ while the second term emits a repelling force on differently labeled prototypes $\mathbf{z}_l$ that invade the perimeter. We refer to these invaders as impostor prototypes. Note that the pairwise label y_{il} is zero if $y_i \neq y_l$ and one otherwise.

If a prototype invades the local perimeter plus margin is monitored by

$$\xi_{ijl}(\mathbf{M}) = \left[1 + d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{z}_j) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{z}_l)\right]_+, \tag{3}$$

where $[a]_+ = \max(a, 0)$ is the hinge loss. It activates only if the prototype is closer to the sample $\mathbf{x}_i$ than the target prototype $\mathbf{z}_j$ plus margin. Finally, the overall energy function is a sum of the local contributions:

$$\epsilon(\mathbf{M}, \{\mathbf{z}_k\}_{k=1}^K) = \sum_{i=1}^N \epsilon^i(\mathbf{M}, \{\mathbf{z}_k\}_{k=1}^K). \tag{4}$$

In order to minimize the energy function we use an alternating optimization based on gradient descent w.r.t. the prototype positions $\{\mathbf{z}_k\}_{k=1}^K$ and the distance metric $\mathbf{M}$. At each iteration we take a sufficiently small gradient step and monitor boundary conditions as $\mathbf{M} \succeq 0$. In the following, we derive alternating update rules in terms of prototypes and the metric matrix.

3.2. Learning Prototypes

First, we derive the update rules w.r.t. to the prototype locations. As the particular role of an individual prototype as target or impostor is ambiguous on global scope we express the gradient

$$\frac{\partial \epsilon(\mathbf{M}, \{\mathbf{z}_k\}_{k=1}^K)}{\partial \mathbf{z}_k} = \sum_{i=1}^N \frac{\partial \epsilon^i(\mathbf{M}, \{\mathbf{z}_k\}_{k=1}^K)}{\partial \mathbf{z}_k} \tag{5}$$

as sum over the (unambiguous) gradient contribution of each data sample $\mathbf{x}_i$ on the respective prototype $\mathbf{z}_k$. A prototype can be a target neighbor ($k = j$), an impostor ($k = l$), or simply irrelevant as too far away. Therefore, we specify the gradient contribution of a sample on a prototype as follows:

$$\frac{\partial\epsilon^i(\mathbf{M},\{\mathbf{z}_k\}_{k=1}^K)}{\partial\mathbf{z}_k} = \begin{cases} \frac{\partial\epsilon^i(\mathbf{M},\{\mathbf{z}_k\}_{k=1}^K)}{\partial\mathbf{z}_j} & \text{if } k{=}j \\ \frac{\partial\epsilon^i(\mathbf{M},\{\mathbf{z}_k\}_{k=1}^K)}{\partial\mathbf{z}_l} & \text{if } k{=}l \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{6}$$

Taking into account that

$$\frac{\partial d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{z}_k)}{\partial\mathbf{z}_k} = -2(\mathbf{x}_i-\mathbf{z}_k)^\top\mathbf{M} \tag{7}$$

we can re-write the gradients defined in Eq. (6). Substituting Eq. (7) into Eq. (6) for a target prototype ($k = j$, attraction force) we get

$$\begin{aligned}\frac{\partial\epsilon^i(\mathbf{M},\{\mathbf{z}_k\}_{k=1}^K)}{\partial\mathbf{z}_j} &= (1-\mu)\sum_{j\rightsquigarrow i} -2(\mathbf{x}_i-\mathbf{z}_j)^\top\mathbf{M} \\ &+ \mu \sum_{l \text{ s.t. } (i,j,l)\in\mathcal{I}} -2(\mathbf{x}_i-\mathbf{z}_j)^\top\mathbf{M},\end{aligned} \tag{8}$$

where $\mathcal{I} = \{(i,j,l)|\xi_{ijl} > 0\}$ is the set of active sample-impostor triplets. Similarly, we get

