Accuracy and specificity trade-off in k-nearest neighbors classification

Luis Herranz, Shuqiang Jiang

Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing China

Abstract. The k-NN rule is a simple, flexible and widely used non-parametric decision method, also connected to many problems in image classification and retrieval such as annotation and content-based search. As the number of classes increases and finer classification is considered (e.g. specific dog breed), high accuracy is often not possible in such challenging conditions, resulting in a system that will often suggest a wrong label. However, predicting a broader concept (e.g. dog) is much more reliable, and still useful in practice. Thus, sacrificing certain specificity for a more secure prediction is often desirable. This problem has been recently posed in terms of accuracy-specificity trade-off. In this paper we study the accuracy-specificity trade-off in k-NN classification, evaluating the impact of related techniques (posterior probability estimation and metric learning). Experimental results show that a proper combination of k-NN and metric learning can be very effective and obtain good performance.

1 Introduction

Visual recognition is a basic problem in computer vision, and is a key component in image retrieval and automatic annotation systems. User generated annotations tend to be ambiguous and noisy, and often not representative of the content. In addition, the tagging process is tedious and time consuming, so often users just do not annotate their own content. Automatically classifying and annotating images is thus highly desirable.

Exploiting hierarchical semantic relations between related classes can improve the classification accuracy[1] and can be used to speed up classification[2]. However, the goal of these methods is still to predict a label among the original set of training labels (leaf nodes in the hierarchy).

In practice, when the number of categories becomes larger and the differences between them are more subtle (fine-grained classification), the accuracy is not high, and the suggested label is often not accurate. Recently, Deng et al[3] proposed a different approach with the objective of trading off accuracy and specificity. When the confidence in a particular SVM prediction is not high enough, hierarchical semantic relations are leveraged to suggest less specific tags, but with higher confidence. Thus, if the prediction of a dog breed is not reliable, perhaps just simply suggest *dog*.

The k-nearest neighbors (k-NN) rule is a widely used non-parametric classification method despite its simplicity. It has the advantage of not requiring training and the capability to easily incorporate new information. The idea is to find the k nearest neighbors training samples to a query feature vector and select the most frequent class. However, this may be difficult in practice when the number of classes increases and becomes more difficult to discriminate between closely related classes, leading to high uncertainty in the prediction (see Fig. 1a). Recently, semantic hierarchies have been used in k-NN classification, in particular to learn tree structured metrics[4], but accuracy-specificity trade-offs in k-NN have not been studied.

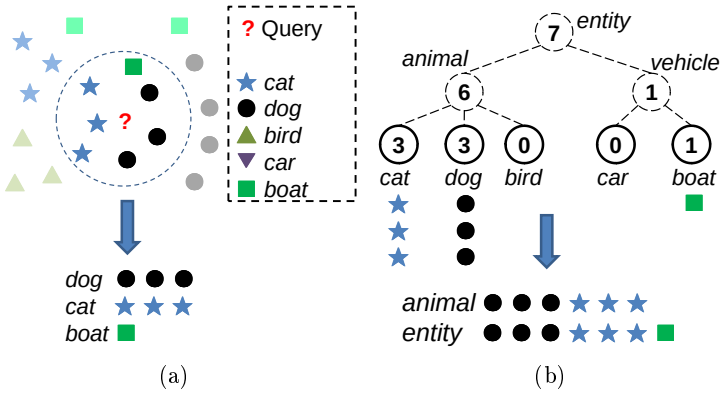


Fig. 1. k-NN classification with semantic hierarchy: (a) votes in flat classification, (b) vote aggregation in broader concepts.

In this paper we study the accuracity-specifity trade-off in k-NN classification, by considering voting in internal nodes of the hierarchy (see Fig. 1b). We focus on related techniques, such as metric learning[3,5,6,7] and posterior probability estimation[8,9]. We include specific analysis and evaluation metrics (semantic similarity, accuracy-specificity F score) to evaluate the performance of the method in the accuracy-specificity framework. The rest of the paper is organized as follows. Section 2 describes the accuracy-specificity framework. In Section 3 we describe its extension to k-NN classification. Section 4 and 5 present the experimental evaluation and the conclusions.

2 The accuracy-specificity framework

In this section we briefly review the specificity-accuracy framework and the Dual Accuracy Reward Trade-off Search (DARTS) algorithm proposed by Deng et al[3].

