Large Scale Metric Learning for Distance-Based Image Classification

Thomas Mensink, Jakob Verbeek, Florent Perronnin, Gabriela Csurka

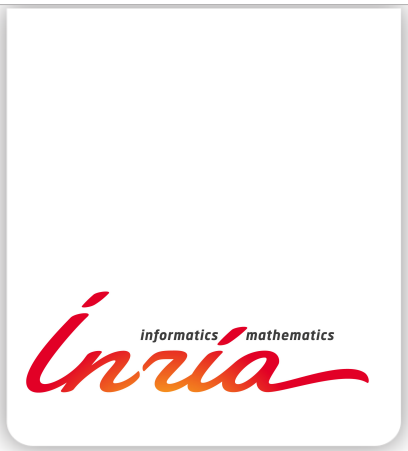
Large Scale Metric Learning for Distance-Based Image Classification

Thomas Mensink, Jakob Verbeek,
Florent Perronnin, Gabriela Csurka

RESEARCH REPORT
N° 8077
September 2012

Project-Teams LEAR - INRIA
and TVPA - XRCE

ISSN 0249-6399 ISRN INRIA/RR--8077--FR+ENG



Large Scale Metric Learning for Distance-Based Image Classification

Thomas Mensink[*†], Jakob Verbeek[*],
Florent Perronnin[†], Gabriela Csurka[†]

Project-Teams LEAR - INRIA and TVPA - XRCE

Research Report n° 8077 — September 2012 — 27 pages

Abstract: This paper studies large-scale image classification, in a setting where new classes and training images could continuously be added at (near) zero cost. We cast this problem into one of learning a low-rank metric, which is shared across all classes and explore k-nearest neighbor (k-NN) and nearest class mean (NCM) classifiers. We also introduce an extension of the NCM classifier to allow for richer image representations.
Experiments on the ImageNet 2010 challenge dataset —which contains more than 1M training images of 1K classes— shows, surprisingly, that the NCM classifier compares favorably to the more flexible k-NN classifier. Moreover, the NCM performance suggests that 256 dimensional features is comparable to that of linear SVMs, which were used to obtain the current state-of-the-art performance.
Experimentally we study the generalization performance to classes that were not used to learn the metrics, and show how a zero-shot model based on the ImageNet hierarchy can be combined effectively with small training datasets. Using a metric learned on 1K classes of the ImageNet 2010 Challange, we show results for the ImageNet-10K dataset, and obtain performance that is competitive with the current state-of-the-art, while requiring significant less training time.

Key-words: Metric Learning, k-Nearest Neighbor Classification, Nearest Mean Classification, Large Scale Image Classification, Zero-Shot Learning

[*] LEAR group, INRIA Grenoble, first.lastname@inria.fr
[†] TVPA group, Xerox Research Centre Europe, firstname.lastname@xrce.xerox.com

RESEARCH CENTRE GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Apprentissage de Métriques à Grande Échelle pour la Classification Basée sur la Distance

Résumé : Cet article étudie la classification des images à une grande échelle, dans un contexte où des nouvelles classes et des nouvelles images pourraient être ajoutées en continu pour un coût quasi-nul. Nous avons formalisé ce problème comme l'apprentissage d'une métrique de faible rang partagée par par toutes les classes et nous avons exploré les classifieurs par k plus proches voisins (k-NN) et par plus proche moyenne de classe (NCM). Nous introduisons également une extension du classificateur NCM pour permettre les représentations d'images plus riches.

Nous avons fait des expériences sur des données de ImageNet défi 2010 — qui contient plus de 1M images d'apprentissage des 1K classes — montrant, étonnamment, que le classificateur NCM se compare favorablement au plus flexible classificateur k-NN. En outre, la performance de NCM suggère qu'un vecteur de 256 dimensions donne des résultats comparables à ceux des SVMS linéaires, qui constituent l'état de l'art actuel.

Expérimentalement, nous étudions les performances de généralisation des classes qui n'ont pas été utilisées pour apprendre les paramètres. Nous montrons comment un modèle sans exemple reposant sur la hiérarchie ImageNet peut être combiné efficacement avec des ensembles constitués de peu d'exemples d'apprentissage. En utilisant une métrique appris sur les 1K classes de Imagenet défi 2010, nous obtenons des résultats pour l'ensemble de données ImageNet-10K, qui rivalisent avec l'état de l'art, tout en permettant une phase d'apprentissage plus rapide.

Mots-clés : Apprentissage de métriques, Classifieurs par k plus proches voisins, classifieurs par plus proche moyenne, Classification à grande échelle, Apprentissage sans exemple

1 Introduction

In the last decade we have witnessed an explosion in the amount of images and videos that are digitally available, *e.g*. in broadcasting archives or social media sharing websites. Scalable automated methods are needed to handle this huge volume of data, for retrieval, annotation and visualization purposes. This need has been recognized in the computer vision research community and large-scale methods have become an active topic of research in recent years. The introduction of the ImageNet dataset (Deng et al., 2009), which contains more than 14M manually labeled images of 22K classes, has provided an important benchmark for large-scale image classification and annotation algorithms.

In this paper we focus on the problem of large-scale image annotation, where the goal is to assign automatically a set of relevant labels from a given vocabulary to an image, *e.g*. names of objects appearing in the image, or a general label like the scene type of the image. The predominant approach to this problem is to treat it as a classification problem. To ensure scalability, often linear classifiers such as SVMs are used (Sánchez and Perronnin, 2011; Lin et al., 2011). Additionally, to speed up classification, dimension reduction techniques could be used (Weston et al., 2011), or a hierarchy of classifiers could be learned (Bengio et al., 2011; Gao and Koller, 2011). Recently, impressive results have been reported on 10,000 or more classes (Deng et al., 2010; Weston et al., 2011; Sánchez and Perronnin, 2011). A drawback of these methods, however, is that when images of new categories become available, new classifiers have to be trained at a relatively high computational cost.

Many real-life large-scale datasets are open-ended and dynamic: new images are continuously added to existing classes, new classes appear over time, and the semantics of existing classes or concepts might evolve too. Therefore, the main objective of our work is to propose and study approaches which enable the addition of new classes and new images to existing classes at (near) zero cost. Such a method could be used continuously, while new images and classes become available, and additionally iterated from time to time with a (computationally heavier) method to learn a metric using all available data to ensure best performance. In this paper we explore two techniques which allow for adding new images and classes on the fly:

1. The k-nearest neighbor (k-NN) classifier, where images of new classes are added in the database, and can be used for classification without further processing. This is a highly non-linear and non-parametric classifier that has shown competitive performance for image annotation (Deng et al., 2010; Weston et al., 2011; Guillaumin et al., 2009a). Its main drawback is that the NN search for classification is computationally demanding for large and high-dimensional datasets.

2. The nearest class mean classifier (NCM), where classes are represented by their mean feature vector (Veenman and Tax, 2005; Zhou et al., 2008). The cost of computing the mean can be neglected with respect to the cost of feature extraction and this operation does not require accessing images of other classes. Contrary to the k-NN classifier, this is a linear classifier which leads to efficient classification. The complete distribution of the training data of a class is, however, only characterized by its mean and it is therefore unclear whether this is sufficient for competitive performance.

The success of both methods critically depends on the metric which is used to compute the distance between an image and other images (for k-NN) or class means (for NCM). Metric learning has received much attention in the machine learning and computer vision communities recently and has been shown to significantly increase the performance of distance-based classifiers. Therefore, we cast our classifier learning problem as one of learning a low-rank Mahalanobis distance which is shared across all classes. The dimensionality of the low-rank matrix is used as regularization parameter, and to reduce the classification and storage cost compared to a full-rank Mahalanobis distance.

In this paper we explore several strategies for learning such a metric. For k-NN classification, we use the Large Margin Nearest Neighbor (LMNN) framework (Weinberger et al., 2006) and investigate two variations similar to the ideas presented in (Checkik et al., 2010; Weinberger and Saul, 2009) that

significantly improves its classification performance, we also introduce an efficient gradient evaluation method. For the NCM classifier, we propose a novel metric learning algorithm based on multi-class logistic discrimination, where a sample from a class is enforced to be closer to its class mean than to any other class mean in the projected space.

This paper extends our earlier work (Mensink et al., 2012), as follows: First, for the k-NN classifier, in Section 3, we provide more technical details on the SGD triplet sampling strategy, and we present an efficient gradient evaluation method. Second, for the NCM classifier, in Section 4, we introduce an extensions which uses multiple centroids per class, we explore different learning objectives, and we examine the critical points of the low-rank objective. Third, we extend the experimental section by including an experiment where the NCM classifier is used to learn a metric for instance level image retrieval.

Most of our experiments are conducted on the ImageNet Large Scale Visual Recognition Challenge 2010 (ILSVRC'10) dataset, which consists of 1.2M train images of 1,000 classes. To apply the proposed metric learning techniques on such a large-scale dataset, we employ stochastic gradient descend (SGD) algorithms, which access only a small fraction of the training data at each iteration (Bottou, 2010). To allow metric learning on high-dimensional image features of large scale datasets that are too large to fit in memory, we use in addition product quantization (Gray and Neuhoff, 1998), a data compression technique that was recently used with success for large-scale image retrieval (Jégou et al., 2011) and classifier training (Sánchez and Perronnin, 2011). As a baseline approach, we use the state-of-the-art approach of (Sánchez and Perronnin, 2011), which was also the winning entry in the 2011 edition of the challenge: Fisher vector image representations (Perronnin et al., 2010) are used to describe images and one-vs-rest linear SVM classifiers are learned independently for each class. Surprisingly, we find that the NCM classifier outperforms the more flexible k-NN classifier. Moreover, the NCM classifier performs on par with the SVM baseline, and results are improved when we use the extension to allow for multiple centroids per class.

We also show the generalization performance to new classes of the k-NN and NCM classifiers. In a first experiment, we train the metric on a subset of 800 classes of ILSVRC'10 and include the 200 held-out classes at test time. We only observe a small drop in performance compared to the experiment where the metric is learned with all classes. In a second experiment, we train the metric on ILSVRC'10 and apply it to the larger ImageNet-10K dataset, which consist of 4.5M training images of 10K classes (Deng et al., 2010). Once the metric is learned, we learn the 10K classifiers on 64K dimensional features in less than an hour on a single CPU, while learning one-vs-rest linear SVMs on the same data takes on the order of 280 CPU days. Finally, we explore a zero-shot setting where we estimate the class mean of novel classes based on related classes in the ImageNet hierarchy. In an experiment, where we use zero training images for the novel classes, but only the class means based on the ImageNet hierarchy we obtain comparable results as in (Rohrbach et al., 2011). We also show that the zero-shot class mean can be effectively combined with the empirical mean of a small number of training images. This provides an approach that smoothly transitions from settings without training data to ones with abundant training data.

