A Latent Variable Ranking Model for Content-based Retrieval

Ariadna Quattoni[1], Xavier Carreras[1], and Antonio Torralba[2]

[1] Universitat Politècnica de Catalunya, Dept. LSI, 08034 Barcelona ⋆
[2] Massachusetts Institute of Technology, CSAIL, Cambridge MA 02139
{aquattoni,carreras}@lsi.upc.edu torralba@csail.mit.edu

Abstract. Since their introduction, ranking SVM models [11] have become a powerful tool for training content-based retrieval systems. All we need for training a model are retrieval examples in the form of triplet constraints, i.e. examples specifying that relative to some query, a database item a should be ranked higher than database item b. These types of constraints could be obtained from feedback of users of the retrieval system. Most previous ranking models learn either a global combination of elementary similarity functions or a combination defined with respect to a single database item. Instead, we propose a "coarse to fine" ranking model where given a query we first compute a distribution over "coarse" classes and then use the linear combination that has been optimized for queries of that class. These coarse classes are hidden and need to be induced by the training algorithm. We propose a latent variable ranking model that induces both the latent classes and the weights of the linear combination for each class from ranking triplets. Our experiments over two large image datasets and a text retrieval dataset show the advantages of our model over learning a global combination as well as a combination for each test point (i.e. transductive setting). Furthermore, compared to the transductive approach our model has a clear computational advantages since it does not need to be retrained for each test query.

1 Introduction

In content-based retrieval the task is to retrieve items in a database that are most relevant to a given query. What differentiates content-based retrieval from other retrieval scenarios is that the query is itself an item. For example, in content-based image retrieval the query would be an image. In this paper, we consider a retrieval framework where the relevance scoring of items is computed using a pair-wise similarity or relevance function that measures how relevant a database item is for a given query. Since items can be complex objects such as documents, images, or videos, defining a priori an appropiate relevance function can be challenging, because the semantics of objects is unknown. Therefore, there is an active line of reseach that pursues learning relevance functions using some form of user supervision.

Since their introduction, ranking SVMs [11] have become a powerful tool for optimizing the similarity function of such content-based retrieval systems. All we need

⋆ Supported by the Spanish Ministry of Science and Innovation (A.Q. JCI-2009-04240; X.C. RYC-2008-02223) and the EU PASCAL2 Network of Excellence (FP7-ICT-216886).

for training are examples in the form of triplet constraints. Each triplet specifies that a database item a is more similar to a query than another database item b. These types of constraints could be obtained from feedback of users of the retrieval system. In the standard ranking SVMs [18, 1, 9, 8, 5], one assumes that a set of elementary pair-wise similarity functions is given and uses the triplets to learn an optimal weigthed combination of these functions. In this global ranking model the same weighted combination is used for all queries independently of where they lie in the query space.

However, when we search for the most relevant items for a query, the importance of a particular feature or similarity function might vary depending on the type of query. Consider for example a content-based image retrieval setting, if the query is an image of a natural scene, color might be important but it might be less important for an indoor scene. Local learning algorithms [4, 23, 9, 8, 20] attempt to address this problem by adjusting the parameters of the model to the properties of the training set in different areas of the input space. In the transductive setting, the simplest local algorithm proceeds in two steps. First, for a given test sample we find a set of nearest neighbors among the available training samples. And second, a model is learned using only these neighbors as training data. The same local learning idea can be applied to learn a similarity function from ranking constraints. In this case, for each test query we would learn a local combination of elementary similarity functions using only ranking constraints from the training queries in its vicinity. However, this approach has two potential limitations. First, it is sensitive to the way in which vicinity is defined. Put it differently, the local approach might fail if neighbors in the feature space used for representation do not correlate with the ground-truth similarity. Second, retraining a model for each test query might be prohibitive for certatin applications that demand fast retrieval. Another approach to learn local models is to train a linear combination of elementary similarities for each item in the database [9, 8]. One of the limitations of modeling the problem in this way is that the number of parameters grows linearly with the number of database items, making generalization challenging.

