Improving Local Learning for Object Categorization by Exploring the Effects of Ranking

Tien-Lung Chang[1,2] Tyng-Luh Liu[1] Jen-Hui Chuang[2]
[1]Institute of Information Science, Academia Sinica, Taipei 115, Taiwan
liutyng@iis.sinica.edu.tw
[2]Dept. of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan

Abstract

Local learning for classification is useful in dealing with various vision problems. One key factor for such approaches to be effective is to find good neighbors for the learning procedure. In this work, we describe a novel method to rank neighbors by learning a local distance function, and meanwhile to derive the local distance function by focusing on the high-ranked neighbors. The two aspects of considerations can be elegantly coupled through a well-defined objective function, motivated by a supervised ranking method called P-Norm Push. While the local distance functions are learned independently, they can be reshaped altogether so that their values can be directly compared. We apply the proposed method to the Caltech-101 dataset, and demonstrate the use of proper neighbors can improve the performance of classification techniques based on nearest-neighbor selection.

1. Introduction

Supervised learning for classification is an area that incorporates rigorous analysis, practical techniques, and rich applications. When dealing with computer vision tasks, its effectiveness over other approaches is particularly manifest, owing to the inherently complicated nature of the problem itself as well as the data. In this work, our goal is to propose a new technique based on *ranking* that improves *local learning*. To demonstrate the advantage of our method, we apply it to object categorization, and carry out insightful comparisons to other related ones.

In the literature of learning for classification, techniques based on local learning now attract much attention, mostly because a single global model may not fit nowadays data complexity. Among the numerous localized approaches, the nearest-neighbor (NN) technique may be one of the simplest in concept and in practice. While the nearest-neighbor framework is originally proposed as a tool for pattern recognition since its introduction in the 1950's [16], many researchers also adopt it as a classification method and achieve satisfactory results. In particular, there are approaches that directly form a *k-nearest-neighbor classifier*, and focus on organizing training data to improve accuracy and efficiency [19, 25]. The emphasis could also be on learning local distance functions to best separate training samples according to their class label and, for example, use *nearest-neighbor classifier* as a post-procedure [7]. On the other hand, one could try to find certain nearest neighbors first, and then learn a metric or a local classifier [26].

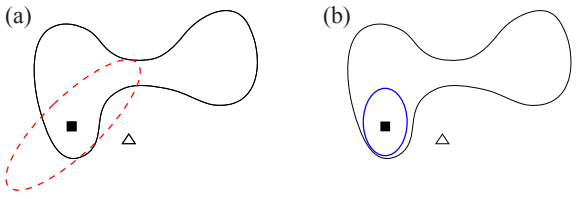


Figure 1. The dashed ellipse in (a) and the ellipse in (b) show level contours of two local distance functions learned at the black square. In (a) the distance function tries to fit more data but it causes a nearest-neighbor search scheme easier to encounter samples that are in a different class.

While the concept of nearest neighbor or k nearest neighbors has been broadly employed, we notice that the term "neighbor" itself is often defined only with a simple distance function such as the L^2 distance (Euclidean distance), or with a learned distance function that is optimized over either the whole data or those within a manually specified region. Therefore, the selected "neighbors" may include some undesired samples, as is illustrated in Figure 1. Our approach toward addressing this problem is closely related to the "P-Norm Push" ranking formulation by Rudin [15]. Specifically, for each labeled sample, the proposed technique learns a local distance function and ranks its neighbors at the same time. As a result, we avoid manually predefining the neighbors of a training sample in learning the respective local distance function, which will turn out to be

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

(a) (b) (c) (d) (e)

Figure 2. Some examples of intraclass variations: Consider the *Wild_Cat* class in Caltech-101. (a) A typical instance of a wild cat for reference. (b) A wild cat in a different pose. (c) A wild cat under a different lighting condition. (d) A drawing of wild cat with some unnatural features. (e) A wild cat example in which over 60% of the image belongs to the background.

more precisely explaining the relationship between the sample and its high-ranked neighbors (whose ranks are given by the distance function itself). These distance functions are learned independently to fit local properties of each sample, and are subsequently reshaped altogether to avoid data overfitting and achieve a unified effect. Classifying a testing sample can then be conveniently done through these distance functions to find neighbors in training data which are very likely in the same class with this testing sample.

Throughout this work we focus on the classification problem of visual object categories, and show that the proposed method can compete with other related techniques as well as the ability to improve them. The paper is organized as follows. In Section 2 we discuss related work, and then explain the P-Norm Push formulation in Section 3. The details of our approach are presented in Section 4, while some experiments and discussions are included in Section 5.

2. Related work

Despite a long history of research on visual object classification and its related problems, some noteworthy rapid progress has been made in recent years, especially on those applications involving numerous object classes. For example, the first reported recognition rate in 2004 on the popular 102-class image dataset Caltech-101 is 17% [6]. In 2006 a number of methods, *e.g.*, [7, 26], have boosted the accuracy rate to near 60%. More recently, the march of recognition rate on this dataset is now around 87% [23].