$$\frac{\partial\epsilon^i(\mathbf{M},\{\mathbf{z}_k\}_{k=1}^K)}{\partial\mathbf{z}_l} = -\,\mu \sum_{l \text{ s.t. } (i,j,l)\in\mathcal{I}} -2(\mathbf{x}_i-\mathbf{z}_l)^\top\mathbf{M} \tag{9}$$

for an impostor prototype ($k = l$, repelling force). Finally, we can specify the iterative update rule at iteration t for the prototypes as

$$\mathbf{z}_k^{(t+1)} = \mathbf{z}_k^{(t)} - \eta\,\frac{\partial\epsilon\left(\mathbf{M}^{(t)}, \left\{\mathbf{z}_k^{(t)}\right\}_{k=1}^K\right)}{\partial\mathbf{z}_k^{(t)}}, \tag{10}$$

where η denotes the learning rate. Reasonable choices for the initial prototypes are all variants of clustering algorithms such as k-means or using training samples as initialization. We emphasize that compared to the update rules of LVQ or P-NN [36] our formulation is more general and natively integrates in k-NP classification. Further, it accounts for different scalings and correlations of the feature space.

3.3. Distance Metric Learning

Next, we derive the update rule w.r.t. the distance metric in terms of the local contribution of each sample to its neighboring prototypes. Hence, the derivative can be expressed as

$$\frac{\partial\epsilon(\mathbf{M},\{\mathbf{z}_k\}_{k=1}^K)}{\partial\mathbf{M}} = \sum_{i=1}^N \frac{\partial\epsilon^i(\mathbf{M},\{\mathbf{z}_k\}_{k=1}^K)}{\partial\mathbf{M}}. \tag{11}$$

To estimate $\mathbf{M}$, gradient descent is performed along the gradient defined by the set of active sample-impostor triplets $\mathcal{I}$. We can write the gradient as

$$\begin{aligned}\frac{\partial\epsilon^i\left(\mathbf{M},\{\mathbf{z}_k\}_{k=1}^K\right)}{\partial\mathbf{M}} &= (1-\mu)\sum_{j\rightsquigarrow i}\mathbf{C}_{ij} \\ &+ \mu \sum_{(j,l) \text{ s.t. } (i,j,l)\in\mathcal{I}} (\mathbf{C}_{ij}-\mathbf{C}_{il})\ ,\end{aligned} \tag{12}$$

where $\mathbf{C}_{ik}$ denotes the outer product of pairwise differences. This is the gradient of the distance function $d^2_{\mathbf{M}}$:

$$\mathbf{C}_{ik} = (\mathbf{x}_i-\mathbf{z}_k)(\mathbf{x}_i-\mathbf{z}_k)^\top = \frac{\partial d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{z}_k)}{\partial\mathbf{M}}. \tag{13}$$

Eq. (12) conceptually tries to strengthen the correlation between the sample and target prototypes while weakening it between the sample and impostor prototypes. Finally, we can specify the iterative update rule at iteration t as

$$\mathbf{M}^{(t+1)} = \mathbf{M}^{(t)} - \eta\,\frac{\partial\epsilon\left(\mathbf{M}^{(t)}, \left\{\mathbf{z}_k^{(t)}\right\}_{k=1}^K\right)}{\partial\mathbf{M}^{(t)}}. \tag{14}$$

Initially, we start with the Euclidean distance ($\mathbf{M} = \mathbf{I}$). Note that after each iteration we check if $\mathbf{M}$ induces a valid pseudo-metric. To satisfy metric conditions we use a projection operator similar to [13] by back-projecting the current solution on the cone of positive semidefinite (p.s.d.) matrices.