In this paper we propose a different *coarse to fine* strategy, aimed at addressing the limitations of previous methods while still taking advantage of local learning. The idea is, given a query, to first compute a distribution over *coarse* classes and then use a weighted combination that has been optimized for the queries of that class. A main challenge in this strategy is that coarse classes are hidden and need to be induced by the learning algorithm. We propose a latent variable ranking model that learns both the hidden classes and the weights of the linear combination for each class, using ranking triplets. Our contribution is a new latent variable ranking model for inducing similarity functions, which also learns the *vicinity* function that minimizes the global ranking loss. Learning a function that maps queries to latent classes implies that at test time we only need to evaluate the latent function and compute the corresponding similarity, without any training required. We performed experiments over two large image datasets and a large textual dataset and show that our model outperforms state-of-the-art approaches for the task. In our experiments only a few latent classes were sufficient to see some of the benefits of local learning. Some of these classes seem to correspond to the *natural* coarse variability of the dataset (e.g. indoor vs. outdoor scenes in image collections).

2 Related Work

In this section we review the most relavant related work on learning from ranking constraints as well as relevant work on local learning.

One approach to inducing ranking functions from triplet constraint consists of learning a bilinear similarity measure of the form: $s(x_q, x_r) = x_q^\top Z x_r$. Chechik *et al.* [5] presented an online dual algorithm to find an optimal Z using ranking constraints. The loss function used in [5] is a generalization of the hinge loss for the ranking setting, which is the same loss function used in this paper. Finally, Lanckriet *et al.* [13] proposed a method to learn a global linear combination of predefined kernel functions.

Ranking constraints are also used in the context of semi-supervised clustering. In this setting the goal is to leverage the constraints to improve the quality of the partition generated by the clustering algorithm [21, 2, 12, 3]. Hertz *et al.* [10] developed a similarity learning method based on boosting that used a semi-supervised constrained ranking algorithm to train the gaussian mixture weak learners.

Local learning algorithms have had a long history in machine learning. Bottou and Vapnik [4] provided both an empirical and theorethical study of a transductive classification algorithm based on training a neural network for each test sample using only training samples from its vicinity. His empirical results showed that local learning could greatly improve the performance of an optical character recognizer. More recently, Zhang *et al.* [23] proposed a local model for object recognition where a support vector machine is trained for each test sample using only nearest neigboor training samples. For pose estimation, Urtasun and Darrell [20] proposed a local learning algorithm where a mixture of gaussian processes is learnt for each test image using training examples in its vicinity. Frome *et al.* [9, 8] proposed an alternative local similarity learning algorithm where the similarity between a focal database image x_f and a query image x_q is assumed to be a linear combination of elementary similarity functions: $D(x_q, x_f) = \sum_l w_{f,l} d_l(x_f, x_q)$. The algorithm learns the set of weights $w_{f,l}$ for each focal image by minimizing a hinge loss over the ranking constraints. The main difference between our approach and previous local learning methods for learning similarity functions is that while previous approaches implicitly induced a partition of the data into local neighborhoods, we learn explicitly the soft *vicinity* function that minimizes a global ranking loss. In other words, the function that assigns latent classes to queries can be regarded as inducing a soft partition over the data points. This approach has the advantage that we can directly control the number of partitions and parameters, as opposed to having a set of parameters for each database item. Moreover our approach does not require to retrain the model at test time, as opposed to the transductive approach.

Yan and Hauptmann [22] proposed a probabilistic latent-variable model for query retrieval. As in our case, their model employs a mixture model for the query space, but in their case the combination of elementary rankers is modeled using a regression function. For learning, they optimize a likelihood function. The main difference with our work is that we use a max-margin approach to ranking with latent query classes, and optimize a pairwise ranking loss function. Our formulation results in a simple and efficient subgradient algorithm for parameter estimation.

3 Ranking Models

We assume a set of training queries indexed by $\mathcal{Q}$, where each index $q \in \mathcal{Q}$ is associated with a query $\mathbf{x}_q \in \mathcal{X}$. Similarly we assume a database indexed by $\mathcal{R}$, where each index $r \in \mathcal{R}$ is associated with a database item $\mathbf{x}_r \in \mathcal{X}'$. The representation of database items needs not to be the same as that of the queries. Instead, our method requires a set of m elementary similarity functions that take a query and a database item and compute a score. Thus, we assume vectors $\mathbf{k}_{q,r} \in \mathbb{R}^m$ where every component $k^j_{q,r}$ indicates the score of the j-th similarity function between query q and database item r.