Among the many efforts on solving this particular application, one important issue is often discussed: the intraclass variations due to the sparsely distributed objects for describing a general concept. In Figure 2, we show some examples of intraclass variations, which are caused by different poses, lighting conditions, representations (painting or photographing), and backgrounds. These variations make the visual object classification a very challenging problem, since it is hard to find an overall applicable criterion to separate objects from different classes. Previous approaches to addressing intraclass variations generally proceed on two fronts: one is to develop a *good* image representation so that it becomes easier to distinguish different classes by simple calculations on the representation, and the other is to learn some sophisticated classifiers by machine learning techniques. Although the emphasis could be on one or the other, current methods mostly have their formulations established by investigating both the two aspects. After all, a more meaningful image representation can often improve the efficiency of a fine classifier, and vice versa.

Representing an image for object categorization appears frequently in the form of a set of patches. (We also adopt the representation.) While such a representation is already flexible, adding various types of image feature to record different properties of a patch have been further proposed to handle issues caused by, say, the changes in an object's shape and pose. The SIFT framework by Lowe [12] is one of the most successful and widely used appearance feature descriptor. In its original version, Lowe first uses a *difference-of-Gaussian* (DoG) function to find scale-space extrema in an image, and then calculates a 3-D histogram over gradient locations and orientations within each patch covering an extremum. Subsequently, Csurka *et al*. [5] describe a *bag-of-keypoints* model, based on the SIFT descriptor, to quantize the collected features into a finite dictionary, and represent each image as a histogram over this dictionary. They then apply the SVM classification method [22] to the new representation to achieve better scores than those from applying SVM to the original image data. While the SIFT feature seems to be effective in object recognition, related studies on feature comparisons [14, 20] have pointed out that SIFT is more conducive in the context of matching than in classification. On the other hand, the shape descriptor *geometric blur* (GB) by Berg and Malik [2] has also attracted much attention lately. Like the SIFT feature, geometric blur captures the gradient information in a patch. The main differences to SIFT are that in geometric blur the gradient information is blurred according to the distance to the patch center, and the information is sampled sparsely. In this work, the geometric blur is adopted as the main feature on patches, and the details will be explained later.

Besides finding a robust feature, there are efforts to correlate the relationships between features to generate a more appropriate image representation, such as the *pyramid rep-*

resentation [9] and the *clique representation* [10]. By feature reweighting, Marszaek and Schmid [13] directly embed the spatial relationship between features into training and testing. In [21] Sudderth *et al*. model the spatial relationship with the *transformed Dirichlet processes*. Alternatively, Wang *et al*. [24] explore the occurrence of feature pairs with the *dependent Hierarchical Dirichlet process*.

From the perspective of learning, localized and adaptive methods stress more on variances between individuals. For example, the *exemplar method* [4] can be considered as to generate multiple classifiers for dealing with different subsets in each class. A *local distance function learning* for classification can be seen in [7]. The work by Lin *et al*. [11] suggests a method for generating *spatially adaptive classifiers* which combine features of different types according to local properties. One notable fact is that the above-mentioned three techniques all need to find some "nearest neighbors": in [4] the testing process is to find the most similar exemplar for a test image; in [7] the authors utilize a k-NN classifier with the learned distance functions in testing; in [11] the testing process is to first find the nearest neighbor in training data for the test image and then apply the corresponding classifier. Furthermore, although in [26] the emphasis is on the speed-up effect by incorporating k-NN into an SVM, the use of k-NN also improves the accuracy in some situations. These observations all indicate the critical role of the "nearest neighbors" in localized methods on object categorization.

3. Neighbor ranking in classification

As stated in Section 1, we are to learn, say, for sample I, a distance function to rank its neighbors for improving object classification. Since a closer sample generally implies a higher probability to be included in a k-NN scheme, one would expect the distance function to be learned is affected more by those samples near I, specified by the distance function *itself*. That is, if we put the samples into an ordered neighbor list according to the measurements by the distance function in an increasing manner, the top portion of the list should be more influential in learning. To this end, we consider P-Norm Push in that the technique inherently tends to pay more attention to the top portion of a ranked list.

3.1. P-Norm Push

The P-Norm Push framework by Rudin [15] is designed for the supervised ranking problem. The method is based on a key observation that in many ranking applications only the top portion of the list is used. For instance, when using a search engine to query information from the Internet, most users only look at the information given in the first few pages. Thus, in [15], the author proposes that a specific price is assigned for each misrank, and the penalties given to the misranks near the top are significantly higher than those given to the misranks towards the bottom. Specifically, consider a set of training samples labeled as $+1$ or -1. Let the set of positive samples be $\{\mathbf{x}_m\}_{m\in\mathcal{M}}$ and the set of negative samples be $\{\mathbf{x}_n\}_{n\in\mathcal{N}}$. The aim now is to find a ranking function $f : \mathbf{x} \in \mathcal{X} \to \mathbb{R}$ from the domain of $\mathbf{x}$, denoted as $\mathcal{X}$, to the set of real numbers. The main difference between ranking and other regression problems is that for an arbitrary sample $\mathbf{x}$, the exact value of $f(\mathbf{x})$ is not important, but the order relation between $f(\mathbf{x}_m)$ and $f(\mathbf{x}_n)$ is, where $\mathbf{x}_m$ and $\mathbf{x}_n$ are a positive-negative sample pair.