The rest of the paper is organized as follows. In Section 2 we discuss a selection of related work which is most relevant to this paper. In Section 3 we present metric learning techniques for k-NN classifiers, and in Section 4 we introduce the NCM classifier together with an extension to use multiple means (NCMC). We present extensive experimental results in Section 5, analyzing different aspects of the proposed methods and comparing them to the current state-of-the-art in different application settings such as large scale image annotation, transfer learning and image retrieval. Finally we discuss future research directions and our conclusions in Section 6.

2 Related Work

In this section we discuss some of the most relevant work on large-scale image annotation, metric learning, and transfer learning.

Large-scale image annotation. The ImageNet dataset (Deng et al., 2009) has been a catalyst for research on large-scale image annotation. The current state-of-the-art (Sánchez and Perronnin, 2011; Lin et al., 2011) uses efficient linear SVM classifiers trained in a one-vs-rest manner in combination with high-dimensional bag-of-word (Csurka et al., 2004; Zhang et al., 2007) or Fisher vector representations (Perronnin et al., 2010). Besides one-vs-rest training, large-scale ranking-based formulations have also been explored in (Weston et al., 2011). Interestingly, this approach performs joint classifier learning and dimensionality reduction of the image features. Operating in a lower-dimensional space acts as a regularization during learning, and also reduces the cost of classifier evaluation at test time. Our proposed NCM approach also learns low-dimensional projection matrices but the weight vectors are constrained to be the class means. This allows the efficient addition of novel classes.

In (Deng et al., 2010; Weston et al., 2011) k-NN classifiers were found to be competitive with linear SVM classifiers in a very large-scale setting involving 10,000 or more classes. The drawback of k-NN classifiers, however, is that they are expensive in storage and computation, since in principle all training data needs to be kept in memory and accessed to classify new images. The storage issue is also encountered when SVM classifiers are trained since all training data needs to be processed in multiple passes. Product quantization (PQ) was introduced in (Jégou et al., 2011) as a lossy compression mechanism for local SIFT descriptors in a bag-of-features image retrieval system. It has been subsequently used to compress bag-of-word and Fisher vector image representations in the context of image retrieval (Jégou et al., 2012) and classifier training (Sánchez and Perronnin, 2011). We also exploit PQ encoding in our work to compress high-dimensional image signatures when learning our metrics.

Metric learning. There is a large body of literature on metric learning, but here we limit ourselves to highlighting just several methods that learn metrics for (image) classification problems. Other methods aim at learning metrics for verification problems and essentially learn binary classifiers that threshold the learned distance to decide whether two images belong to the same class or not, see *e.g.* (Nowak and Jurie, 2007; Guillaumin et al., 2009b). Metric learning is also used for text retrieval, where documents are described by high-dimensional but sparse feature vectors, and ranked according to their relevance for a given query, see *e.g.* (Bai et al., 2010).

Among those methods that learn metrics for classification, the Large Margin Nearest Neighbor (LMNN) approach of (Weinberger et al., 2006; Weinberger and Saul, 2009) is specifically designed to support k-NN classification. It tries to ensure that for each image a predefined set of target neighbors from the same class are closer than samples from other classes. Since the cost function is defined over triplets of points —that can be sampled in an SGD training procedure— this method can scale to large datasets. In (Weinberger et al., 2006) the set of target neighbors is chosen and fixed using the ℓ_2 metric in the original space; this can be problematic as the ℓ_2 distance might be quite different than the optimal metric. Therefore, we explore two variants of LMNN that avoid using such a pre-defined set of target neighbors, similar to the ideas presented in (Checkik et al., 2010; Weinberger and Saul, 2009), both variants leading to significant improvements.

The large margin nearest local mean classifier (Chai et al., 2010) assigns a test image to a class based on the distance to the mean of its nearest neighbors in each class. This method was reported to outperform LMNN but requires computing all pairwise distances between training instances and therefore does not scale well to large datasets. Similarly, TagProp (Guillaumin et al., 2009a) suffers from the same problem, it consists in assigning weights to training samples based on their distance to the test instance and in

computing the class prediction by the total weight of samples of each class in a neighborhood. Because of their poor scaling properties, we do not consider these methods in our experiments.

Closely related to our metric learning approach for the NCM classifier is the LESS model of (Veenman and Tax, 2005). They learn a diagonal scaling matrix to modify the ℓ_2 distance by rescaling the data dimensions, and include an ℓ_1 penalty on the weights to perform feature selection. However, in their case, NCM is used to address small sample size problems in binary classification, *i.e*. cases where there are fewer training points (tens to hundreds) than features (thousands). Our approach differs significantly in that (i) we work in a multi-class setting and (ii) we learn a low-dimensional projection which allows efficiency in large-scale. The method of (Zhou et al., 2008) is also related to our method since they use a NCM classifier and an ℓ_2 distance in a subspace that is orthogonal to the subspace with maximum within-class variance. However, their technique involves computing the first eigenvectors of the within-class covariance matrix, which has a computational cost between $O(D^2)$ and $O(D^3)$, which again is undesirable for high-dimensional feature vectors. Moreover, this metric is heuristically obtained, rather than directly optimized for maximum classification performance.

Transfer learning. The term transfer learning is used to refer to methods that share information across classes during learning. Examples of transfer learning in computer vision include the use of part-based or attribute class representations. Part-based object recognition models (Fei-Fei et al., 2006) define an object as a spatial constellation of parts, and share the part detectors across different classes. Attribute-based models (Lampert et al., 2009) characterize a category (*e.g*. a certain animal) by a combination of attributes (*e.g*. is yellow, has stripes, is carnivore), and share the attribute classifiers across classes. Other approaches include biasing the weight vector learned for a new class towards the weight vectors of classes that have already been trained (Tommasi and Caputo, 2009). Zero-shot learning (Larochelle et al., 2008) is an extreme case of transfer learning where for a new class no training instances are available but a description is provided in terms of parts, attributes, or other relations to already learned classes. In (Rohrbach et al., 2011) various transfer learning methods were evaluated in a large-scale setting using the ILSVRC'10 dataset. They found transfer learning methods to have little added value when training images are available for all classes. In contrast, transfer learning was found to be effective in a zero-shot learning setting, where classifiers were trained for 800 classes, and performance was tested in a 200-way classification across the held-out classes.

In this paper we also aim at transfer learning, in the sense that we allow only a trivial amount of processing on the data of new classes (storing in a database, or averaging), and rely on a metric that was trained on other classes to recognize the new ones. In contrast to most work on transfer learning, we do not use any intermediate representation in terms of parts or attributes, nor do we train classifiers for the new classes. While also considering zero-shot learning, we further evaluate performance when combining a zero-shot model inspired by (Rohrbach et al., 2011) with progressively more training images per class, from one up to thousands. We find that the zero-shot model provides an effective prior when a small amount of training data is available.

3 Metric Learning for k-NN Classification

In this section we discuss metric learning for k-NN classifiers, where we follow and extend the approach of LMNN (Weinberger et al., 2006; Weinberger and Saul, 2009). We learn Mahalanobis distances of the form

$$d_M(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} - \boldsymbol{x}')^\top M (\boldsymbol{x} - \boldsymbol{x}'), \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are D dimensional vectors, and M is a positive definite matrix. We focus on low-rank metrics with $M = W^\top W$ and $W \in \mathbb{R}^{d \times D}$, where $d \leq D$ acts as regularizer and improves efficiency for computation and storage. We do not consider using the more general formulation of $M = W^\top W + S$,

where S is a diagonal matrix, as used in (Bai et al., 2010). This formulation still has a small number of parameters, $D(d+1)$, but still requires computing distances in the original high-dimensional space which is costly using the dense high-dimensional (4K-64K) features we use to represent images, see Section 5 for more details. The Mahalanobis distance induced by W is equivalent to the ℓ_2 distance after linear projection of the feature vectors on the rows of W:

$$d_W(\boldsymbol{x}, \boldsymbol{x}') = \ (\boldsymbol{x} - \boldsymbol{x}')^\top W^\top W(\boldsymbol{x} - \boldsymbol{x}') \ = \ \| W\boldsymbol{x} - W\boldsymbol{x}' \|_2^2 \, . \tag{2}$$

3.1 Large Margin Nearest Neighbor Metric Learning

For successful k-NN classification, the majority of the nearest neighbors should be of the same class. This is reflected in the metric learning approach of LMNN (Weinberger et al., 2006), their learning objective is based on triplets consisting of a query image q, a positive image p from the same class, and a negative image n from another class. The objective is to get the distance between query image q and positive image p smaller than the distance between query image q and negative image n. The 0/1-loss for such a triplet is upper-bounded by the hinge-loss on the distance difference:

$$L_{qpn} = \left[1 + d_W(\boldsymbol{x}_q, \boldsymbol{x}_p) - d_W(\boldsymbol{x}_q, \boldsymbol{x}_n)\right]_+, \tag{3}$$

where $[z]_+$ is the positive part of z, *i.e.* $\max(0, z)$. The hinge-loss is zero if the negative image n is at least one distance unit farther from the query q than the positive image p, and the loss is positive otherwise. The sum of the per-triplet loss is the final learning objective:

$$L = \sum_{q=1}^{N} L_q \tag{4}$$

$$L_q = \sum_{p \in P_q} \sum_{n \in N_q} L_{qpn}, \tag{5}$$

where P_q and N_q denote a predefined set of positive and negative images for each query image x_q. The sub-gradient of the loss of a triplet is given by:

$$\nabla_W L_{qpn} = [\![L_{qpn} > 0]\!] \ 2 \ W \left((\boldsymbol{x}_q - \boldsymbol{x}_p)(\boldsymbol{x}_q - \boldsymbol{x}_p)^\top - (\boldsymbol{x}_q - \boldsymbol{x}_n)(\boldsymbol{x}_q - \boldsymbol{x}_n)^\top \right), \tag{6}$$

where we use Iversons bracket notation $[\![\cdot]\!]$ that equals one if its argument is true, and zero otherwise.

In LMNN the set of targets P_q for a query q is set to the query's k nearest neighbors from the same class, using the ℓ_2 distance. The rationale is the following one: if we can ensure that these targets are closer than the instances of the other classes, then the k-NN classification will succeed. In practice, however, it is not always possible to achieve this goal with a given set of target neighbors, since it implicitly assumes that the ℓ_2 distance in the original space is a good similarity measure. Therefore, we consider two alternatives to using a fixed set of target neighbors:

1. The set of targets P_q is defined to contain all images of the same class as q. We refer to this method as 'All' in the experiments. This is similar to (Checkik et al., 2010) where the same type of loss was used to learn image similarity defined as the scalar product between images feature vectors after a learned linear projection.