Given a query, our goal is to rank the database items with respect to their similary to the query. Consider a simple similarity function that consists of a linear combination of elementary similarity functions, $\mathrm{sim}(q,r) = \mathbf{z}^\top \mathbf{k}_{q,r}$, where $\mathbf{z} \in \mathbb{R}^m$ are the weights of the function. In this paper we only consider positive similarity combinations and therefore $z^j \geq 0$ for all j.

Given supervised training data, the weights $\mathbf{z}$ of the linear combination could be optimized using an appropriate objective function. For example, assume we have a set $\mathcal{C}$ of training constraints. Each element of $\mathcal{C}$ is a triple of the form $\langle q, a, b \rangle$, indicating that a query $q \in \mathcal{Q}$ is more similar to item $a \in \mathcal{R}$ than to item $b \in \mathcal{R}$. We can define the loss of a sim function on a training set $\mathcal{C}$ as a natural extension of the classification hinge loss for ranking:

$$\mathrm{L}(\mathcal{C}) = \sum_{\langle q,a,b \rangle \in \mathcal{C}} \max\{\, 0 \,,\ \mathrm{sim}(q,b) - \mathrm{sim}(q,a) + 1 \,\} \tag{1}$$

Thus when the similarity function satisfies a ranking constraint with margin 1 we pay no loss for that constraint; otherwise we pay a loss proportional to how much the ranking constraint is violated. Using this loss function we could set the weights $\mathbf{z}$ to minimize the following regularized objective: $\mathrm{L}(\mathcal{C}) + \frac{\lambda}{2}\|\mathbf{z}\|^2$, where λ is the regularization parameter. This corresponds to a ranking SVM [11]. One simple strategy to minimize this objective is to use a primal sub-gradient method [19], which is the approach we use in this paper.

3.1 Ranking Models with Latent Variables

In the approach described above the same linear combination of elementary similarity functions is used for all queries. However, the importance given to each similarity function in the combination might vary depending on the query. In our ranking model we will address this by introducing latent classes. The idea behind this approach is to first assign a query to a class and then use a similarity function that specializes on ranking queries for that class.

Consider that for a given query q there is a distribution over G latent classes: $p(h_q = g \mid q)$, where $h_q = g$ indicates that g is the latent class of query q. We can then define the similarity function:

$$\mathrm{sim}(q,r) = \sum_{g=1}^{G} p(h_q = g \mid q)\ \mathbf{z}_g^\top \mathbf{k}_{q,r} \quad , \tag{2}$$

where the distribution over latent classes is given by a log-linear model,

$$p(h_q = g \mid q) = \frac{\exp^{\mathbf{w}_g^\top \mathbf{x}_q}}{\sum_{g'=1}^{G} \exp^{\mathbf{w}_{g'}^\top \mathbf{x}_q}} \quad . \tag{3}$$

With these definitions our similarity function is fully determined by two parameter matrices: $Z = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_G]$, where each column $\mathbf{z}_g$ defines the combination of elementary similarities for latent class g; and $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_G]$ that defines the conditional distribution of latent classes given a query, using a parameter vector $\mathbf{w}_g$ for each class. To learn the parameters of this function, we can optimize the objective

$$\mathrm{L}(\mathcal{C}) + \frac{\lambda_{\mathrm{w}}}{2}\|W\|^2 + \frac{\lambda_{\mathrm{z}}}{2}\|Z\|^2 \quad , \tag{4}$$

where λ_{w} and λ_{z} control the regularization of the two sets of parameters.

3.2 Training the parameters

In this section we describe how to estimate the model parameters W and Z that minimize the objective function of Eq. (4). We use an alternating optimization strategy, which consists of a series of iterations that optimize one set of parameters given fixed values for the others. We start with some initial distribution of latent classes for each training query. Then we iterate between fixing W and minimizing the objective with respect to Z, and fixing Z and optimizing with respect to W. We use a gradient-based method for each optimization step. Since our objective is non-differentiable because of the hinge loss, we use a sub-gradient of the objective, which we compute as follows. Given parameter values, let Δ be the set of ranking constraints in $\mathcal{C}$ that attain non-zero loss,

$$\Delta = \{ \langle q, a, b \rangle \in \mathcal{C} \mid \mathrm{sim}(q, a) - \mathrm{sim}(q, b) < 1 \} \quad . \tag{5}$$