3.4. Evaluation Complexity

In the following, we want to to assess the efficiency of our proposed method at test time. For that purpose we derive the computational complexity and compare it to related algorithms. Since $\mathbf{M}$ is p.s.d. we can decompose it as $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$. This allows to express the squared distance in Eq. (1) by computing the Euclidean distance after performing a linear transformation as

$$d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{z}_k) = (\mathbf{x}_i-\mathbf{z}_k)^\top\underbrace{\mathbf{L}^\top\mathbf{L}}_{\mathbf{M}}(\mathbf{x}_i-\mathbf{z}_k) \ = \tag{15}$$

$$= (\mathbf{L}\mathbf{x}_i-\mathbf{L}\mathbf{z}_k)^\top(\mathbf{L}\mathbf{x}_i-\mathbf{L}\mathbf{z}_k) \ = \tag{16}$$

$$= ||\mathbf{L}(\mathbf{x}_i-\mathbf{z}_j)||^2. \tag{17}$$

One advantage of this formulation is that for all prototypes $\{\mathbf{z}_k\}_{k=1}^{K}$ the linear transformation $\mathbf{L}\mathbf{z}_k$ can be precomputed prior to testing. Further, the test sample $\mathbf{L}\mathbf{x}_i$ has to be projected only once at test time. The complexity of the matrix-vector multiplication is $\mathcal{O}(D^2)$, where D is the feature dimensionality. Then, the complexity of comparing one test sample with one prototype is $\mathcal{O}(D^2 + D)$. Straight forward for the k-NP search we arrive at $\mathcal{O}(D^2 + DKC)$, where K is the number of prototypes for each of the C classes. In typical scenarios this reduces to $\mathcal{O}(DKC)$ as the matrix-vector multiplication is negligible due to $D^2 \ll DKC$. Thus, the k-NP search is linear in terms of the overall number of prototypes and the feature dimensionality.

The comparison to related methods reveals some interesting results. If we assume the same number of anchor points a local-linear SVM has the same complexity $\mathcal{O}(DCA)$, where A is the number of anchor points. For a kernel SVM and the kernelized hashing approach of [19] the evaluation scales linearly with the number of support vectors S or kernel samples times the kernel complexity K_c, $\mathcal{O}(SK_c)$. The locality-sensitive hashing approach of [16] scales with $\mathcal{O}(MD)$. M is the length of the short list of samples generated by approximate search in Hamming space [5]. This list is considered for exact search under the learned metric. k-NN based approaches scale linearly with the number of training samples N, $\mathcal{O}(DN)$. Note that in typical scenarios the overall number of prototypes is considerable smaller than the number of training samples: $KC \ll N$.

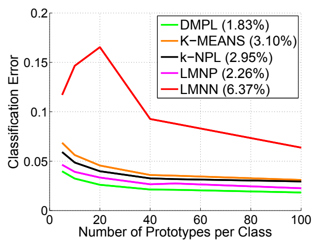
Thus, if we recapitulate the characteristics of the different approaches our method scales linearly with the number of prototypes. This allows for classification with a fixed time budget which is beneficial for time critical applications. Succeeding, we show that we outperform methods that have a lower computational complexity. Further, we are comparable to, or even improve over methods with a similar or higher complexity. Only drastically more complex methods show better results.

4. Experiments

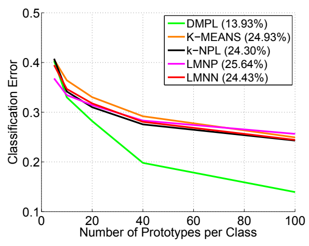
To show the broad applicability of our method we conduct experiments on various standard benchmarks with rather diverse characteristics. The goals of our experiments are twofold. First, we want to show that with a drastically reduced prototype set we get comparable or even better results than related work. Second, we want to prove that we are more efficient in evaluation. This is clearly beneficial for large scale or real-time applications.

4.1. Machine Learning Databases

In the following, we benchmark our proposed method on MNIST [10], USPS [15], LETTER [10] and CHARS74k [3]. First, we give a brief overview of the databases. Second, we compare our method Discriminative Metric and Prototype Learning (DMPL) to several baselines such as learning only the prototypes or the distance metric. Finally, we compare the performance related to the evaluation complexity between our method and state-of-the-art approaches.