To achieve this purpose in P-Norm Push, a $\mathrm{Height}(\cdot)$ function is introduced to each negative sample $\mathbf{x}_n$, and is defined as the number of positive samples that are ranked beneath it. That is,

$$\mathrm{Height}(\mathbf{x}_n) = \sum_{m\in\mathcal{M}} 1_{[f(\mathbf{x}_m)\le f(\mathbf{x}_n)]}\,. \tag{1}$$

The idea of P-Norm Push is to *push* a negative sample with large *height* down from the top. Hence a monotonically and rapidly increasing function from $\mathbb{R}^+$ to $\mathbb{R}^+$, $g(r) = r^p, p \gg 1$, is adopted to give a proper price $g(\mathrm{Height}(\mathbf{x}_n))$ on a negative sample $\mathbf{x}_n$, where p is adjustable for different needs. With (1), we obtain the primal objective functional of P-Norm Push:

$$R(f) = \sum_{n\in\mathcal{N}} g(\mathrm{Height}(\mathbf{x}_n)) \tag{2}$$

$$= \sum_{n\in\mathcal{N}} \left(\sum_{m\in\mathcal{M}} 1_{[f(\mathbf{x}_m)\le f(\mathbf{x}_n)]} \right)^p . \tag{3}$$

More details and discussions about $\mathrm{Height}(\cdot)$ and the price function g can be found in [15].

3.2. Local learning with ranking

To explain how we use P-Norm Push to improve local learning for object categorization, we first need to introduce some notations for the ease of our discussions. We shall denote the ℓth training sample by I_ℓ and the whole training set as $\{I_\ell\}_{\ell\in\mathcal{L}}$. The expression $C(\ell)$ is used to represent the function which extracts the index set such that if $m \in C(\ell)$ then I_m has the same class label of I_ℓ. (We are now dealing with a multiclass classification problem.) The notation D_ℓ denotes the specific distance function learned for I_ℓ, and the distance from I_ℓ to some I_m is represented as $D_\ell(I_m)$. Note that in this setting $D_\ell(I_m)$ is not necessarily equal to $D_m(I_\ell)$, the distance from I_m to I_ℓ.

Consider now a given sample I_ℓ. In our formulation, whenever the term "nearest neighbors" is mentioned, it means that only a few samples that are in some sense close to I_ℓ are considered. In a classification task, these few

neighbors are often assumed to have higher probabilities to be in the same class with I_ℓ. Furthermore, in local learning for classification, if the meanings of the terms "neighbor," "near," and "far" with respect to I_ℓ all come from a learned distance function D_ℓ, we in fact need not to worry about the exact distances from I_ℓ to others, but we do care about the relative magnitude between distances $D_\ell(I_m)$ and $D_\ell(I_n)$ where $m \in C(\ell)$ and $n \notin C(\ell)$. One can easily check that this is indeed very similar to the ranking problem if we also list the samples according to the distances $D_\ell(I_{\ell'})$, $\ell' \in \mathcal{L} \setminus \{\ell\}$. The only difference is that in a ranking problem the samples listed in the top portion are with *higher* values, but here the samples in the top portion of a neighbor list are with *smaller* values. In view of that only a few nearest neighbors are used, we therefore need to make sure that the top portion of the neighbor list is correctly constructed. And this aspect of consideration is identical to the main property in the P-Norm Push framework.

Similar to the formulation of a distance function described by Frome *et al*. [7], we define the distance function D_ℓ for I_ℓ to be a weighted sum of several elementary distance functions $d_{\ell i}$s:

$$D_\ell(I_m) = \sum_{i=1}^{E_\ell} w_{\ell i} d_{\ell i}(I_m), \tag{4}$$

where E_ℓ is the number of elementary distance functions introduced on I_ℓ, and is indeed the number of features detected in I_ℓ. For notation simplicity, we further let $\mathbf{w}_\ell$ be the weight vector $[w_{\ell 1}\, w_{\ell 2}\, \cdots\, w_{\ell E_\ell}]^T$ and $\mathbf{d}_\ell^m$ denote the vector $[d_{\ell 1}(I_m)\, d_{\ell 2}(I_m)\, \cdots\, d_{\ell E_\ell}(I_m)]^T$. Thus equation (4) can be rewritten in an inner product form:

$$D_\ell(I_m) = \mathbf{w}_\ell \cdot \mathbf{d}_\ell^m. \tag{5}$$

Henceforward what we have to learn is a distance function parameterized by $\mathbf{w}_\ell$ to sort samples other than I_ℓ. We now introduce a *cost* on each sample not in the same class with I_ℓ (cf. the Height function defined for negative samples in [15]), and the definition is given by

$$\mathrm{Cost}(I_{n:n\notin C(\ell)}) = \sum_{m\in C(\ell)} 1_{[\mathbf{w}_\ell\cdot\mathbf{d}_\ell^m \geq \mathbf{w}_\ell\cdot\mathbf{d}_\ell^n]}\,. \tag{6}$$