Notice that

$$\mathrm{sim}(q, b) - \mathrm{sim}(q, a) = \sum_{g=1}^{G} p(h_q = g \mid q)\, \mathbf{z}_g^\top \, (\mathbf{k}_{q,b} - \mathbf{k}_{q,a}) \quad . \tag{6}$$

The subgradient of the loss function with respect to parameter $Z_{g,j}$ is

$$\frac{\partial \mathrm{L}}{\partial Z_{g,j}} = \sum_{\langle q,a,b \rangle \in \Delta} p(h_q = g \mid q)(k_{q,b}^j - k_{q,a}^j) \quad , \tag{7}$$

where the sum is over all constraints that have non-zero loss under the current parameter settings. In the expression, the contribution of the j-th elementary similarity function to parameter $Z_{g,j}$ is weighted by the probability that queries attaining some loss are in class g. Thus, the updates to parameters of class g will be dominated by queries belonging to the class.

When parameters Z are fixed, it is useful to define the following quantities,

$$\epsilon_{q,g} = \sum_{a,b \text{ s.t. } \langle q,a,b \rangle \in \Delta} \mathbf{z}_g^\top (\mathbf{k}_{q,b} - \mathbf{k}_{q,a}) \quad . \tag{8}$$

Intuitively, $\epsilon_{q,g}$ is the contribution to the loss made by class g on the constraints of query q. More negative values indicate better ranking performance for that query under class g. Clearly, under fixed Z the values ϵ are constant. The subgradient of the loss function with respect to $W_{g,j}$ is

$$\frac{\partial \mathrm{L}}{\partial W_{g,j}} = \sum_{q \in \mathcal{Q}} \epsilon_{q,g} \ \left[p(h_q = g \,|\, q) - p(h_q = g \,|\, q)^2 \right] \ x_q^j \quad . \tag{9}$$

For negative values of $\epsilon_{q,g}$ the update will increase the probability that query q belongs to class g, and viceversa. Thus the algorithm will assign queries to latent classes that perform well on that query.

4 Experiments

In this section we present experiments on learning ranking functions using image and textual datasets. The main goal of the experiments is to show that adding latent variables to a state-of-the-art ranking SVM [11] results in clear improvements in three different datasets. In addition we also compare our method to an SVM trained locally for each test query (e.g., [23] used this approach for a classification task), which is a simple but expensive approach to specialize the ranking function to a query.

Obtaining ground truth for training and testing retrieval algorithms is a challenging task. In the absence of feedback from real users of a retrieval system, alternative approaches have been proposed to obtain ground truth similarity data (e.g., [5]). In this paper we use semantic annotations available in the datasets to generate ground truth similarities, as will be described next. These similarities are then used to automatically generate training constraints. Ideally, however, the constraints used for learning would be generated from user interactions with a retrieval system.

4.1 Datasets and Ground-Truth Similarities

The *SUN* dataset [6] contains 12,000 annotated images covering a large range of indoor and outdoors scenes. More than 200 object categories have been manually annotated using LabelMe [17] by a single annotator and verified for consistency. In total the dataset contains 152,000 annotated object instances. Images in this dataset were grouped into five random partitions: 2,000 images were used as training queries, 1,000 as validation queries, 2,000 as test queries, 6,000 as database images, and finally 1,000 images were reserved as novel-database images. To generate *ground-truth similarities* for training and testing with this dataset we derived a similarity score from the manual annotations. More precisely, for every image we computed a spatial histogram of object ocurrences using the ground-truth segmentations and corresponding object tags. From this we generated a pairwise similarity matrix S, where $S_{q,r}$ is the intersection between the histogram of query q and the histogram of database image r. Figure 3 shows examples of the top ground-truth neighbors obtained by this process. The similarity function is not perfect but seems to be a reasonable aproximation to the true semantic similarity between scenes.

The *REUTERS-IMG* dataset [16] consists of 25,538 images collected from the Reuters News website. Each image in this database is associated with a caption, which we used to compute image similarities. This dataset contains images from a wide set of topics such as sports, entertainment and politics. We randomly selected 17,000 images used as the database; and three disjoint sets of 500 images used as train, validation and test queries. The *ground truth similarities* for each query image were generated using the image captions. For each image, we considered only the content words of the caption (i.e., we disregard stop words). Then, the similarity between an image query and a database image was set to the number of common content words in their captions.