(a) MNIST [10]



(b) CHARS74k [3]

Figure 2: **Benchmark of our proposed method (DMPL) to baseline approaches**: K-Nearest Prototype Learning (kNPL, Eq. (10)), Large Margin Nearest Prototype (LMNP, Eq. (14)) learning, k-means and plain LMNN [29]. We compare the 1-NP classification error in relation to the number of prototypes per class. The numbers in parenthesis denote the classification error with 100 prototypes per class.

The MNIST database [10] of hand written digits contains in total 70,000 images in one train-test split. 60,000 samples are used for training and 10,000 for testing. The images have a resolution of 28×28 pixels and are in grayscale. Similarly USPS [15] contain grayscale images of hand written digits with a resolution of 16×16 pixels. 7291 images are organized for training and 2007 images for testing.

In contrast, the LETTER [10] database contains a large number of synthesized images showing one of the 26 capital letters of the English alphabet. The images are represented as 16 dimensional feature vector which describes statistical moments and edge counts.

Chars74K [3] contains a large mixed set of natural and synthesized characters. The images comprise either one of the 26 capital or lowercase letters and digits. Thus, the dataset features 62 classes. 7,705 characters are cropped of natural images, 3,410 are hand drawn and 62,992 are synthesized. Similar to [36] we apply a color space conversion to grayscale and resizes each image to 8×8 pixels. Further, the database is split into one train-test set where 7400 samples are organized for testing and the rest for training.

For MNIST we perform a dimensionality reduction of the raw features by PCA to a 164 dimensional subspace, to make the learning more tractable. For all other databases we use the raw data without calculating any complex features, in order to get a fair comparison.

In Figure 2 we compare our method (DMPL) to baseline approaches on the respective benchmarks. Therefore, we plot the classification error in relation to the number of prototypes. In particular, we report the following results:

Methods		MNIST	USPS	LETTER	Chars74K
Prototype Methods	DMPL $_{\text{1-NP}}$ (40 prototypes)	2.13	5.68	3.13	19.81
	DMPL $_{\text{1-NP}}$ (100 prototypes)	1.83	4.93	2.48	13.93
	DMPL $_{\text{3-NP}}$ (200 prototypes)	1.66	4.83	2.50	14.05
	Parametric NN (40 prototypes) [36]	3.13	7.87	6.95	29.46
	Ensemble of P-NN (800 prototypes) [36]	1.65	4.88	2.90	19.53
Nearest Neighbors	Nearest Neighbor $_{\text{1-NN,3-NN}}$	2.92 - 3.09	4.88 - 5.08	4.30 - 4.35	17.97 - 19.99
	LMNN $_{\text{1-NN,3-NN}}$ [28, 29]	1.70 - 2.09	4.73 - 4.78	2.93 - 3.54	17.07 - 19.08
SVMs	Linear [9]	8.18	8.32	23.63	35.08
	Kernel [4, 7, 25, 2]	1.36 - 1.41	4.24 - 4.58	2.12 - 2.80	16.86
Locally linear classifiers	Lin. SVM + LCC (4,096 anchor p.) [34, 33, 27]	1.64 - 2.28	4.38	4.12	20.88
	Lin. SVM + DCN (L1 = 64,L2 = 512) [22]	1.51	-	-	-
	Local Linear SVM (100 anchor p.) [21]	1.85	5.78	5.32	25.11
	LIB-LLSVM + OCC [35]	1.61	3.94	6.85	18.72
	ALH [31]	2.15	4.19	2.95	16.26
Locality sensitive Hashing	Spectral hashing [30, 12]	4.25 - 5.27	8.72 - 13.35	7.42 - 33.67	26.03
	Fast Image Search for Learned Metrics (ϵ = 0.6) [16]	5.51	5.53	8.55	-
	KLSH [19] (10,000 kernel samples)	6.15	5.68	7.38	88.76
	Spherical Hashing [12]	2.22	5.13	19.00	16.65

Table 1: **Comparison of classification error rates on MNIST, USPS, LETTER and Chars74k**. Our method (denoted DMPL) outperforms several state-of-the-art approaches while being more efficient. With 200 prototypes we improve even over LMNN which requires the full dataset for classification. The top performing method of each category is highlighted. K-NP refers to the number of prototypes used for classification.