2. The set of targets P_q is dynamically determined to contain the k images of the same class that are closest to q using the current metric. We refer to this method as 'Dynamic' in the experiments. This method corresponds to minimizing the loss function also with respect to the choice of P_q. Hence different target neighbors can be selected, that are closer than the original ones according to the current metric. A similar approach has been proposed in (Weinberger and Saul, 2009), where every T iterations P_q is redefined using target neighbors according to the current metric.

In the next section we propose an efficient gradient evaluation method, which allows to approximate the dynamic set of target neighbors at each iteration at a negligible additional cost compared to using a fixed set of target neighbors or using all images of the same class as target neighbors.

3.2 Efficient SGD Training: Triplet Sampling Strategy and Gradient Calculation

The objective function of the k-NN metric learning approach described in the previous section is defined over triplets of images. Even if we disregard the set of target neighbors, each query image is paired with all images of other classes as negative image. Thus, if we have N training images evenly distributed among C classes, this would yield already $N(N - \frac{N}{C})$ pairs. For ILSVRC'10, we have $N \approx 10^6$ and $C = 10^3$ which leads to roughly 10^{12} pairs, and to even more triplets.

Evaluating the exact gradient over all training images is computational unfeasible, therefore we rely on SGD training, where in each iteration we estimate the gradient using triplets from a limited set of $m \ll N$ images. Below we detail how we can increase the efficiency of the SGD training by using an appropriate sampling strategy to sample triplets of images, and by using an efficient algorithm for gradient evaluation.

Triplet sampling strategy. Using a small number of m images per SGD iteration is advantageous since the cost of the gradient evaluation is in large part determined by the cost of computing $W\boldsymbol{x}_i$, the projections of the m image signatures to the low dimensional space using projection matrix W, and the cost of decompressing the PQ encoded signatures if these are used.

In our sampling strategy, we first select uniformly at randomly a class c from which we will sample query images. We then sample ρm images from class c, with $0 < \rho < 1$, and the remaining $(m - \rho m)$ images are uniformly sampled from the other classes. We can consider the number of triplets t we can generate as a function of ρ for a given 'budget' of m images to be accessed. In the case where P_q is set according to the 'All' strategy, the number of triplets t we can generate for a specific ρ is given by:

$$t(\rho) = \frac{1}{2}(\rho m)(\rho m - 1)(m - \rho m), \tag{7}$$

since we can pair the ρm images with the $\rho m - 1$ other images from the same class, and each pair forms a triplet with any of the $m - \rho m$ negative sampled images. The number of triplets t can be approximated by:

$$t(\rho) \approx \frac{1}{2}m^3\rho^2(1 - \rho), \tag{8}$$

and hence, the number of triplets is maximized when we choose $\rho \approx \frac{2}{3}$. In our experiments, we use $\rho = \frac{2}{3}$ and $m = 300$ images per iteration, leading to about 4 million triplets per iteration.

Roughly the same number of triplets could also be generated by sampling two images for each of the $C = 1,000$ classes. In this case, there are two query images per class, forming a pair with the other positive image, and each pair can form a triplet with the $2(C - 1)$ images of other classes, leading to $4C(C - 1) \approx 4M$ triplets. In this manner, however, we would need to access $m = 2,000$ images, which is about 7 times more costly than using the described approach with $m = 300$.

When we use one of the other methods for P_q we do the following:

- For a fixed set of target neighbors, we still sample $\frac{m}{3}$ negative images, and take as many query images together with their target neighbors until we obtain $\frac{2m}{3}$ images allocated for the positive class.
- For a dynamic set of target neighbors we simply sample $\frac{2m}{3}$ positive and $\frac{m}{3}$ negative images, and select the dynamic target neighbors among the sampled positive images. Although approximate, this avoids computing the dynamic target neighbors among all the images in the positive class and still gives good results, see Section 5.

Efficient gradient evaluation. For either choice of the target set P_q, the gradient can be computed without explicitly iterating over all triplets, by making use of sorting distances w.r.t. query images. By observing that the gradient Eq. (6) takes the form of outer products of the feature vectors, we see that the gradient can also be written in matrix notation as:

$$\nabla_W L = (WX)AX^\top, \tag{9}$$

where X contains the feature vectors $\boldsymbol{x}_i$ of the m images used in a particular SGD iteration as columns, and A is a coefficient matrix. This shows that, once A is available, the gradient can be computed in time $O(m^2)$, even if a much larger number of triplets is used. Here we consider the case when all images of the same class as q are used as targets in P_q.

A closer look at the gradient reveals that:

$$\begin{aligned}\nabla_W L_q = &+2W \sum_p \Big(\sum_n [\![L_{qpn} > 0]\!] \Big) (\boldsymbol{x}_p \boldsymbol{x}_p^\top - \boldsymbol{x}_q \boldsymbol{x}_p^\top - \boldsymbol{x}_p \boldsymbol{x}_q^\top) \\ &- 2W \sum_n \Big(\sum_p [\![L_{qpn} > 0]\!] \Big) (\boldsymbol{x}_n \boldsymbol{x}_n^\top - \boldsymbol{x}_q \boldsymbol{x}_n^\top - \boldsymbol{x}_n \boldsymbol{x}_q^\top).\end{aligned} \tag{10}$$

Therefore, the coefficient matrix A can be computed from the number of hinge-loss generating triplets in which each $p \in P_q$ and each $n \in N_q$ occurs:

$$A_{qn} = 2 \sum_p [\![L_{qpn} > 0]\!], \qquad A_{pq} = -2 \sum_n [\![L_{qpn} > 0]\!], \tag{11}$$

$$A_{qq} = \sum_p A_{qp} - \sum_n A_{qn}, \qquad A_{pp} = \sum_q A_{qp}, \qquad A_{nn} = \sum_q A_{qn}. \tag{12}$$

To compute the coefficients A_{qn} and A_{qp} we use the following algorithm:

1. Sort all distances w.r.t. query q in ascending order; to account for the offset in the hinge-loss use for each positive image $d_W(\boldsymbol{x}_q, \boldsymbol{x}_p) + 1$ as distance.
2. Accumulate, from start to end, the number of negative images up to each position.
3. Accumulate, from end to start, the number of positive images after each position.
4. Read-off the number of hinge-loss generating triplets of image p or n w.r.t. query q.

The same algorithm can be applied when using a small set of fixed, or dynamic target neighbors. In particular, it allows to dynamically determine the target neighbors at a negligible additional cost, since the list is already sorted. In this case only the selected target neighbors obtain non-zero coefficients, therefore we simply accumulate the number of target neighbors after each position, instead of the number of positive images, in step 3 of the algorithm.

The cost of this algorithm is $O(m \log m)$ per query, and thus $O(m^2 \log m)$ when using $O(m)$ query images per iteration. This is significantly faster than explicitly looping over all $O(m^3)$ triplets to check if they generate a hinge-loss.

Note that while this algorithm enables fast computation of the gradient of the hinge-loss, the value of the hinge-loss itself cannot be determined using this method. However, this is not problematic if an SGD approach is used for learning, since it only requires gradient evaluations, not function evaluations.

4 Metric Learning for the Nearest Class Mean Classifier

In this section we define our NCM classifier, where we use a multi-class logistic regression objective to learn a low-rank Mahalanobis distance, see Eq. (2), between images and class means. We start with

introducing the basic model, and it relations to existing models. Then we present an extension to use multiple centroids per class, which transforms the NCM into a non-linear classifier. Finally we explore some variants of the objective which allow for smaller SGD batch sizes, and we give some insights in the critical points of the objective function.

4.1 Nearest Class Mean Classifier

We formulate the NCM classifier using multi-class logistic regression and define the probability for a class c given an image feature vector $\mathbf{x}$ as:

$$p(c|\boldsymbol{x}) = \frac{\exp\big(-d_W(\boldsymbol{x}, \boldsymbol{\mu}_c)\big)}{\sum_{c'=1}^{C} \exp\big(-d_W(\boldsymbol{x}, \boldsymbol{\mu}_{c'})\big)}, \tag{13}$$

where $\boldsymbol{\mu}_c$ is the mean of the feature vectors $\boldsymbol{x}_i$ from class $c \in \{1, \ldots, C\}$. This definition may also be interpreted as giving the posterior probabilities of a generative model where $p(\boldsymbol{x}_i|c) = \mathcal{N}(\boldsymbol{x}_i; \mu_c, \Sigma)$, a Gaussian with mean $\boldsymbol{\mu}_c$, and class-independent covariance matrix Σ, which is set such that $\Sigma^{-1} = W^\top W$. The class probabilities $p(c)$ are set to be uniform over all classes.

To learn the projection matrix W, we minimize the negative log-likelihood of the ground-truth class labels $y_i \in \{1, \ldots, C\}$ of the training images:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \ln p(y_i|\boldsymbol{x}_i). \tag{14}$$

The gradient of this objective function is easily verified to be:

$$\nabla_W \mathcal{L} = \frac{2}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} \Big([\![y_i = c]\!] - p(c|\boldsymbol{x}_i) \Big) W (\boldsymbol{\mu}_c - \boldsymbol{x}_i)(\boldsymbol{\mu}_c - \boldsymbol{x}_i)^\top. \tag{15}$$

4.2 Relation to Existing Models

The NCM classifier is linear in $\boldsymbol{x}_i$ since we assign an image i to the class c^* with minimum distance:

$$\begin{aligned} c^* &= \operatorname*{argmin}_c \big\{ d_W(\boldsymbol{x}_i, \boldsymbol{\mu}_c) \big\}, & (16)\\ &= \operatorname*{argmin}_c \big\{ \| W\boldsymbol{x}_i \|_2^2 + \| W\boldsymbol{\mu}_c \|_2^2 - 2\boldsymbol{\mu}_c W^\top W \boldsymbol{x}_i \big\}, & (17)\\ &= \operatorname*{argmin}_c \big\{ b_c - \boldsymbol{w}_c^\top \boldsymbol{x}_i \big\}, & (18) \end{aligned}$$

where the class specific bias b_c and weight vector $\boldsymbol{w}_c$ are defined as:

$$\begin{aligned} b_c &= \| W\boldsymbol{\mu}_c \|_2^2, & (19)\\ \boldsymbol{w}_c &= 2\boldsymbol{\mu}_c^\top (W^\top W). & (20) \end{aligned}$$

This observation allows us to relate the NCM classifier to other linear methods. For example, we obtain standard multi-class logistic regression, if the restrictions on b_c and $\boldsymbol{w}_c$ are removed. However, the restriction allows us to add new classes at near-zero cost, since the class specific parameters b_c and $\boldsymbol{w}_c$ are defined only using the class mean $\boldsymbol{\mu}_c$ and a class-independent metric W.