The *RCV1* dataset [14] is a publicy available benchmark collection of textual documents, where documents are categorized by topics according to a topic hierarchy. We randomly selected 25,000 documents as the database, and three disjoint sets of 1,000 documents used as train, validation and test queries. To compute *ground truth similarities* between two documents we looked at the intersection of the sets of topics assigned to each of the documents; specifically, we counted the number of unique topic nodes that are assigned to both documents.

For all datasets, we create the ranking constraints for training as follows. For each training query:

1. Find its top k nearest neighbors among the database items using the ground-truth similarity scores.
2. For each neighbor sample l items at random from the remaining database items.
3. Generate the corresponding $k * l$ ranking triplets.

A total of $q * k * l$ ranking constraints were generated by this process. In all of our experiments, k to 40 and l to 4 resulting in a total of 320,000 training ranking triplets for the SUN database and 80,000 training triplets for the REUTERS-IMG database, and 160,000 training triplets for the RCV1 dataset.

As image representation we used spatial hog histograms [7], a standard representation widely used for image classification and object detection. To represent texts, we used the standard bag-of-words representation that comes with the data, where each term is weighted by IDF. In the ranking models, the elementary similarity functions are computed using these base features, in particular we take each feature x_j and set $k^j_{q,r} = \exp^{-|\mathbf{x}_{qj} - \mathbf{x}_{rj}|}$.

4.2 Comparison of Models

We evaluate the ranking performance of three models:

- Global SVM: A single weighted combination of elementary similarities is learned using all constraints. This corresponds to the standard ranking SVM [11]
- Transductive SVM: A weighted combination of elementary similarities is learned for each test query using only the ranking constraints from its k nearest training queries. [3] For comparison with other models we report results for the best performing k.

[3] We only tried this model on the image datasets. Running this model requires access at test time to a function that retrieves the k nearest training queries for a given test query. We used the histogram intersection between hog features as the similarity score.

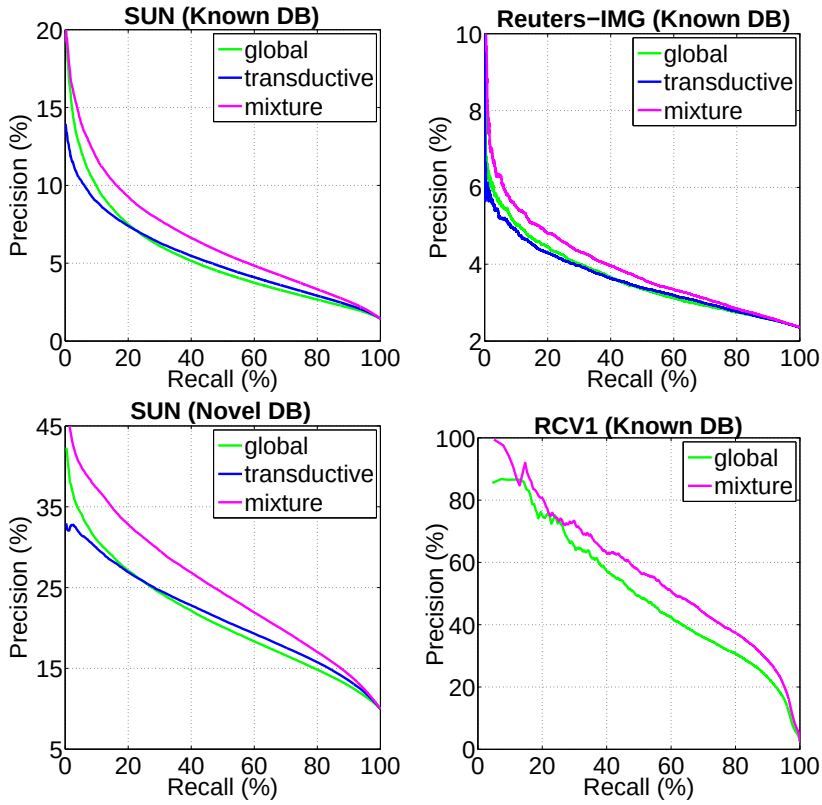

Fig. 1. Precision-recall curves of the three databases using the three different models. On the SUN database, we show retrival curves on the "known database" and the "novel database" settings. For the other datasets we only present the "known database" setting.