	DMPL vs ...	**LMNN** [28]	**SVM** [9]	**LLC** [27, 34, 33]	**LL-SVM** [21]	**P-NN** [36]	**EP-NN** [36]	**FSLM** [16]
100	Rel. Compl.	60	$\frac{1}{100}$	40	1	$\frac{1}{2.5}$	8	1.94
	Rel. Error	+ 0.13%	- 6.35%	- 0.45% to + 0.19%	- 0.02%	- 1.30%	+ 0.18%	- 3.68%
200	Rel. Compl.	30	$\frac{1}{200}$	20	$\frac{1}{2}$	$\frac{1}{5}$	4	0.97
	Rel. Error	- 0.05%	- 6.52%	- 0.62% to + 0.02%	- 0.19%	- 1.47%	+ 0.01%	- 3.86%

Table 2: **Relative comparison of the evaluation complexity and the difference of classification errors** using MNIST [10]. We compare DMPL 1-NP with 100 and DMPL 3-NP with 200 prototypes vs related state-of-the-art. For instance, compared to LMNN DMPL 3-NP with 200 prototypes is 30 times faster and has a 0.05 percentage points lower classification error.

The direct assignment of the k-means cluster label, thus ignoring discriminative information in learning at all. Second, we compare to training standard LMNN on the prototypes. Here, the main goal is to stress the difference between optimizing for k-NP classification or k-NN classification. Third, we compare to only tuning the positioning of the prototypes, referred as k-Nearest Prototype Learning (kNPL). Finally, we optimize only the distance metric assuming fixed prototypes, referred as Large Margin Nearest Prototype (LMNP) learning.

For the following discussion we focus on the respective results on MNIST visualized in Figure 2 (a), although the relative results are comparable on the different datasets. As expected LMNN and k-means perform initially worse than the prototype based methods. In case of LMNN the performance gap is rather big. By increasing the number of prototypes the gap gets smaller as the k-means centroids behave more similar to the actual data samples. However, ultimately for MNIST a performance gap of about 4.5% remains. Comparing LMNN to LMNP reveals that it is beneficial to optimize the distance metric in respect to the prototypes. The drop in terms of classification error is about 4% with 100 prototypes. Interestingly, k-means is more competitive compared to LMNN right from the beginning. Nevertheless, it is outperformed by both, kNPL and also LMNP. Comparing the baselines to our discriminative metric and prototype learning (DMPL) method reveals that the power lies in the combination of distance metric learning and prototype methods. DMPL outperforms LMNN by roughly 4.5% and k-means by 1.3%. As MNIST is a rather competitive dataset this is a reasonable performance gain.

Next, in Table 1 we benchmark our method to various state-of-the-art approaches. These include recent local linear methods, support vector machines, nearest neighbor and prototype based methods. Further, Table 2 gives a relative comparison of the evaluation complexity of selected methods and their classification error on MNIST. The performance comparison between local linear methods as

LL-SVM [21] and prototype methods is especially interesting as a nearest prototype classifier is essentially a local linear classifier. The decision boundaries are perpendicular to the connection lines between the prototypes.

The first important finding is that DMPL outperforms methods that either use a predefined or learned metric even though being more efficient. Compared to vanilla k-NN search with plain Euclidean distance the main advantage is the ability to model different scalings and correlations of the feature space. Further, using less prototypes DMPL improves over a recent prototype based method [36] that learns only a relevance weighting of the features. One advantage is that DMPL is able to account for different correlations of the feature space. Compared to LMNN the flexibility remains to discriminatively adapt the positioning of the prototypes.