2.1 Accuracy and specificity

In flat classification, only one class is considered correct, so the prediction is either right or wrong. However, in hierarchical classification not only the leaf concept is correct but also the ancestors. The main difference is that broader concepts are easier to predict correctly but at the same time provide less useful information. This can be modeled as a trade-off between accuracy and specificity. We can describe a hierarchy of concepts as a graph $H = (V, E)$, with each node $v \in V$ representing a concept. The leaf nodes $Y \subset V$ are mutually exclusive concepts and form the classes for flat classification. Let $x \in X$ represent a training feature vector and $y \in Y$ the corresponding label. A classifier $\tilde{y} = f(\tilde{x})$ predicts a label $\tilde{y} \in V$ for the test feature vector $\tilde{x}$. Now, evaluating the classifier on a test set, we can obtain the average accuracy as $A(f) = \frac{1}{|S|} \sum_{x \in S} [f(x) \in \pi(y)]$ where $\pi(y)$ represents the set of correct predictions (i.e. those in the path from the correct leaf node to the root node, including y), and $[P]$ is 1 if the statement P is true, otherwise is 0.

In a hierarchy of concepts, several nodes are correct classifications, however we should choose the most informative one, which in our case is the most specific one. Thus, we would prefer *cat* to *entity*, and *Siamese* to *cat*, provided that all of them are correct predictions. A suitable measure is the information gain with respect to predicting the root node[3], measured as $g_v = log_2 |Y| - log_2 \sum_{y \in Y} [v \in \pi(y)]$ which increases from the root node (zero gain) to leaf nodes (maximum gain). We assumed all leaf nodes are equally probable (i.e. uniform prior). Fig. 2a shows the corresponding information gain of the semantic hierarchy in Fig. 1b. We can compute the average information gain (of correct predictions) in the classifier f as $G(f) = \frac{1}{|S|} \sum_{x \in S} g_{f(x)} [f(x) \in \pi(Y)]$.

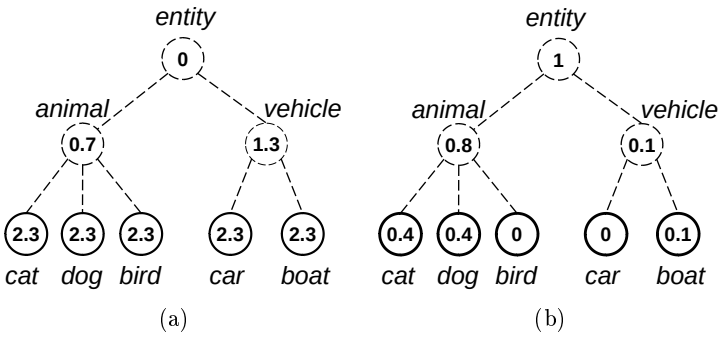


Fig. 2. Example of k-NN classification with hierarchical concepts: (a) information gain at each node, (b) estimation of the likelihood in the example in Fig. 1a ($k = 7$).

2.2 Making conservative predictions

A typical flat classifier does not consider other option than venturing a (fine grained) prediction, no matter whether it is right or wrong. However, sometimes it is possible to estimate the confidence of the classifier in its prediction, which enables a way to reject the prediction. In the accuracy-specificity framework this can be seen as predicting the root node, when the classifier estimates that the prediction is highly probable to be wrong

$$f(x) = \begin{cases} f(x) & \text{if } p(v|x) \geq \alpha \\ \tilde{v} & \text{otherwise} \end{cases}$$

where $\tilde{v} \in V$, and $0 < \alpha \leq 1$ is an arbitrary minimum accuracy to accept a prediction.

When a full hierarchy is available, intermediate nodes can also be selected, allowing a finer trade-off between accuracy and specificity. In particular, the objective of the DARTS algorithm[3] is to maximize the information gain given a certain accuracy guarantee α

$$\begin{aligned} \text{maximize} \quad & G(f) \\ \text{s.t.} \quad & A(f) \geq \alpha \end{aligned} \tag{1}$$

Using Lagrange multipliers, the constrained optimization problem (1) can be expressed as

$$L(f) = G(f) + \lambda (A(f) - \alpha) \tag{2}$$

and then find the value of λ maximizing the Lagrange function $L(f)$. In the DARTS algorithm, this value is found by estimating posterior probabilities and selecting the node with the maximum expected information gain. For SVMs, Platt scaling[10] is used to estimate probabilities at each node, and the value of λ is found via binary search.