– Mixture: A mixture ranking model. For comparison with other models we report results for the optimal number of hidden classes, which was chosen using the validation set.

We used a primal projected sub-gradient algorithm [19], and tuned the regularization constants on the validation data. The mixture model took typically less than 10 global iterations to converge. To measure ranking performance we report precision-recall curves for the task of retrieving the 100 top neighbors in the database for a number of test queries. For the SUN data we also evaluate the ranking models in a "novel database" setting, where the database of images is different from that of training.

4.3 Results

Figure 1 shows performance of the different ranking models on the three datasets.[4] The mixture model outperforms the global and transductive models in all cases. [5] To give a concrete example, in the known-database setting for the SUN data, for a recall of 20% the global and transductive models obtain around 7% precision versus 9% precision of the mixture model. This means that to obtain 20 of the top 100 neighbors, on average 286 images of the 6,000 database images need to be browsed with the global and transductive models. In contrast, for the mixture model 220 images must be browsed on average. This constitutes a 25% error reduction. To illustrate the behavior of the mixture model, Figure 3 shows some example test queries with their corresponding ground-truth neighbors, together with neighbors predicted by the mixture ranking model.

Figure 2 illustrates the behavior of the methods on validation data for SUN. The left plot shows the average ranking given to the top 100 true neighbors as a function of the number of neighbors used to train the transductive model. For example, an average ranking performance of $p\%$ means that on average a true neighbor is ranked among the top $p\%$ of the database. The best performance is obtained using around 128 neighbors. The right plot shows average ranking as a function of the number of hidden classes used by the mixture model. We observe the largest improvement when the number of hidden classes is incresed from two to four, and after that we observe smaller improvements. In fact, it seems that four is the "intrinsic dimensionality" discovered by the mixture model because even the models with more latent classes put more than 90% of their probability mass on the top four classes. By probability mass of each class we mean: $\mathrm{mass}(g) = \sum_{q \in \mathcal{Q}} \frac{1}{|Q|} p(h_q = g \,|\, q)$.

Figure 4 illustrates, for some latent classes, the top 10 images with highest probability for that class (we omit latent classes with probability mass lower than 0.0001), for the SUN database, where we used the model with optimal number of hidden classes (8). The numbers below each column correspond to the probability mass assigned to that class by the model. The model seems to have learned one latent class for indoor scenes and 4 latent classes for outdoor scenes.

5 Conclusion

Since their introduction, ranking SVM models [11] have become a powerful tool for training retrieval systems. All we need for training a model are retrieval examples in the form of triplet constraints. When we try to apply these approaches to textual and image databases, an important challenge is the heterogeneous nature of the data space.

[4] Note that the known-database and novel-database precision-recall performances are not directly comparable with each other. This is because the tasks are different. In the SUN data the known-database task is to rank $6,000$ database images while the novel-database task is to rank $1,000$ images. What is important is that the relative performance of the different models is maintained across the two settings.

[5] We conducted significance tests, similar to [15], which in essence consist of a sign test for the task of ranking a pair of random images with respect to its similarity to a random query. We found all differences between the mixture and other models to be significant with $p < 0.001$.

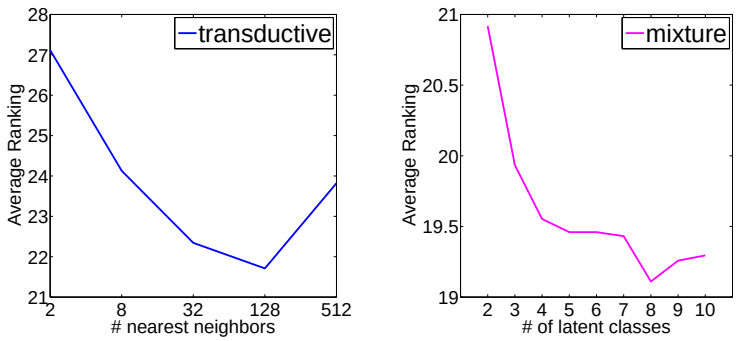

Fig. 2. Validation results on SUN. Left: average ranking of the transductive model as a function of the number of nearest neighbors used to train the local models. Right: average ranking of the mixture model, as a function of the number of latent classes.