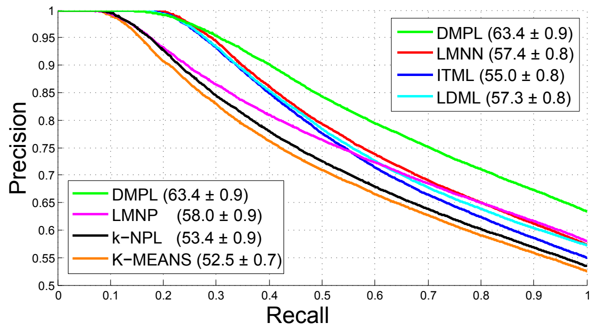
Second, like DMPL locality sensitive hashing based approaches focus on efficient retrieval. Nevertheless compared to our method they trade off classification power for efficiency. The results show that kernelized hashing needs a large number of kernel samples to obtain comparable results. Standard LSH approaches need to consider a large number of samples for exact search, diminishing at least some of the speed advantages.

Finally, compared to kernel SVMs our method is outperformed only slightly while being able to perform classification with a fixed time budget. For kernel SVMs it is known that the number of support vectors scale linearly with the size of the dataset. Local linear methods such as LL-SVM [21] bypass this issue. Interestingly, they share our computational complexity. On MNIST LL-SVM matches our performance, however on USPS and LETTER we are able to improve over LL-SVM. Only local linear methods using a much larger number of anchor points are able to improve over our method.

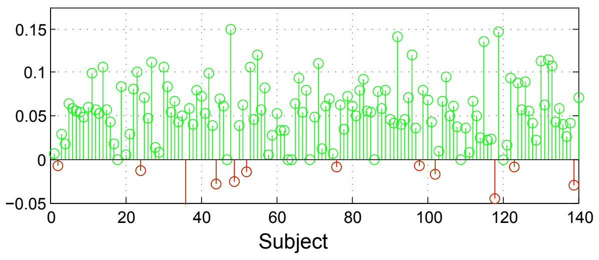
Recapitulating the different results and relating them to the evaluation complexity of related works it reveals that we get competitive results and are more efficient.

4.2. Public Figures Face Database

In the following, we demonstrate our method for face identification on the Public Figures Face Database (PubFig) [20]. PubFig is a large, real-world face dataset consisting of 58,797 images of 200 people. The evaluation set contains 42,461 images of 140 individuals. PubFig is considered as very challenging as it exhibits huge variations in pose, lighting, facial expression and general imaging and environmental conditions. To represent the faces we use the description of visual face traits [20]. They describe the presence or absence of 73 visual attributes, such as gender, race, hair color *etc*. Further, we apply a homogeneous χ^2 feature mapping [26]. For the face identification benchmark we organize the data similar to the existing verification protocol in 10 folds for cross-validation. Therefore, we split the images of each individual into 10 disjoint sets.



(a)



(b)

Figure 3: **Face identification benchmark on the PubFig database** [20]: The data is organized in 10 non-overlapping folds for cross-validation. (a) Precision / Recall curves by ranking and thresholding classifier scores. Numbers in parenthesis denote the precision and std. dev. at full recall. (b) Difference of precision per person between DMPL and LMNN.

In Figure 3 (a) we benchmark our method using 100 prototypes per class to recent Mahalanobis metric learning methods. We report the face identification performance in a refusal to predict style. In that sense, recall means the percentage of samples which have a higher classifier score than the current threshold. Precision means the ratio of correctly labeled samples.

In particular, we show that DMPL generalizes better than LMNN [28], ITML [8] or LDML [11] which require on the full training set for classification. At full recall the performance difference to LMNN is 6.00%. Further, comparing the results to the baseline approaches kNPL and LMNP reveals once more that the power lies in the combination of metric learning and prototype learning. In Figure 3 (b) we compare the relative change in classification accuracy per person between our method and LMNN. Only for a small number of classes the performance drops slightly while for the vast number the performance increases. Thus, there is no bias in terms of overrepresented classes. Intuitively, one interpretation is that the fixed number of prototypes helps to compensate for overfitting.

5. Conclusion

In this work we presented a novel method to condense a dataset to a fixed number of discriminative prototypes. In particular, we jointly choose the positioning of prototypes and also optimize the Mahalanobis distance metric with respect to these. This leads to a drastically reduced effort at test time while maintaining the discriminative essence of the original dataset. Our method performing k-nearest prototype classification on the condensed dataset leads to even better generalization compared to k-NN classification. To show the merit of our method we conducted several experiments on various challenging large-scale benchmarks.