That is, the *cost* of a particular sample $I_{n:n\notin C(\ell)}$ is the number of samples that are in the same class with I_ℓ and located further than I_n according to D_ℓ. A sample $I_{n:n\notin C(\ell)}$ with a large *cost* is expected to be pushed far away from I_ℓ. Here the price function $g(r) = r^p$ in P-Norm Push is again adopted and the objective function to be minimized is

$$F(\mathbf{w}_\ell) = \sum_{n\notin C(\ell)} \left(\sum_{m\in C(\ell)} 1_{[\mathbf{w}_\ell\cdot\mathbf{d}_\ell^m \geq \mathbf{w}_\ell\cdot\mathbf{d}_\ell^n]} \right)^p. \tag{7}$$

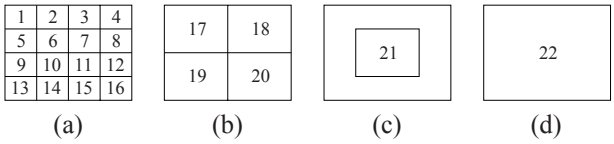


Figure 3. We compute the HSV histograms in 22 overlapping regions in different scales. Each histogram is then normalized by the total number of pixels in the region.

While in [15] Rudin points out that the indicator function in (7) can be replaced by any non-negative, monotonically increasing function which is an upper bound of the indicator function for easier optimization, in our case we still use the original objective in that we can quickly neglect useless $d_{\ell i}$s in the optimization algorithm. The details will be explained in Section 5.1.2.

3.3. Distance function

The D_ℓ in (4) is in a general form. If all samples in an application are of the same size and an elementary distance function $d_{\ell i}(I_m)$ just returns the squared difference between the two respective ith pixels of I_ℓ and I_m, then D_ℓ is simply a *squared Mahalanobis distance* with a diagonal covariance matrix. However, we choose to represent an image as a bag-of-features, and set an elementary distance $d_{\ell i}(I_m)$ to be the smallest distance between the ith feature of I_ℓ and any detected feature of the same type in I_m.

In our experiments we adopt two kinds of features, as they are often used in related work, *e.g.*, [7, 8, 26]. They are the *geometric blur* and *HSV color histogram*. We further apply two different settings to each kind, and obtain four types of features. As in [8], the two types of geometric blur features, termed as GB1 and GB2, are extracted under different scales with radii of 42 and 70 pixels, respectively. The HSV histograms are also extracted under two schemes: one is to compute the histograms on some 84×84-pixel patches (sampled as in GB1 and GB2), the other is to compute the histograms in 22 regions extracted with a pyramid scheme similar to that in [9] (see Figure 3). The two types of HSV histogram are named as HSV1 and HSV2, respectively. We compute an HSV histogram in the same way as in [7] and extract the geometric blur features by modifying the original version[1] in [2]. Notice that for GB1, GB2, and HSV1, the feature-to-set distance is calculated between features of the same type; for HSV2, the feature-to-set distance is from distances between features at the same position.

3.4. Preliminary results

So far we have explained how to learn a distance function that is suitable for ranking neighbors. To justify the formulation, we carry out an experiment for k-nearest-neighbor

[1] http://www.cs.berkeley.edu/∼aberg/demos/gb_demo.tar.gz

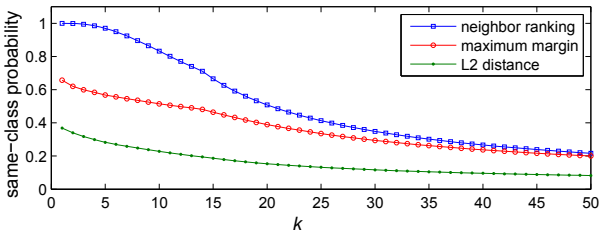
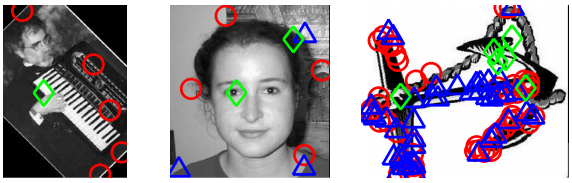


Figure 4. This graph shows the neighbor selection test with three kinds of distances: the blue curve (with square markers) shows the results from our learned distances; the red curve (with circle markers) shows the results from a maximum margin method; the green curve (with dot markers) shows the results from a combination of the L^2 distances defined on geometric blur and color histograms.