To address this problem, previous work has explored local approaches where one trains different retrieval functions for each neigborhood in the data space. The problem of this approach is that neigborhoods need to be defined a priori, and therefore the resulting model is susceptible to the quality of these neighborhoods.

The main contribution of this paper is to show that a neighborhood function can be learned jointly with a ranking function that is a mixture of specialized ranking functions. We regard the assignment of queries to neighborhoods as a latent variable problem, resulting in a latent variable ranking model which is novel to our knowledge. Although the optimization problem to train this type of models is not convex, we derive an efficient alternating method that works very well in practice. Experiments over three datasets show that the mixture model outperforms both a standard SVM ranker and and transductive-style approach.

References

1. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning a mahalanobis metric from equivalence constraints. J. Mach. Learn. Res. 6, 937–965 (December 2005)
2. Basu, S., Banerjee, S., Mooney, R.: Semi-supervised clustering by seeding. In: ICML (2002)
3. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the twenty-first international conference on Machine learning. pp. 11–. ICML '04, ACM, New York, NY, USA (2004)
4. Bottou, E., Vapnik, V.: Local learning algorithms. Neural Computation 4, 888–900 (1992)
5. Chechik, G., Sharma, V., Shalit, U., Bengio, S.: Large scale online learning of image similarity through ranking. Journal of Machine Learning Research 11, 1109–1135 (March 2010)
6. Choi, M.J., Lim, J.J., Torralba, A., Willsky, A.S.: Exploiting hierarchical context on a large database of object categories. In: Proceedings of CVPR (2010)
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
8. Frome, A., Sha, F., Singer, Y., Malik, J.: Learning globally-consistent local distance functions for shape-based image retrieval and classification. In: In ICCV (2007)