The NCM classifier is also closely related to the WSABIE method of (Weston et al., 2011). In the latter, a class c is scored using

$$f_c(\boldsymbol{x}_i) = \boldsymbol{v}_c^\top W \boldsymbol{x}_i, \tag{21}$$

where $W \in \mathbb{R}^{d \times D}$ is also a low-rank projection matrix, and $\boldsymbol{v}_c$ is a class specific weight vector, of dimensionality d, learned from data. In NCM however, we enforce that $\boldsymbol{v}_c = W\boldsymbol{\mu}_c$, which allows us to add new classes without the need to learn $\boldsymbol{v}_c$ from labeled data.

The NCM classifier can also be related to the solution of ridge-regression, or regularized linear least-squares regression, which also uses a linear score function $f_c(\boldsymbol{x}_i) = b_c + \boldsymbol{w}_c^\top \boldsymbol{x}_i$. However the parameters b_c and $\boldsymbol{w}_c$ are learned by optimizing the squared loss:

$$L_{\text{RR}} = \frac{1}{N} \sum_i \Big(f_c(\boldsymbol{x}_i) - y_{ic} \Big)^2 + \lambda \, \| \, \boldsymbol{w}_c \, \|_2^2, \tag{22}$$

where λ acts as regularizer, and where $y_{ic} = 1$, if image i belongs to class c, and $y_{ic} = 0$ otherwise. The ridge regression loss L_{RR} can be minimized in closed form and leads to

$$b_c = \frac{n_c}{N}, \quad \text{and} \qquad \boldsymbol{w}_c = \frac{n_c}{N} \boldsymbol{\mu}_c^\top (\Sigma + \lambda I)^{-1}, \tag{23}$$

where Σ is the (class-independent) data covariance matrix, and n_c denotes the number of images in class c. Just like the NCM classifier, the ridge-regression solution also allows to add new classes at low cost, the class specific parameters can be found from the class mean $\boldsymbol{\mu}_c$ and count n_c, once the data covariance matrix Σ has been estimated. Moreover, if n_c is equal for all classes, the score function is similar to our NCM classifier where we set W such that $W^\top W = (\Sigma + \lambda I)^{-1}$.

4.3 Non-Linear Classification using Multiple Centroids per Class

In this section we extend the NCM classifier to allow for more flexible class representations and non-linear classification, by using multiple centroids per class. Lets assume that we have obtained for each class a set of centroids $\mathcal{M}_c$, consisting of k centroids m_{cj}, then we define the posterior probability for class c for an image i as,

$$p(c|\boldsymbol{x}_i) = \sum_{j \in \mathcal{M}_c} p(j|\boldsymbol{x}_i) = \frac{1}{Z} \sum_{j \in \mathcal{M}_c} \exp\big(\, - d_W(\boldsymbol{x}_i, \boldsymbol{m}_{cj}) \, \big), \tag{24}$$

$$Z = \sum_c \sum_{j \in \mathcal{M}_c} \exp\big(\, - d_W(\boldsymbol{x}, \boldsymbol{m}_{cj}) \, \big), \tag{25}$$

where Z denotes the normalizer. This model also corresponds to a generative model, in this case the probability for an image feature vector $\boldsymbol{x}_i$, to be generated by a class, is given by a Mixture-of-Gaussians with equal mixing weights:

$$p(\boldsymbol{x}_i|c) = \frac{1}{|\mathcal{M}_c|} \sum_{j \in \mathcal{M}_c} \mathcal{N}\left(\boldsymbol{x}_i; \boldsymbol{m}_{cj}, \Sigma\right). \tag{26}$$

Also in this case the covariance matrix Σ is shared among all classes. We refer to this method as the Nearest Class Multiple Centroids (NCMC) classifier.

We obtain the k class centroids $\boldsymbol{m}_{cj}$ for by applying the k-means clustering algorithm to the image features $\boldsymbol{x}_i$ for all images i from class c, we use the image features before projection on W. Obviously, by setting $k = 1$, we obtain the NCM classifier. On the other hand, in the limit that each image in the train set is used as a class centroid, we obtain a formulation comparable to a k-NN classifier. In which each neighbour get a weight by the soft-min of its distance, according to Eq. (24), this is similar as in TagProp (Guillaumin et al., 2009a).

Given a set of class centroids, we learn the projection matrix W by minimizing the negative log-likelihood, Eq. (14), as before for the NCM classifier. The derivative of w.r.t. to W becomes:

$$\nabla_W \mathcal{L} = \sum_i \sum_c \sum_{j \in \mathcal{M}_c} \Big[p(j|\boldsymbol{x}_i) - p(j|\boldsymbol{x}_i, y_i) \Big] \, W(\boldsymbol{m}_{cj} - \boldsymbol{x}_i)(\boldsymbol{m}_{cj} - \boldsymbol{x}_i)^\top, \text{ and} \tag{27}$$

$$p(j|\boldsymbol{x}_i, y_i) = \begin{cases} \frac{p(j|\boldsymbol{x}_i)}{\sum_{j' \in \mathcal{M}_c} p(j'|\boldsymbol{x}_i)} & \text{if } j \in \mathcal{M}_{y_i}, \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

Instead of using a fixed set of class means, it could be advantageous to iterate the clustering to obtain the class means $\boldsymbol{m}_{cj}$, and learning the projection matrix W. Such a strategy allows the class means to represent the distribution of the images in the projected space more precise, and replacing class means which become redundant after projection. Therefore, such an iterative approach might improve the performance, however experimental validation of such a strategy falls beyond the scope of this paper.

4.4 NCM Objectives for Small SGD Batches

A disadvantage of the NCM and NCMC objectives is that each gradient evaluation requires the distance from an image to all means or centroids, due to the normalizer Z, and for each distance the class specific bias b_c, implies that each mean has to be projected on the metric W, *c.f.* Eq. (19). The cost of the projecting m image feature vectors $\boldsymbol{x}_i$ and C class mean vectors $\boldsymbol{\mu}_c$ on the matrix $W \in \mathbb{R}^{d \times D}$ is $O\big((m + C)Dd\big)$, and the cost of computing the distances between the m images and the C class means after projection equals $O(mCd)$. Thus, the total complexity to compute the Mahalanobis distances is $O\big(md(D + C) + CDd\big)$.

Since only the first term scales with the number of images m used in an SGD update, it is not advantageous to use $m \ll C = 1,000$ since the cost would be dominated by the second term in that case. This limits the size of the sample batches used in the SGD updates, and as a result each update is relatively expensive. Below, we consider two alternatives of the objective function that allow using smaller batch sizes, which allows for less costly SGD updates.

First, we consider replacing the multi-class logistic regression loss by a sum of binary one-versus-one losses,

$$L_{\text{sum}} = \sum_i \sum_{c \neq y_i} - \ln \sigma\big(d_W(\boldsymbol{x}_i, \boldsymbol{\mu}_c) - d_W(\boldsymbol{x}_i, \boldsymbol{\mu}_{y_i})\big), \tag{29}$$

where we use a log-sigmoid to provide a smooth and differentiable loss that encourages each class center to be further away from the sample than that of the ground-truth class. Clearly, the sum of binary one-versus-one losses provides a bound on the multi-class zero-one loss: if the multi-class loss is 1 at least one of the binary losses is 1 as well. This bound is tight if the multi-class loss is zero, but may be looser otherwise. The same sum of one-versus-one loss terms is used in the Multi-Class SVM formulation of (Weston and Watkins, 1999), albeit there a hinge-loss is used instead of the log-loss. We can also relate L_{sum} to objective functions used in ranking problems (Joachims, 2002; Grangier and Bengio, 2008) and LMNN (Weinberger et al., 2006), if we see the image vector $\boldsymbol{x}_i$ as a query and the objective is to rank classes according to their distance. The loss is defined as a sum over triplets (i, y_i, c), where for each triplet a penalty is incurred if the non-relevant class c is ranked higher than the relevant class y_i according to the metric W. For LMNN the used per triplet penalty is the hinge-loss, see Eq. (3).

The interest of the L_{sum} objective is that it decomposes additively over the classes, this allows to sample per SGD iteration up to a single triplet (i, y_i, c), or a small number of $m \ll C$ triplets, to estimate the gradient at a cost of $O(mDd)$. Unfortunately, we experimentally find that optimizing for the L_{sum} objective decreases performance significantly compared to the original multi-class logistic regression objective of Eq. (14).

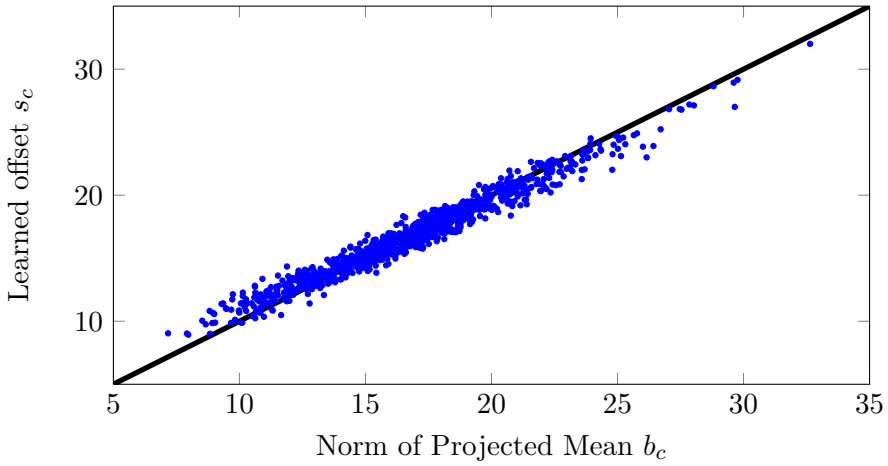


Figure 1: A strong correlation is observed between class-specific bias s_c and the norm of the mean b_c computed after projection on W.