selection, and compare our proposed distance function with the raw distance and the distance learned by a *maximum margin* method. The raw distance is from setting the weight $w_{\ell i}$ for a particular sample I_ℓ to 1 if the ith feature is of type GB1 or GB2, and to $\frac{1}{18}$ if the ith feature is of type HSV1 or HSV2. With the maximum margin method, we learn a distance function for a particular I_ℓ via the following constrained optimization:

$$\begin{aligned} \min_{\mathbf{w}_\ell,\xi} \quad & \frac{1}{2}\|\mathbf{w}_\ell\|^2 + \lambda \sum_{m,n} \xi_{mn}\,, \\ \text{s.t.} \quad & \forall\, \ell, m, n,\ \ m \in C(\ell), n \notin C(\ell)\,, \\ & w_{\ell i} \geq 0,\ \ \xi_{mn} \geq 0\,, \\ & \mathbf{w}_\ell \cdot \mathbf{d}_\ell^n - \mathbf{w}_\ell \cdot \mathbf{d}_\ell^m \geq 1 - \xi_{mn}\,. \end{aligned} \tag{8}$$

This optimization is similar to those introduced in [18] and [7]. We follow the technique described in [8] to optimize (8). The experiment is done with the Caltech-101 dataset by randomly selecting 15 images from each class and learning a distance function on each image. The k nearest neighbors of each image are identified according to its respective distance function. For each sample I_ℓ, we calculate the probability the k nearest neighbors fall in $C(\ell)$ (with different values of k). The comparisons between these three distance functions are shown in Figure 4. We can see that our method performs particularly well when k is small, and therefore fulfil the requirement of being capable of selecting a few *good* neighbors. We note that the performance of the maximum margin method may be slightly underestimated since we have not tuned the parameters λ in (8) exhaustively for each image. The other good property of our method is that the learned distance functions are usually sparse. Although in our setting an image I_ℓ may be represented with up to 1222 features, the number of *active* features (*i.e.*, with $w_{\ell i} > 0$) by our algorithm are usually less than 100. Some examples of *active* features are illustrated in Figure 5, where the images have been transformed to gray level for display.



(a) *Accordion* (b) *Face_easy* (c) *Anchor*

Figure 5. Visualizations of *active* features picked by our algorithm. The color information of images is taken off for focusing on feature markers. Each marker represents a location of a patch whose feature is *active*. The different shapes mean different feature types; circle, diamond, and triangle represent GB1, GB2, and HSV1, respectively. The HSV2 feature is omitted here. In our experiments, a harder (with lower recognition rate) image like (c) often requires more *active* features, while an easier one needs only several important features, *e.g.*, the shoulder curve in *Faces_easy* class and the boundary edge in the *Accordion* class.

To further investigate the effectiveness of our neighbor ranking method, we also develop a simple voting scheme for classification. Specifically, for each learned local distance function D_ℓ, $\ell \in \mathcal{L}$, we define a threshold

$$\theta_\ell = \frac{1}{2}(D_\ell(I_m) + D_\ell(I_n)), \tag{9}$$

where, according to the neighbor list of I_ℓ, I_n is the first sample not in the same class with I_ℓ and I_m is the one ranked right above I_n. In testing with a new image I, each distance function with its associated threshold behaves as a classifier. That is, if $D_\ell(I) \leq \theta_\ell$, then D_ℓ supports that I is in the same class with I_ℓ. The final class label of I is voted by all classifiers. Images without class label assigned are classified as background. We test this primitive method on the Caltech-101 dataset by following the setting in Berg *et al*. [1]. Namely, for each class we randomly pick 15 images for training and another 15 for testing, and then switching their roles in the second round. The results are presented in the first column of Table 1. Even with such a rough rule, the outcomes are already comparable with some of the recent novel techniques, *e.g.*, [7, 9, 11, 24].

4. Reshape distance functions for nearest neighbor classification

The distance functions in Section 3 are learned in an independent manner, and are hardly compared with each other. Although a learned distance function alone performs well in selecting the neighbors, it is not feasible to decide if training sample I_ℓ or $I_{\ell'}$ should be placed in a higher position on the neighbor list of a new test sample I by comparing the values of $D_\ell(I)$ and $D_{\ell'}(I)$. In fact, these distance functions act more like rankers, and it is not necessary to keep the exact values with them. Considering that given a

strictly increasing function q, transforming a distance function form $D_\ell(\cdot)$ to $\widetilde{D}_\ell(\cdot) = q(D_\ell(\cdot))$ would not affect the objective function (7). We can utilize such a function to *reshape* these learned local distance functions altogether so that direct comparisons among them can be achieved.