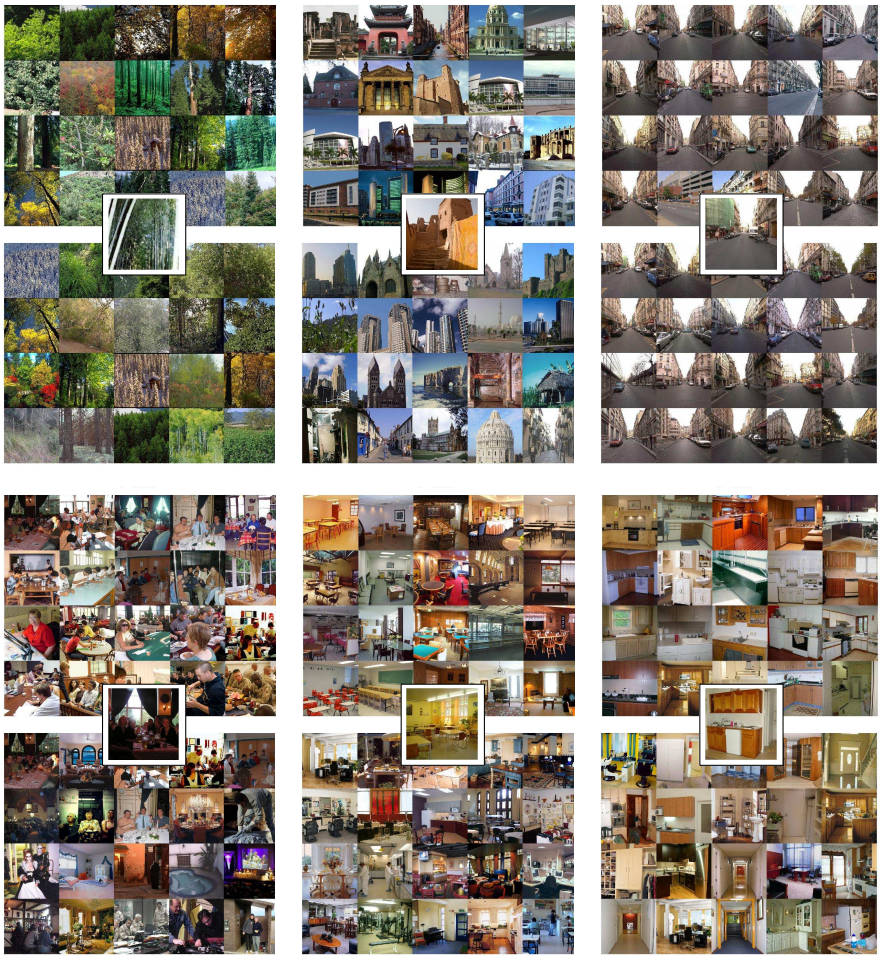

Fig. 3. Examples of six queries of the SUN database (images on the center) together with the top 20 ground truth neighbors (block on top of each query image) and the top 20 neighbors predicted by the the ranking mixture model (block at the bottom of each query image). The dataset includes a large variety of indoor and outdoor scenes.

9. Frome, A., Singer, Y., Malik, J.: Image Retrieval and Classification Using Local Distance Functions. In: Schölkopf, B., Platt, J.C., Hoffman, T., Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) NIPS. pp. 417–424. MIT Press (2006)
10. Hertz, T., Bar-hillel, A., Weinshall, D.: Learning distance functions for image retrieval. In: Proceedings of CVPR. pp. 570–577 (2004)
11. Joachims, T.: Optimizing search engines using clickthrough data. In: ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD). pp. 133–142 (2002)

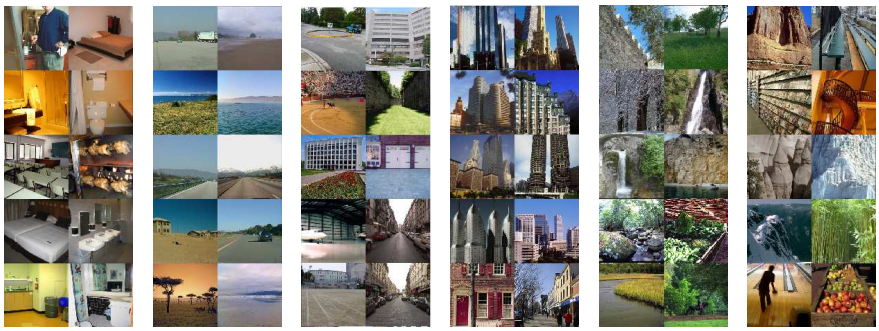

Fig. 4. Analysis of the latent classes on the SUN database. For some latent classes, the plot shows the 10 queries with highest probability for the class. Latent classes are ordered by the probability mass of the class (number below images).

12. Klein, D., Kamvar, S., Manning, C.: From Instance Level Constraints to Space Level Constraints: Making the most of prior knowledge in data-clustering. In: CVPR (2005)
13. Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.: Learning the Kernel Matrix with Semidefinite Programming. In: Journal of Machine Learning Research (2004)
14. Lewis, D.D., Yang, Y., Rose, T.G., Li, F., Dietterich, G., Li, F.: Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research 5, 361–397 (2004)
15. Quattoni, A., Carreras, X., Collins, M., Darrell, T.: An efficient projection for l1,infinity regularization. In: Proceedings of ICML (2009)
16. Quattoni, A., Collins, M., Darrell, T.: Learning visual representations using images with captions. In: Proceedings of CVPR (2007)
17. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: Labelme: A database and web-based tool for image annotation. Tech. rep., Tech. Rep. MIT-CSAIL-TR-2005-056, Massachusetts Institute of Technology (2005)
18. Schultz, M., Joachims, T.: Learning a Distance Metric from Relative Comparisons. In: NIPS (2004)
19. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In: Proc. of Intl. Conf. on Machine Learning. pp. 807–814 (2007)
20. Urtasun, R., Darrell, T.: Sparse probabilistic regression for activity-independent human pose inference. Proceedings of CVPR pp. 1–8 (2008)
21. Wagstaff, K., Cardie, C., rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: ICML (2001)
22. Yan, R., Hauptmann, A.G.: Probabilistic latent query analysis for combining multiple retrieval sources. In: In Proceedings of the 29th international ACM SIGIR conference. pp. 324–331. ACM Press (2006)
23. Zhang, H., Berg, A.C., Maire, M., Malik, J.: Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In: Proceedings of CVPR. pp. 2126–2136 (2006)