Second, we consider replacing the Euclidean distance in Eq. (13) by the negative dot product plus a class specific bias s_c, the probability for class c is then defined as

$$p(c|\boldsymbol{x}_i) = \frac{1}{Z} \exp\left(2\, \boldsymbol{x}_i^\top W^\top W \boldsymbol{\mu}_c - s_c\right), \tag{30}$$

where Z denotes the normalizer. The objective is still to minimize the multi-class logistic regression loss of Eq. (14). The efficiency gain using this formulation stems from the fact that the norm of the projected mean, $b_c = \| W\mu_c \|_2^2$ that appears in the Euclidean distance is replaced by the scalar bias s_c. Therefore, we can avoid projecting the mean-vectors on W, by twice projecting the sample vectors: $\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i^\top W^\top W$, and then computing dot products in high dimensional space $\langle \hat{\boldsymbol{x}}_i, \boldsymbol{\mu}_c \rangle$. For a batch of m images, the first step cost $O(mDd)$, and the latter $O(mCD)$, resulting in a complexity of $O(mD(d+C))$ as compared to $O(md(D+C)+CDd)$ for the Euclidean distance. The dot product formulation is therefore much more efficient when using small batches of $m \ll C$ images.

Experimentally we find that using formulation yields comparable results as using the Euclidean distance. A potential disadvantage of this approach, however, is that we need to determine the class-specific bias s_c when data of a new class becomes available, which would require more training than just computing the data mean for the new class. Interestingly, we find a strong correlation between the learned bias s_c and the norm of the projected mean b_c, shown in Figure 1. Indeed, the classification performance differs insignificantly if at evaluation time we set $s_c = \| W\mu_c \|_2^2 = b_c$ instead of the value that was found during training. Thus, eventhough we train the metric by using class-specific biases, we can use the learned metric in the NCM classifier where we replace the bias with the norm of the projected mean, which is easily computed for data of new classes.

4.5 Critical Points of Low Rank Approximation

We use a low-rank approximation of the Mahalanobis distance where $M = W^\top W$, *c.f.* Eq. (2), as a way to reduce the number of parameters and to gain computational efficiency. Learning a full Mahalanobis distance matrix M, however, has the advantage that the distance is linear in M and that the multi-class logistic regression objective of Eq. (14) is therefore convex in M. Using a low-rank formulation, on the other hand, yields a distance which is quadratic in the parameters W, therefore the objective function

is then no longer convex. In this section we investigate the critical-points of a low-rank formulation by analyzing W when the optimization reaches a (local) minimum, and considering the gradient for the corresponding full matrix $M = (W)^\top W$.

The gradients of the NCM objective w.r.t. to M and respectively W are:

$$\nabla_M \mathcal{L} = \frac{1}{N} \sum_{i,c} \alpha_{ic}\, \boldsymbol{z}_{ic} \boldsymbol{z}_{ic}^\top \equiv H, \tag{31}$$

$$\nabla_W \mathcal{L} = \frac{2}{N} \sum_{i,c} \alpha_{ic}\, W \boldsymbol{z}_{ic} \boldsymbol{z}_{ic}^\top \equiv 2WH, \tag{32}$$

where $\alpha_{ic} = [\![y_i = c]\!] - p(c|\boldsymbol{x}_i)$, and $\boldsymbol{z}_{ic} = \boldsymbol{\mu}_c - \boldsymbol{x}_i$. From the gradient w.r.t. W we immediately observe a $W = 0$ leads to a degenerate case to obtain a zero gradient (the same applies separately per row of W). Below we concentrate on the non-degenerate case.

We observe that H is a symmetric matrix, containing the difference of two positive definite matrices. In the analysis we use the eigenvalue decomposition of $H = V \Lambda V^\top$, with the columns of V being the eigenvectors, and the eigenvalues are on the diagonal of Λ.

We can now express the gradient for W as

$$\nabla_W \mathcal{L} = 2WV\Lambda V^\top. \tag{33}$$

Thus the gradient of the i-th row of W, which we denote by $\boldsymbol{g}_i$, is given by a linear combination of the eigenvectors of H:

$$\boldsymbol{g}_i \equiv \sum_j \lambda_j \langle \boldsymbol{w}_i, \boldsymbol{v}_j \rangle \boldsymbol{v}_j, \tag{34}$$

where $\boldsymbol{w}_i$ and $\boldsymbol{v}_j$ denote the i-th row of W and the j-th column of V respectively. Thus an SGD gradient update will drive a row of W towards the eigenvectors of H that (i) have a large positive eigenvalue, and (ii) are most aligned with that row of W. This is intuitive, since we would expect the low-rank formulation to focus on the most significant directions of the full-rank metric.

Moreover, the expression for the gradient in Eq. (34) shows that at a critical point of the objective function W^*, we have that all linear combination coefficients are zero, *i.e.* $\forall_{i,j} : \lambda_j \langle \boldsymbol{w}_i^*, \boldsymbol{v}_j \rangle = 0$. This shows that at the critical point, for each row $\boldsymbol{w}_i^*$ of W^* and each eigenvector $\boldsymbol{v}_j$ holds that either (a) $\boldsymbol{w}_i^*$ is orthogonal to $\boldsymbol{v}_j$, or (b) that $\boldsymbol{v}_j$ has a zero associated eigenvalue, *i.e.* $\lambda_j = 0$. Thus, at a critical point W^*, the corresponding gradient for the full rank formulation at that point (*i.e.* with $M^* = (W^*)^\top W^*$) is zero in the subspace spanned by W^*.

Given this analysis, we believe it is unlikely to attain poor local minima using the low rank formulation: the gradient updates for W are aligned with the most important directions of the corresponding full-rank gradient, and at convergence the full-rank gradient is zero in the subspace spanned by W. To confirm this, we have also experimentally investigated this by training several times when starting from different random initializations of W, and using different random sampling of the sampled seen in each SGD iteration. We observe that the classification performance difference on the converged metric is at most $\pm.1\%$ on any of the error measures used, and that the number of SGD iterations selected by the early stopping procedure are of the same order.

5 Experimental Evaluation

In this section we experimentally validate our models described in the previous sections. We first describe the dataset and evaluation measures used in our experiments, followed by the presentation of our results for k-NN classification and NCM classification using metric learning.

5.1 Experimental Setup and Baseline Approach

Dataset. In most of our experiments we use the dataset of the ImageNet Large Scale Visual Recognition 2010 challenge (ILSVRC'10)[1]. This dataset contains 1.2M training images of 1,000 object classes (with between 660 to 3047 images per class), an validation set of 50K images (50 per class), and a test set of 150K images (150 per class).

In some of the experiments we use the ImageNet-10K dataset, introduced in (Deng et al., 2010), which consists of 10,184 classes from the nodes of the ImageNet hierarchy with more than 200 images. We follow Sánchez and Perronnin (2011) and use 4.5M images as training set, 50K as validation set and the rest as test set.

Features. We represent each image, with a Fisher vector (FV) (Perronnin et al., 2010) computed over densely extracted 128 dimensional SIFT descriptors (Lowe, 2004) and 96 dimensional local color features (Perronnin et al., 2010), both projected with PCA to 64 dimensions. FVs are extracted and normalized separately for both channels and then combined by concatenating the two feature vectors. We do not make use of spatial pyramids. In our experiments we use FVs extracted using a vocabulary of either 16 or 256 Gaussians. For 16 Gaussians, this leads to a 4K dimensional feature vector, which requires about 20GB for the 1.2M training set (using 4-byte floating point arithmetic). This fits into the RAM of our 32GB servers.

For 256 Gaussians, the FVs are 16 times larger, *i.e.* 64K dimensional, which would require 320GB of memory. Hence, we compress the feature vectors using product quantization (Gray and Neuhoff, 1998; Jégou et al., 2011). In a nutshell, it consists in splitting the high-dimensional vector into small sub-vectors, and vector quantizing each sub-vector independently. We compress the dataset to approximately 10GB using 8-dimensional sub-vectors and 256 centroids per sub-quantizer, which allows storing each sub-quantizer index in a single byte. In each iteration of SGD learning, we decompress the features of a limited number of images, and use these (lossy) reconstructions for the gradient computation.

Evaluation measures. We report the average top-1 and top-5 flat error used in the ILSVRC'10 challenge. The flat error is one if the ground-truth label does not correspond to the top-1 label with highest score (or any of the top-5 labels), and zero otherwise. The idea for using the top-5 error instead of the top-1 error, is to allow an algorithm to identify multiple objects in an image and not be penalized if one of the objects identified was in fact present, but not included in the ground truth (images are annotated with only a single object/topic).

Unless specified otherwise, we report the top-5 flat error on the test set using the 4K dimensional features; we use the validation set for parameter tuning only. In tables, we highlight the best result per row in bold, and do so for each feature set if several are used. Additionally, the baseline performance is also highlighted if it is best.

Baseline approach. For our baseline, we follow the state-of-the-art approach of (Perronnin et al., 2012) and learn weighed one-vs-rest SVMs with SGD, where the number of negative images in each iteration is sampled proportional to the number of positive images for that class. The results of the baseline can be found in Table 2 and Table 5. We see that the 64K dimensional features lead to significantly better results than the 4K ones, despite the lossy PQ compression.

In Table 2 the performance using the 64K features is slightly better than the ILSVRC'10 challenge winners (Lin et al., 2011) (28.0 vs. 28.2 flat top-5 error), and very close to the results of (Sánchez and Perronnin, 2011) (25.7 flat top-5 error), wherein a much higher dimensional image representation of more

[1] See http://www.image-net.org/challenges/LSVRC/2010/index

		k-NN classifiers						
	SVM	ℓ_2	ℓ_2	LMNN		All	Dynamic	
	Full	Full	+ PCA	10	20		10	20
Flat top-1 error	**60.2**	75.4	77.2	72.9	72.8	67.9	**65.2**	66.0
Flat top-5 error	**38.2**	55.7	57.3	50.6	50.4	44.2	**39.7**	40.7

Table 1: Comparison of different k-NN classification methods 4K dimensional features. For all methods, except those indicated by 'Full', the data is projected to a 128 dimensional space.

than 1M dimensions was used. In Table 5 our baseline shows state-of-the-art performance on ImageNet-10K when using the 64K features, obtaining 78.1 vs 81.9 flat top-1 error (Perronnin et al., 2012). We believe this is due to the use of the color features in the image representations.

SGD training and early stopping. To learn the projection matrix W, we use SGD training and sample at each iteration a fixed number of m training images to estimate the gradient. Following (Bai et al., 2010) , we use a fixed learning rate and do not include an explicit regularization term, but rather use the projection dimension $d \leq D$, as well as the number of iterations as an implicit form of regularization. For all experiments we use the following early stopping strategy: (1) we run SGD training for a large number of iterations ($\approx$ 750K-2M), (2) the performance on the validation set is computed every 50k iterations (for the k-NN) or every 10k iterations (for the NCM), and (3) the metric which yields the lowest top-5 error is selected. If on par the metric giving the lowest top-1 error is chosen. Similarly, all hyper-parameters, like the value of k for the k-NN classifiers, are validated in this way. Unless stated otherwise, training is performed using the ILSVRC'10 training set, and validation using the described early stopping strategy on the provided 50K validation set.