The reshape process is carried out as *competitions* over training samples between any two distance functions of different classes. Given a reference sample I_ℓ, we prefer that if I_m is in the same class with I_ℓ and I_n is not, then the reshaped distance $\widetilde{D}_m(I_\ell)$ should be smaller than $\widetilde{D}_n(I_\ell)$. That is, the competition between $\widetilde{D}_m$ and $\widetilde{D}_n$ for I_ℓ depends on the class labels of I_ℓ, I_m and I_n. Furthermore, suppose we make a sorted list of reshaped distances (in increasing order) $\{\widetilde{D}_{\ell'}(I_\ell)\}_{\ell' \neq \ell}$ from the training data and the learned distance functions in Section 3. It is reasonable to pay more attention to the samples in the top portion, since in testing a given I, we usually consider those training samples with smaller distances to I (under the assumption that I is related to some training image(s) I_ℓ). Hence for each fixed reference sample I_ℓ, we can define a *cost* to each reshaped $\widetilde{D}_n(I_\ell), n \notin C(\ell)$ by

$$\text{Cost}(\widetilde{D}_n(I_\ell)) = \sum_{m \in C(\ell)} 1_{[\widetilde{D}_m(I_\ell) \geq \widetilde{D}_n(I_\ell)]}. \tag{10}$$

Although any strictly increasing function could be our reshape function, in this work we restrict it to be a scaling function, $q(r) = ar$, where $a \in R^+$. Now the reshaped function $\widetilde{D}_\ell$ can be parameterized by a single scaler a_ℓ as $\widetilde{D}_\ell = a_\ell D_\ell$ and the overall objective function to be minimized is

$$\widetilde{F}(\mathbf{a}) = \sum_{\ell \in \mathcal{L}} \sum_{n \notin C(\ell)} \left(\sum_{m \in C(\ell)} 1_{[a_m D_m(I_\ell) \geq a_n D_n(I_\ell)]} \right)^{p'}, \tag{11}$$

where $\mathbf{a} = [a_1, a_2, ..., a_{|\mathcal{L}|}]^T$, and $p' \gg 1$ acts as the price function parameter like the p in (7).

5. Experiments

In this section we discuss some implementation details and the results derived by the proposed neighbor ranking method. We again consider the Caltech-101 dataset, collected by Fei-Fei *et al*. [6]. In all our experiments, from each class, we randomly pick 15 images for training and another 15 images for testing. We then exchange their roles and calculate the average recognition rate. Images with larger sizes are scaled down to around 60000 pixels while preserving the aspect ratios. For each image, we extract the four types of features as described in Section 3.3, and sample at most 400 features respectively for each type of GB1, GB2, and HSV1. Hence an image is represented as a bag with at most $1200 + 22$ features. Since the Caltech-101 dataset contains 101 object classes and a background class, we thus obtain 1530 distance functions in the training procedure.

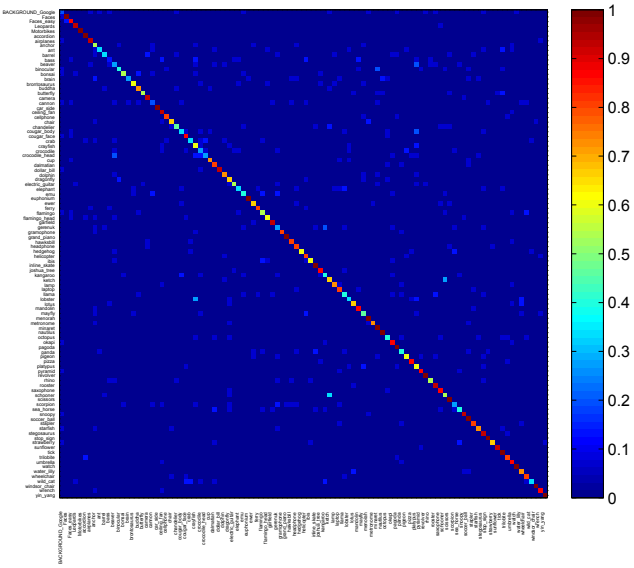


Figure 6. Confusion table by 3-NN+neighbor ranking+reshaping.

After the reshaping, we use a modified 3-NN classifier to assign class labels on test data. Specifically, given a test image I, the classifier first arranges the order of the training images according to their reshaped distances to I. If a prediction on the class label of I cannot be determined by the top three, then the succeeding samples will be considered one at a time until a decision can be made. We have tested this method in several settings with different features. The resulting confusion table is plotted in Figure 6, and the outcomes are listed in the second column of Table 1. Despite the use of a simple classifier and basically two kinds of features, the accuracy rate by our method achieves $69.83 \pm 0.41\%$, which is better than those reported in all previous work on the same setting. However, it is still behind the 87% accuracy rate by Varma and Ray [23], in which they exploit several other robust features.

We also apply the neighbor ranking to the SVM-KNN framework [26]. Given a test image I, the reshaped distances from each training sample to it are calculated first, and then the k nearest neighbors are determined to train a multiclass SVM classifier for I. (We use *libSVM* [3] for the implementation here.) The overall recognition rate in this case can be improved to $71.8 \pm 0.32\%$. Table 1 shows the related results.