Acknowledgments The work was supported by the Austrian Research Promotion Agency (FFG) projects FACTS (832045) and DIANGO (840824).

References

[1] E. Bonilla and A. Robles-Kelly. Discriminative probabilistic prototype learning. In *Proc. ICML*, 2012. 3

[2] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In *Proc. ICML*, 2007. 6

[3] T. E. d. Campo, B. R. Babu, and M. Varma. Character Recognition in Natural Images. In *Proc. VISAPP*, 2009. 5

[4] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *Trans. on Intelligent Systems and Technology*, 2011. 6

[5] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. ACM Symposium on Theory of Computing*, 2002. 5

[6] K. Crammer, R. Gilad-bachrach, A. Navot, and N. Tishby. Margin analysis of the LVQ algorithm. In *Advances NIPS*, 2002. 2

[7] K. Crammer, Y. Singer, N. Cristianini, J. Shawe-taylor, and B. Williamson. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2001. 6

[8] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proc. ICML*, 2007. 1, 2, 7

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 2008. 6

[10] A. Frank and A. Asuncion. UCI machine learning repository, 2010. University of California, Irvine, School of Information and Computer Sciences. Available online at http://archive.ics.uci.edu/ml. 5, 6

[11] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *Proc. ICCV*, 2009. 1, 2, 7

[12] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *Proc. CVPR*, 2012. 6

[13] N. J. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118, 1988. 4

[14] M. Hirzer, P. M. Roth, M. Köstinger, and H. Bischof. Relaxed pairwise learned metric for person re-identification. In *Proc. ECCV*, 2012. 1

[15] J. J. Hull. A database for handwritten text recognition research. *Trans. PAMI*, 1994. 5

[16] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *Proc. CVPR*, 2008. 1, 2, 5, 6

[17] T. Kohonen. *Self-organization and associative memory*. Springer-Verlag New York, Inc., 1989. 2

[18] M. Köstinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *Proc. CVPR*, 2012. 2

[19] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *PAMI*, 2012. 2, 5, 6

[20] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and Simile Classifiers for Face Verification. In *Proc. ICCV*, 2009. 2, 7

[21] L. Ladicky and P. H. S. Torr. Locally linear support vector machines. In *Proc. ICML*, 2011. 6, 7

[22] Y. Lin, T. Zhang, S. Zhu, and K. Yu. Deep coding networks. In *Advances NIPS*, 2010. 6

[23] S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 2002. 2

[24] C. Shen. Non-sparse linear representations for visual tracking with online reservoir metric learning. In *Proc. CVPR*, 2012. 1

[25] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 2005. 6

[26] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intellingence*, 34(3), 2011. 7

[27] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proc. CVPR*, 2010. 6

[28] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances NIPS*, 2006. 1, 2, 3, 6, 7

[29] K. Q. Weinberger and L. K. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proc. ICML*, 2008. 1, 2, 5, 6

[30] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances NIPS*, 2008. 6

[31] T. Yang and V. Kecman. Adaptive local hyperplane classification. *Neurocomputing*, 2008. 6

[32] J. Ye, Z. Zhao, and H. Liu. Adaptive distance metric learning for clustering. In *Proc. CVPR*, 2007. 1

[33] K. Yu and T. Zhang. Improved local coordinate coding using local tangents. In *Proc. ICML*, 2010. 6

[34] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *Advances NIPS*, 2009. 6

[35] Z. Zhang, L. Ladicky, P. Torr, and A. Saffari. Learning anchor planes for classification. In *Advances NIPS*, 2011. 6

[36] Z. Zhang, P. Sturgess, S. Sengupta, N. Crook, and P. H. S. Torr. Efficient discriminative learning of parametric nearest neighbor classifiers. In *Proc. CVPR*, 2012. 3, 4, 5, 6, 7