It is interesting to notice that while the compared methods (k-NN, NCM, and SVM) have different computational complexities, the number of images seen by each algorithm before convergence is rather similar. For example, training of the SVMs, on the 4K features, converge after $T \approx 100$ iterations, and each iteration takes about 64 negative images per positive image, per class. In the ILSVRC'10 dataset, each class has roughly $p = 1,200$ positive images, and consist of $C = 1,000$ classes. Therefore the total number of images seen during training of the SVMs is $TC(65p) = 7.800$M images. The NCM classifier requires much more iterations, $T \approx 500$K, but uses each iteration only $m = 1,000$ images, and trains only a single metric. Therefore the total number of images seen during training is roughly $Tm = 500$M. And finally, the k-NN classifier, requires even more iterations, $T \approx 1.8$M, but uses only $m = 300$ images per iteration, the total number of images seen before convergence is therefore only $Tm = 540$M.

5.2 k-NN Metric Learning Results

We start with an assessment of k-NN classifiers using metrics learned with the methods described in Section 3, and consider the impact of the different choices for the set of target images P_q, and the projection dimensionality. Given the cost of k-NN classifiers, we focus our experiments on the 4K dimensional features. We initialize W as a PCA projection, and determine the number of nearest neighbors to use for classification on the validation set; typically 100 to 250 neighbors are optimal.

Target selection for k-NN metric learning. In the first experiment we compare the three different options of Section 3 to define the set of target images P_q, while learning projections to 128 dimensions. For LMNN and dynamic targets, we experimented with various numbers of targets on the validation set and found that using 10 to 20 targets yields the best results.

The results in Table 1 show that all methods lead to metrics that are better than the ℓ_2 metric in the original space, or after a PCA projection to 128 dimensions. Furthermore, we can improve over LMNN

	4K dimensional features							64K dimensional features			
Projection dim.	32	64	128	256	512	1024	Full	128	256	512	Full
SVM baseline							38.2				**28.0**
k-NN, dynamic 10	47.2	42.2	39.7	**39.0**	39.4	42.4					
NCM, learned metric	49.1	42.7	39.0	37.4	**37.0**	**37.0**		31.7	31.0	**30.7**	
NCM, PCA + ℓ_2	78.7	74.6	71.7	69.9	68.8	68.2	**68.0**				**63.2**
NCM, PCA + inv. cov.	75.5	67.7	60.6	54.5	49.3	46.1	**43.8**				
Ridge-regression, PCA	86.3	80.3	73.9	68.1	62.8	58.9	**54.6**				
WSABIE	51.9	45.1	41.2	39.4	38.7	**38.5**		32.2	30.1	**29.2**	

Table 2: Performance of k-NN and NCM classifiers, as well as baselines, using the 4K and 64K dimensional features, for various projection dimensions, and comparison to related methods, see text for details.

by using all within-class images as targets, or even further by using dynamic targets. The success of the dynamic target selection can be explained by the fact that among the three alternatives, it is the most closely related to the k-NN classifier objective. The best performance on the flat top-5 error of 39.7 using 10 dynamic targets is, however, slightly worse than the 38.2 error rate of the SVM baseline.

Impact of projection dimension on k-NN classification. Next, we evaluate the influence of the projection dimensionality d on the performance, by varying d between 32 and 1024. We only show results using 10 dynamic targets, since this performed best among the evaluated k-NN methods. From the results in Table 2 we see that a projection to 256 dimensions yields the lowest error of 39.0, which still remains somewhat inferior to the SVM baseline.

5.3 Nearest Class Mean Classification Results

We now consider the performance of NCM classifiers and the related methods described in Section 4. In Table 2 we show the results for various projection dimensionalities.

We first consider the results for the 4K dimensional features. Our first, unexpected, observation is that our NCM classifier (37.0) outperforms the more flexible k-NN classifier (39.0), and even slightly outperforms the SVM baseline (38.2) when projecting to 256 dimensions or more. Interestingly, using just the ℓ_2 distance, instead of a learned metric, the situation is reversed and the k-NN classifier is better (55.7, see Table 1) than the NCM classifier (68.0). Our implementation of WSABIE (Weston et al., 2011) scores slightly worse (38.5) than the baseline and our NCM classifier, and does not generalize to new classes without retraining. Ridge-regression, like the NCM classifier, does allow generalization to new classes, but leads to significantly worse results (54.6) and pre-processing the data with PCA further degrades its performance.

We also consider two variants of the NCM classifier where we use PCA to reduce the dimensionality. In one case we use the ℓ_2 metric after PCA. In the other, inspired by ridge-regression, we use NCM with the metric W generated by the inverse of the regularized covariance matrix, such that $W^\top W = \Sigma + \lambda I^{-1}$, see Section 4.2. We tuned the regularization parameter λ on the validation set, as was also done for ridge-regression. From these results we can conclude that, just like for k-NN, the ℓ_2 metric with or without PCA leads to poor results (68.0) as compared to a learned metric. Second, the feature whitening implemented by the inverse covariance metric leads to results (43.8) that are better than using the ℓ_2 metric, and also substantially better than ridge-regression (54.6). The results are however significantly worse than using our learned metric, in particular when using low-dimensional projections.

When we use the 64K dimensional features, the results of the NCM classifier (30.8) are somewhat

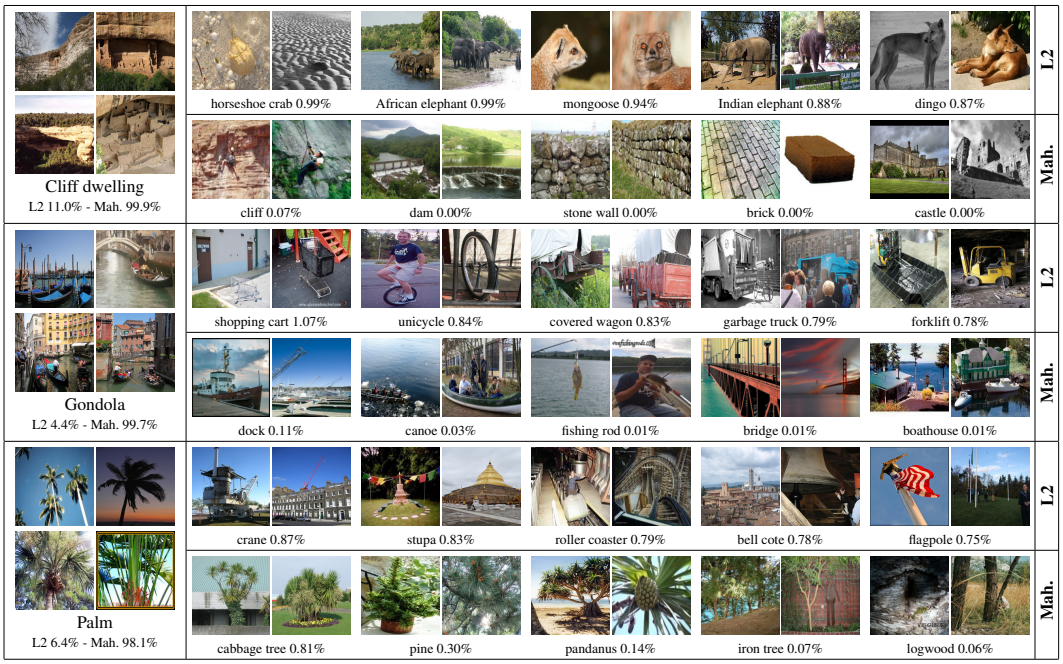


Figure 2: The nearest classes for three reference classes using the the ℓ_2 distance and Mahalanobis metric learned for the NCM classifier. Class probabilities are given for a simulated image signature that equals the mean of the reference class, see text for details.

	NCM	NCMC-test		NCMC		
Proj. Dim.			*(k)*	5	10	15
128	39.0	36.3	*(30)*	36.2	**35.8**	36.1
256	37.4	36.1	*(20)*	35.0	**34.8**	35.3
512	37.0	36.2	*(20)*	34.8	**34.6**	35.1

Table 3: The top-5 performance of the NCMC classifier using the 4K features, compared to the NCM baseline and the best NCMC-test classifier (with the value of k in brackets).

worse than the SVM baseline (28.0); again the learned metric is significantly better than using ℓ_2 distance (63.2). WSABIE obtains an error of 29.2, in between the SVM and NCM.

Influence of metric learning on semantic class neighbors. In Figure 2 we illustrate the difference between the ℓ_2 and the Mahalanobis metric induced by a learned projection from 64K to 512 dimensions. For three reference classes we show the five nearest classes, based on the distance between class means. We also show the probabilities on the reference class and its five neighbor classes according to Eq. (13). The feature vector $\boldsymbol{x}$ is set as the mean of the reference class, *i.e.* a simulated perfectly typical image of this class. For the ℓ_2 metric, we used our metric learning algorithm to learn a scaling of the ℓ_2 metric to minimize Eq. (14). This does not change the ordering of classes, but ensures that we can compare probabilities computed using both metrics. We find that, as expected, the learned metric has more semantically related class neighbors. Moreover, we see that using the learned metric most of the probability mass is assigned to the reference class, whereas the ℓ_2 metric leads to rather uncertain classifications.

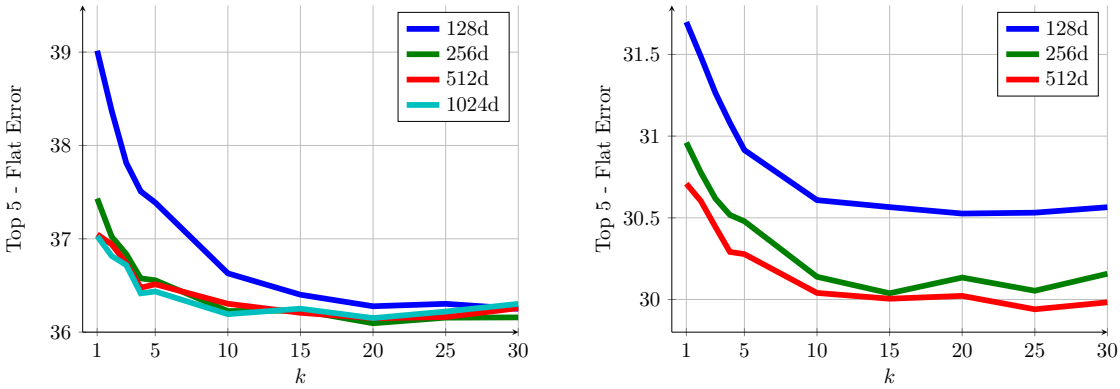


Figure 3: The top-5 performance of the NCMC-test classifier, which at test time uses $k > 1$ on a metric obtained with $k = 1$, for the 4K features (*left*) and the 64k (*right*) features.