5.1. Details and discussions

5.1.1 Price function

The parameters p in (7) and p' in (11) are the price function parameters in the P-Norm Push, and some discussions

	Voting	k-NN3 ($k=3$)	SVM-KNN50 ($k=50$)	SVM-KNN7 ($k=7$)
GB1+GB2	59.75±1.55	67.87±0.42 (67.97±0.41)	70.05±0.31	68.94±0.72
GB1+GB2+HSV1	60.88±1.13	69.64±0.60 (68.43±0.33)	70.26±0.46	70.39±0.85
GB1+GB2+HSV1+HSV2	62.19±0.23	69.83±0.41 (69.15±0.25)	71.80±0.32	71.76±0.93

Table 1. Accuracy rates on Caltech-101 dataset with different features and methods. **Voting:** The rough method stated in Section 3.4. **k-NN3:** The 3-NN classifier by a reshaped distance. **SVM-KNN50:** SVM-KNN in which the $k=50$ nearest neighbors are selected by a reshaped distance. **SVM-KNN7**: The same with SVM-KNN50, but now $k=7$ and the kernel used in the SVM is now derived from our reshaped distance. The scores in the brackets are resulted from the speed-up algorithm stated in Section 5.1.3. Notice that in SVM-KNN50 we focus on the boosting ability of the neighbor selection scheme so that in SVM the kernel is generated by all four types of features.

for different choices can be seen in [15]. While the values can be adjusted according to different needs, here we show a way to adjust p for depressing the highest *cost* with a higher priority. Suppose the training data includes N_c samples from each class, and totally has N samples. For convenience, let $N_o = N - N_c$. Thus the possible maximum value of the objective function (7) is $N_o \times N_c^p$. Let J (J') be the price jump of a strategy involving (not involving) lowering the highest cost h. Thus $J \geq h^p - (h-1)^p$ and $J' \leq N_o(h-1)^p$. To make sure the algorithm will choose to depress the highest cost, we can choose the value p such that $h^p - (h-1)^p > N_o(h-1)^p$. It follows that

$$p > \frac{\ln(N_o+1)}{\ln(\frac{h}{h-1})} \geq \frac{\ln(N_o+1)}{\ln(\frac{N_c}{N_c-1})} \qquad (12)$$

where $h \in \{2, ..., N_c\}$. The lower bound constraint for p is derived for the aforementioned scheme. In practice, since most applications consider only a few nearest samples, a smaller p value can be analogously selected, while achieving satisfactory results.

5.1.2 Optimizing $\mathbf{w}_i$ and $\mathbf{a}$

To optimize the objective function (7) according to $\mathbf{w}_\ell$, we choose to find the best $w_{\ell i}$ in a sequential manner. The process is repeated until convergence. If the value of $w_{\ell i}$ remains zero in three consecutive iterations, it will be marked as *inactive*. This way we can detect redundant or unnecessary features especially in the early stages. Since scaling $\mathbf{w}_\ell$ by a positive number would not affect the objective, $\mathbf{w}_\ell$ is normalized by the sum of its components after each iteration for computational stability.

Optimizing (11) with respect to $\mathbf{a}$ can be done in a similar fashion. The only exception is that the component a_ℓ of $\mathbf{a}$ would never be set to zero. This property can be guaranteed since (11) is also a piecewise constant function. For computational efficiency, we do not optimize (11) over all triples (ℓ, m, n). We only consider those satisfying the following conditions: $I_m \in C(\ell)$, $I_n \notin C(\ell)$, and I_ℓ is one of the k nearest neighbors of both I_m and I_n (computed respectively by D_m and D_n). This way works because that the top portion of each neighbor list is more meaningful after the optimization on $w_{\ell i}$ in the previous step.

5.1.3 Speeding up

We also implement a speed-up version to eliminate even more useless features earlier. This is done by pre-selecting initial features. Why this is possible is that in optimizing (7) we find that only a relatively small number of features are *active* in the final stage. To pre-select features we first sort all features of an image I_ℓ according to the ranking ability of each feature i: $\sum_{n \notin C(\ell)} \left(\sum_{m \in C(\ell)} 1_{[d_{\ell i}(I_m) \geq d_{\ell i}(I_n)]} \right)^p$ and sample relatively densely in the top portion while sparsely in the lower portion. To only select features with high ranking ability is not practical since less powerful features may complement robust features. We have tested our method with the sampling ratios $\frac{1}{3}$, $\frac{1}{5}$, and $\frac{1}{8}$, on the top, middle, and bottom portions respectively to get competing results in about $\frac{1}{5}$ training time.