Non-linear classification using multiple class centroids. In these experiments we use the non-linear NCMC classifier, introduced in Section 4.3, where each class is represented by a set of k centroids. We obtain the k centroids per class by using the k-means algorithm in the ℓ_2 space, if we set $k = 1$ we obtain the NCM classifier. Since the cost of training these classifiers is much higher, we run two sets of experiments.

In Figure 3, we show the performance of using the NCMC classifier at test time with $k = [2, \ldots, 30]$, while using a metric obtained by the NCM objective ($k = 1$), this method is denoted as NCMC-test. For each value of k the early stopping strategy is used to determine the best metric. In Table 3, we show the performance of the NCMC classifier, trained with the NCMC objective, using the 4K features. In the same table we compare the results to the NCM method and the best NCMC-test method.

From the results we observe that a significant performance improvement can be made by using the non-linear NCMC classifier, especially when using a low number of projection dimensionalities. In the case we use the 4K features with 128 projection dimensions, we improve 3.2 absolute points by training using the NCMC objective over the NCM objective. For the other projection dimensionalities, using NCMC-test yields a moderate improvement of about 1 absolute point. Apparently, in this setting the non-linear classification with higher projection dimensionalities, adds less to the discriminant power of the linear NCM classifier.

When learning using the NCMC classifier we can further improve the performance of the non-linear classification, albeit for a higher training cost. When using as little as 512 projection dimensions, we obtain the very impressive performance of 34.6 on the top-5 error, using $k = 10$ centroids. That is an improvement of about 2.4 absolute points over the NCM classifier (37.0), and 3.6 absolute points over SVM classification (38.2), *c.f.* Table 2.

5.4 Generalization to New Classes and Using Few Samples

Given the encouraging classification accuracy of the NCM classifier observed above —and its superior efficiency as compared to the k-NN classifier— we now explore its ability to generalize to novel classes. We also consider its performance as a function of the number of training images available to estimate the mean of novel classes.

Generalization to novel classes not seen during training. In this experiment we use approximately 1M images corresponding to 800 random classes to learn metrics, and evaluate the generalization perfor-

	4K dimensional features									64K dimensional features				
	SVM	k-NN			NCM					SVM	NCM			
Projection dim.	Full	128	256	Full	128	256	512	1024	Full	Full	128	256	512	Full
No training				54.2					66.6					61.9
Trained on all	37.6	39.0	38.4		38.6	36.8	**36.4**	36.5		**27.7**	31.7	30.8	**30.6**	
Trained on 800		42.2	42.4		42.5	40.4	39.9	**39.6**			39.3	**37.8**	**37.8**	

Table 4: Performance of 1,000-way classification among test images of 200 classes not used for metric learning, and control setting with metric learning using all classes.

	4K dimensional features						64K dimensional features					Previous Results		
Method	NCM					SVM	NCM				SVM	SVM		
Proj. dim.	128	256	512	1024	Full	Full	128	256	512	Full	Full	21K	131K	128K
Top-1 error	91.8	90.6	90.5	**90.4**	95.5	**86.0**	87.1	**86.3**	**86.1**	93.6	**78.1**	93.6	83.3	81.9
Top-5 error	80.7	78.7	**78.6**	**78.6**	89.0	**72.4**	71.7	70.5	**70.1**	85.4	**60.9**			

Table 5: Performance of the NCM classifier on the ImageNet-10K dataset, using metrics learned on the ILSVRC'10 dataset, and comparisons to the baseline SVM, the NCM using ℓ_2 distance (denoted as full), and previously reported results: SVM results with 21K dimensional features (Deng et al., 2010), 131K dimensional features (Sánchez and Perronnin, 2011), and 128K dimensional features (Perronnin et al., 2012).

mance on 200 held-out classes. The error is evaluated in a 1,000-way classification task, and computed over the 30K images in the test set of the held-out classes, the early stopping strategy uses the validation set of the 200 unseen classes. Performance among test images of the 800 train classes changes only marginally and would obscure the changes among the test images of the 200 held-out classes.

In Table 4 we show the performance of NCM and k-NN classifiers for several projection dimensions, and compare it to the control setting where the metric is trained on all 1,000 classes. The results show that both classifiers generalize remarkably well to new classes. For comparision we also include the results of the SVM baseline, and the k-NN and NCM classifiers using the ℓ_2 distance, evaluated over the 200 held-out classes. In particular for 1024 dimensional projections of the 4K features, the NCM classifier achieves an error of 39.6 over classes not seen during training, as compared to 36.5 when using all classes for training. For the 64K dimensional features the drop in performance is larger, but it is still surprisingly good considering that training for the novel classes consists only in computing their means.

To further demonstrate the generalization ability of the NCM classifier using learned metrics, we also compare it against the SVM baseline on the ImageNet-10K dataset. We use projections learned and validated on the ILSVRC'10 dataset, and only compute the means of the 10K classes. The results in Table 5 show that even in this extremely challenging setting the NCM classifier performs remarkably well compared to earlier mentioned SVM-based results of (Deng et al., 2010; Sánchez and Perronnin, 2011) and our baseline, all of which require training 10K classifiers. We note that, to the best of our knowledge, our baseline results exceed the previously known state-of-the-art (Deng et al., 2010; Sánchez and Perronnin, 2011) on this dataset. Training our SVM baseline system took 9 and 280 CPU days respectively for the 4K and 64K features, while the computation of the means for the NCM classifier took approximately **3** and **48** CPU minutes respectively. This represents roughly a *8,500 fold speed-up* as compared to the baseline, without counting the time to learn the projection matrix.

Accuracy as a function of the number of training images of novel classes. In this experiment we consider the error as a function of the number of images that are used to compute the means of novel classes. Inspired by (Rohrbach et al., 2011), we also include results of a zero-shot learning experiment,

where we use the ImageNet hierarchy to estimate the mean of novel classes from the means of related training classes. We follow ideas of (Rohrbach et al., 2011) and estimate the mean of a novel class $\boldsymbol{\mu}_z$ based on the means of all its ancestor nodes in the ILSVRC'10 class hierarchy:

$$\boldsymbol{\mu}_z = \frac{1}{|\mathcal{A}_z|} \sum_{a \in \mathcal{A}_z} \boldsymbol{\mu}_a, \tag{35}$$

where $\mathcal{A}_z$ denotes the set of ancestors of node z, and $\boldsymbol{\mu}_a$ is the mean of ancestor a. The mean of an internal node, $\boldsymbol{\mu}_a$, is computed as the average of the means of all its descendant training classes.

If we view the estimation of each class mean as the estimation of the mean of a Gaussian distribution, then the mean of a sample of images $\boldsymbol{\mu}_s$ corresponds to the Maximum Likelihood (ML) estimate, while the zero-shot estimate $\boldsymbol{\mu}_z$ can be thought of as a prior. We can combine the prior with the ML estimate to obtain a maximum a-posteriori (MAP) estimate $\boldsymbol{\mu}_p$ on the class mean. The MAP estimate of the mean of a Gaussian is obtained by

$$\boldsymbol{\mu}_p = \frac{n\boldsymbol{\mu}_s + m\boldsymbol{\mu}_z}{n+m}, \tag{36}$$

where the ML estimate is weighed by n the number of images that were used to compute it, and the prior mean obtains a weight m determined on the validation set (Gauvain and Lee, 1994),

In Figure 4 we analyze the performance of the NCM classifier trained on the images of the same 800 classes used above, with a learned projection from 4K and 64K to 512 dimensions, the metric and the parameter m are validated using the images of the 200 held out classes of the validation set. We again report the error among test images of the held-out classes, both in a 1,000-way classification as above, and in a 200-way classification as in (Rohrbach et al., 2011). We repeat the experiment 10 times, and show error-bars at three times standard deviation. For the error to stabilize we only need approximately 100 images to estimate the class means. The results also show that the prior leads to zero-shot performance of 66.5 (4K features) and 64.0 (64K features). These results are comparable to the result of 65.2 reported in (Rohrbach et al., 2011), even though a different set of 200 test-classes were used. Note that they also used different features, however their baseline performance of 37.6 top-5 error is comparable to our 4K features (38.2). More importantly, we show that the zero-shot prior can be effectively combined with the empirical mean to provide a smooth transition from the zero-shot setting to a setting with many training examples. Inclusion of the zero-shot prior leads to a significant error reduction in the regime where ten images or less are available.

Instance level image retrieval. Query-by-example image retrieval can be seen as an image classification problem where only a single positive sample (the query) is provided and negative examples are not explicitly provided. In this case the class mean simplifies to the query, and we use a metric learned for our NCM classifier on an auxiliary supervised dataset to retrieve the most similar images for a given query.

Using classifiers to learn a metric for image retrieval was recently also considered in (Gordo et al., 2012). They found the Joint Subspace and Classifier Learning (JSCL) method to be most effective. This basically amounts to jointly learning a set of classifiers and a projection matrix W using the WSABIE scoring function, Eq. (21), and minimizing the hinge-loss on class labels. After training the classifiers are discarded and only the learned projection matrix W is used to compute distances between query and database images.