5.1.4 Distances and SVM

Defining the kernel in SVM with a distance function can be done in a straightforward way via the *kernel trick* formula [17, 25]: $K(x,y) = \langle x, y \rangle = \frac{1}{2}(\langle x, x \rangle + \langle y, y \rangle - \langle x - y, x - y \rangle) = \frac{1}{2}(D^2(x,0) + D^2(y,0) - D^2(x,y))$. However in our work the distance function is asymmetric so that we first need to resolve this issue. We present two approaches based on the use of different distances in testing. The first one (denoted as SVM-KNN50, see Table 1) is to select the k nearest neighbors with the learned distance functions but uses the raw distance (described in Section 3.4) in SVM. To make sure the kernel is positive-definite we follow [26] to let $D(I_\ell, I_m)$ be defined as $\frac{1}{2}(D_\ell(I_m) + D_m(I_\ell))$, where D_ℓ and D_m are the raw distances. The other (denoted as SVM-KNN7) is to use the learned distance functions both in selecting the k nearest neighbors and in SVM. For this scheme we need a further definition on the distance from a test image I_t to a training image I_ℓ:

$$\widetilde{D}_t(I_\ell) := \frac{\sum_{m \in \mathcal{B}(I_t)} (\widetilde{D}_m(I_t) \times \widetilde{D}_m(I_\ell))}{\sum_{m \in \mathcal{B}(I_t)} \widetilde{D}_m(I_t)}, \qquad (13)$$

where $\mathcal{B}(I_t)$ includes the 3 nearest neighbors of I_t in the training data.

We obtain similar outcomes with these two settings in very different k values. With the first setting we get the best result when k is around 50 and the score does not change much as k varies in $30 \sim 60$, while with the second we set $k = 7$ to achieve the best score. The later result seems that the classification is dominated by the neighbor selection. This is probably due to the crudely produced kernel from the more asymmetric distance functions.

6. Conclusion

We have presented a novel approach to improving local learning by incorporating supervised neighbor ranking in distance function learning. The effectiveness of the proposed technique is demonstrated by dealing with a challenging multiclass classification problem, visual object categorization. We show that together with a simple k-nearest-neighbor classifier, our method can yield satisfactory results, as well as has the ability to improve some existing localized learning methods, *e.g.*, SVM-KNN [26]. The classification rates by the simple settings described in our experiments can compete with those in most related work, except that by Varma and Ray [23], in which several robust features and a more sophisticated learning scheme are used. As our framework has the flexibility in easily adopting more features in the distance function, we would explore the effects of adopting those features used in [23] for our future work.

Acknowledgements

This work is supported in part by grants 95-2221-E-001-031, 96-2221-E-001-021 and 96-EC-17-A-02-S1-032.

References

[1] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, pages I: 26–33, 2005. 5

[2] A. Berg and J. Malik. Geometric blur and template matching. In *CVPR*, pages I:607–614, 2001. 2, 4

[3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 6

[4] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *CVPR*, pages 1–8, 2007. 3

[5] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV*, 2004. 2

[6] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples an incremental bayesian approach tested on 101 object categories. In *Proceedings of the Workshop on Generative-Model Based Vision*, Washington, DC, June 2004. 2, 6

[7] A. Frome, Y. Singer, and J. Malik. Image retrieval and classification using local distance functions. In *NIPS*, pages 417–424. MIT Press, Cambridge, MA, 2007. 1, 2, 3, 4, 5

[8] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007. 4, 5

[9] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages II: 2169–2178, 2006. 3, 4, 5

[10] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *CVPR*, pages 1–8, 2007. 3

[11] Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh. Local ensemble kernel learning for object category recognition. In *CVPR*, pages 1–8, 2007. 3, 5

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

[13] M. Marszaek and C. Schmid. Spatial weighting for bag-of-features. In *CVPR*, pages II: 2118–2125, 2006. 3

[14] K. Mikolajczyk, B. Leibe, and B. Schiele. Local features for object class recognition. In *ICCV*, pages II: 1792–1799, 2005. 2

[15] C. Rudin. Ranking with a p-norm push. In *COLT*, pages 589–604, 2006. 1, 3, 4, 7

[16] S. Salzberg, A. Delcher, D. Heath, and S. Kasif. Best-case results for nearest-neighbor learning. *PAMI*, 17(6):599–608, June 1995. 1

[17] B. Schölkopf. The kernel trick for distances. In *NIPS*, pages 301–307, 2000. 7

[18] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *NIPS*, Cambridge, MA, 2004. MIT Press. 5

[19] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, pages 750–757, 2003. 1

[20] M. Stark and B. Schiele. How good are local features for classes of geometric objects. In *ICCV*, October 2007. 2

[21] E. B. Sudderth, A. B. Torralba, W. T. Freeman, and A. S. Willsky. Describing visual scenes using transformed dirichlet processes. In *NIPS*, 2005. 3

[22] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998. 2

[23] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, October 2007. 2, 6, 8

[24] G. Wang, Y. Zhang, and F. Li. Using dependent regions for object categorization in a generative framework. In *CVPR*, pages II: 1597–1604, 2006. 3, 5

[25] G.-J. Zhang, J.-X. Du, D.-S. Huang, T.-M. Lok, and M. R. Lyu. Adaptive nearest neighbor classifier based on supervised ellipsoid clustering. In *FSKD*, volume 4223 of *Lecture Notes in Computer Science*, pages 582–585. Springer, 2006. 1, 7

[26] H. Zhang, A. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, pages II: 2126–2136, 2006. 1, 2, 3, 4, 6, 7, 8