For this experiment we use the same public benchmarks used in (Gordo et al., 2012). First, the INRIA Holidays data set introduced by (Jégou et al., 2008) consists of 1,491 images of 500 scenes and objects. In the standard evaluation protocol, one image per scene / object is used as query to search withing the remaining images; the accuracy is measured as the mean average precision over the 500 queries (mAP). Second, the University of Kentucky Benchmark dataset (UKB) introduced by (Nistér and Stewénius, 2006) contains 10,200 images of 2,550 objects. We follow the standard evaluation protocol, where each

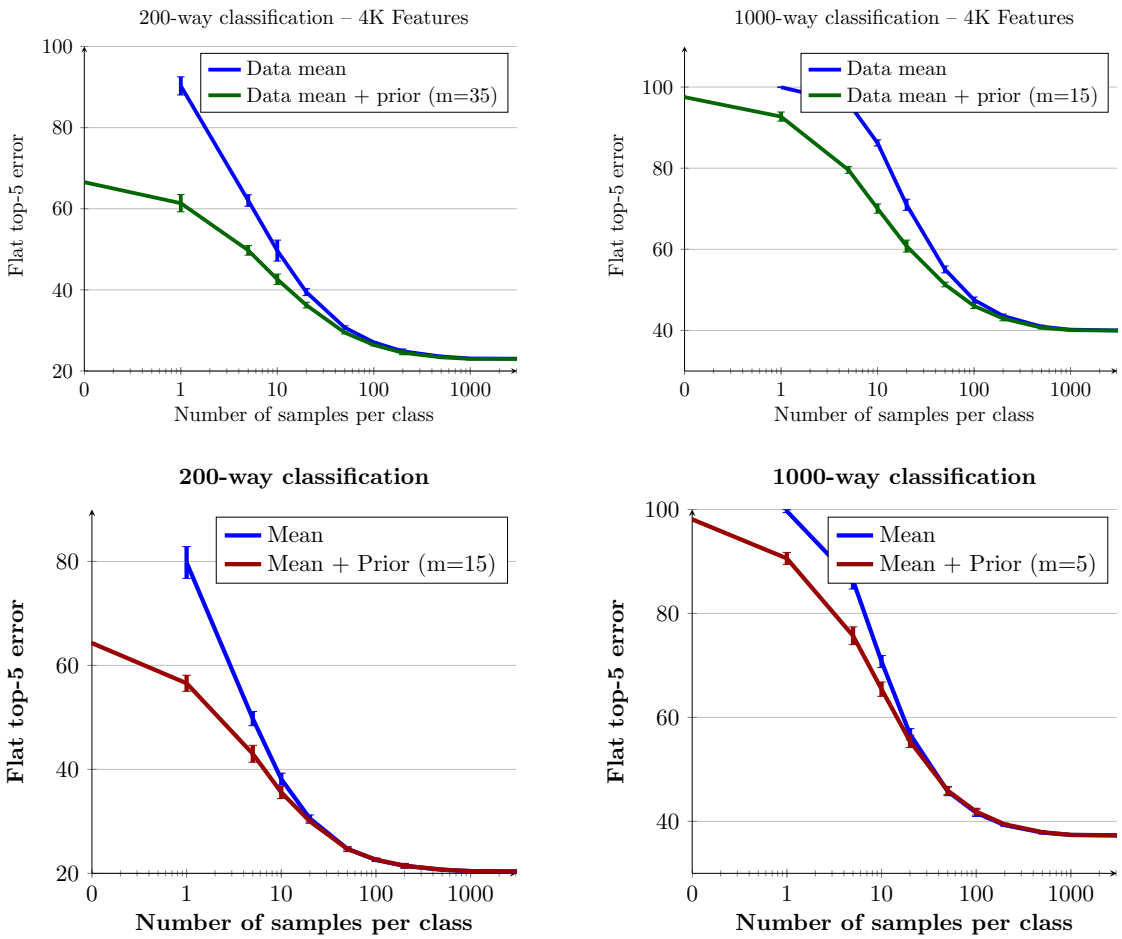


Figure 4: Performance of NCM as a function of the number of images used to compute the means for classes not used during training, with and without the zero-shot prior.

image is used as query to search in the database. The performance is measured by 4×recall@4 averaged over all queries, hence the maximal score is 4. For both datasets we extract the 4K image features also used in our earlier experiments, which are the same ones as those used in (Gordo et al., 2012). To compute the distance between two images, we use the cosine-distance, *i.e.* the dot-product on ℓ_2-normalized vectors.

In analogy to (Gordo et al., 2012) we use the NCM objective to train a metric from the ILSVRC'10 data set, and do early stopping based on retrieval performance. To avoid tuning on the test data the cross-validation is performed on the other dataset, *i.e.* when testing on UKB and we regularize based on Holidays and vice-versa. In Table 6 we compare the performance of the NCM based metric with that of JSCL, and also include a baseline PCA method and performance using the high-dimensional descriptors without any projection. Finally, for the Holidays dataset we included the NMC metric while using early-stopping based on classification performance on the ILSVRC'10 validation data set (NCM-class).

From the results we observe that the NCM metric yields similar performance gains as the JSCL method on both datasets. In both cases a projection to only 128 dimensions yields an equal or better retrieval performance as using the original 4K dimensional features. On the Holidays dataset the NCM metric outperforms the JSCL metric, while on the UKB dataset JSCL slightly outperforms NCM. Both the

INRIA Holidays dataset without projection: 77.4%				
Dim.	PCA	JSCL	NCM	NCM-class
32	61.3	67.7	**69.3**	63.3
64	68.0	73.6	**75.4**	68.8
128	72.3	76.4	**79.6**	73.1
256	75.0	78.3	**80.2**	74.0
512	76.8	78.9	**80.6**	73.5

UKB dataset without projection: 3.19			
Dim.	PCA	JSCL	NCM
32	2.82	3.04	**3.07**
64	3.01	**3.23**	**3.23**
128	3.08	3.31	**3.33**
256	3.15	**3.36**	3.32
512	3.18	**3.36**	3.31

Table 6: Instance level image retrieval results, *left*, on the Holidays dataset (performance in mAP), and *right*, on the UKB dataset (performance is $4\times$ recall@4). NCM metric learning is compared to the PCA baseline and the JSCL metric learning method (Gordo et al., 2012).

NCM and JSCL methods are effective to learn a projection metric for instance level retrieval, employing class level labels, and outperform the unsupervised projection matrix obtained by PCA.

Note that it is curial to use retrieval performance for early stopping; without it (see the NCM-class results) are in fact worse than the original descriptors, and comparable to using PCA. Thus, the classification objective determines a good "path" through the space of projection matrices, yet it is crucial to regularize for retrieval performance. We explain this discrepancy by the fact that instance level retrieval does not require the suppression of the within class variations which is needed for good classification. This observation suggests also that even better metrics maybe learned by training NCM on a large set of queries with corresponding matches.

6 Conclusions

In this paper we have evaluated techniques to learn low-rank, class-independent Mahalanobis distances to support k-NN and NCM classifiers for large scale image classification. We employ high-dimensional dense Fisher vectors that lead to the current state-of-the-art results using our one-vs-rest SVM baseline approach. Both the k-NN and NCM classifiers allow for extensions at (near) zero cost to new classes not used for training, a feature not shared by our SVM baseline, but which is essential for the use on real-life open-ended datasets where new images and classes are continuously added. Surprisingly we found that the NCM classifier outperforms the more flexible k-NN approach. Moreover, using a learned metric, the performance of the NCM classifier is comparable to that of SVM baseline (even better with 4K dimensional features, but somewhat worse using the 64K dimensional features), while projecting the data to as few as 256 dimensions.

We have introduced the non-linear NCMC classifier, an extension of the NCM classifier, that uses multiple centroids to represent a class. Interestingly the used number of centroids offers a complexity trade off: from the linear NCM classifier to the non-linear and non-parametric k-NN classifier in the case when each image in the data set is used as a class centroid. Experimentally we have shown that the NCMC classifier, using 10 centroids per class, significantly improves over the NCM and the k-NN classifiers.

Our learned metrics generalize well to unseen classes, as shown by the experiments where the metric is learned on a subset of 800 classes and evaluated on the 200 held out classes, and further corroborated by our experiments on the ImageNet-10K dataset. For the ImageNet-10K dataset we obtain competitive performance at a negligible cost compared to the feature extraction process: a 8,500 fold speed-up as compared to training 10,000 binary one-vs-rest classifiers. In addition, we have shown that our NCM classifiers can be used in a zero-shot setting where no training images are available for novel classes, and that the zero-shot model significantly boosts performance when combined with a class mean estimated from a limited amount of training images.

Finally we have shown that NCM provides a unified way to treat classification and retrieval problems, since query-by-example image retrieval can be seen as a classification problem where only a single positive sample per class is provided. We have evaluated the NCM metric for image retrieval and found performance that is comparable to the current state-of-the-art on two public benchmarks.

References

B. Bai, J. Weston, D. Grangier, R. Collobert, Y. Qi, K. Sadamasa, O. Chapelle, and K. Weinberger. Learning to rank with (a lot of) word features. *Information Retrieval – Special Issue on Learning to Rank*, 13:291–314, 2010.

S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2011.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010.

J. Chai, H. Liua, B. Chenb, and Z. Baoa. Large margin nearest local mean classifier. *Signal Processing*, 90(1):236–248, 2010.

G. Checkik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 2010.

G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Int. Workshop on Stat. Learning in Computer Vision*, 2004.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

J. Deng, A. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010.

L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. PAMI*, 28(4): 594–611, 2006.

T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, 2011.

J.-L. Gauvain and C.-H. Lee. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Trans. Speech and Audio Processing*, 2(2):291–298, 1994.

A. Gordo, J. Rodríguez, F. Perronnin, and E. Valveny. Leveraging category-level labels for instance-level image retrieval. In *CVPR*, 2012.

D. Grangier and S. Bengio. A discriminative kernel-based model to rank images from text queries. *IEEE Trans. PAMI*, 30(8):1371–1384, 2008.

R. Gray and D. Neuhoff. Quantization. *IEEE Trans. Information Theory*, 44(6):2325–2383, 1998.

M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009a.

M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *ICCV*, 2009b.

H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. PAMI*, 33(1):117–128, 2011.

H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Trans. PAMI*, 2012. to appear.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, 2002.

C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.

H. Larochelle, D. Erhan, and Y. Bengio. Zero-data learning of new tasks. In *AAAI Conference on Artificial Intelligence*, 2008.

Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: Fast feature extraction and SVM training. In *CVPR*, 2011.

D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *ECCV*, 2012.

D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *CVPR*, 2007.

F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010.

F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid. Towards good practice in large-scale learning for image classification. In *CVPR*, 2012.

M. Rohrbach, M. Stark, and B. Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *CVPR*, 2011.

J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

T. Tommasi and B. Caputo. The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories. In *BMVC*, 2009.

C. Veenman and D. Tax. LESS: a model-based classifier for sparse subspaces. *IEEE Trans. PAMI*, 27(9): 1496–1500, 2005.

K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2006.

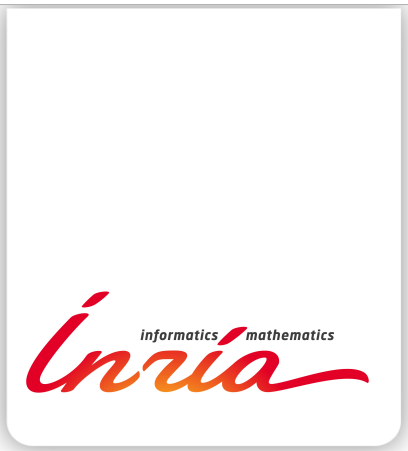
J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the European Symposium on Artificial Neural Networks*, 1999.

J. Weston, S. Bengio, and N. Usunier. WSABIE: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011.

J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *IJCV*, 73(2):213–238, 2007.

X. Zhou, X. Zhang, Z. Yan, S.-F. Chang, M. Hasegawa-Johnson, and T. Huang. Sift-bag kernel for video event analysis. In *ACM Multimedia*, 2008.

Contents



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399