3 Application to *k*-nearest neighbors classification

Including the internal nodes of a hierarchy in k-NN can be done by simply aggregating votes from children nodes (see Fig. 1b). Then the only parameter to be set is k. However, good performance depends on the metric used to measure the distance. To include rejection and trading off accuracy and specificity, we also need to estimate posterior probabilities.

3.1 Estimating class probabilities

For a given k and a test image x, we can define its k-nearest neighborhood $\mathcal{N}_k(x)$ as a set with the points in the training set with lower distance to x. For convenience we assume they are ordered by increasing distance. The simplest estimator of the posterior probability k-NN is the fraction of neighbors that belong to that class[8]

$$p\left(v|x\right) = \frac{k^{(v)}}{k} = \frac{1}{k} \sum_{u \in \mathcal{N}_k(x)} \left[u \in v\right] \tag{3}$$

where $k^{(v)}$ indicates the number of the k-nearest neigbors belonging to the class represented by the node v. Note that this estimator is also valid for internal nodes.

The probability estimator in (3) ignores the distance and the order of the neighbors. We can include weights in (3) to emphasize closer neighbors and estimate the probability as

$$p\left(v|x\right) = \frac{1}{k} \sum_{\substack{i=1 \\ u_i \in \mathcal{N}_k\left(x\right)}}^{k} w_i \left[u_i \in v\right] \tag{4}$$

where u_i represents the i-th nearest neighbor and $w_i \geq 0$ the corresponding weight. The weights are learned using the method proposed by Atiya[9], which uses a softmax representation to model weights and then maximizes the likelihood. We enforce decreasing weights with distance $w_1 \geq \ldots \geq w_i \geq \ldots \geq w_k$ to avoid randomness.

3.2 Metring learning

The selection of the nearest neighbors depends essentially on the particular metric $d\left(x_i, x_j\right)$ used to evaluate distances. Common used metrics (e.g. , such as the Euclidean distance, may not be the most suitable in general. If training data is available, an appropriate metric can be learned to capture the specific characteristics of the feature space.

During the last ten years, automatic metric learning has been intensively studied, resulting in large number of learning methods (see [7] for a recent survey). There is no clear algorithm performing better than others, and sometimes the Euclidean distance still has better performance than a learned metric. Due to the complexity of visual feature spaces, the performance of different methods usually varies significantly from problem to problem, and from dataset to dataset. For that reason, we will consider three widely used metric learning approaches and evaluate them in our case.

Most metric learning methods learn a distance of the form $d_M\left(x_i, x_j\right) = \sqrt{\left(x_i - x_j\right)^T \boldsymbol{M} \left(x_i - x_j\right)}$. The metric is parametrized by the positive semi-definite matrix $\boldsymbol{M}$, which is usually learned from a regularized convex optimization problem, with constraints representing the relations between pairs of samples. In such pairwise relations, pairs can be reduced to two classes: similar (same label) and dissimilar (different label). In particular the ITML[11] algorithm enforces constants of the type $d^2_{\boldsymbol{M}}\left(x_i, x_j\right) \leq t_{similar}$ if $y_i = y_j$ for similar pairs (and $d^2_{\boldsymbol{M}}\left(x_i, x_j\right) \geq t_{dissimilar}$ if $y_i \neq y_j$ for dissimilar pairs). The cost function is the Bregman divergence between $\boldsymbol{M}$ and a target matrix (typically the identity).

In the large margin metric learning (LMNN) framework, the goal is to pull target neighbors (same label) into the k-neighborhood, and push impostors (different label) away. Here, the constraints are relative and local to the neighborhood. Given triplets (x_i, x_j, x_k), the distance to impostors must be larger (with a safety margin) than the distance to target neighbors ($d^2_{\boldsymbol{M}}(x_i, x_j) \leq d^2_{\boldsymbol{M}}(x_i, x_k) + 1$, where x_j represents a genuine neighbor of x_i and x_k an impostor. We considered the original LMNN algorithm[5] and BoostMetric[6] which decomposes $\boldsymbol{M}$ in a combination of rank-one matrices which enables fast learning using boosting.

To satisfy the constraints, in practice slack variables are included in the constraints and in the objective function. One problem with these formulations is the polynomial complexity, as the number of constraints grows as $O\left(n^2\right)$ in the case of pairwise constraints and as $O\left(n^3\right)$ in the case of triplets. In practice, a small subset of these constraints is subsampled to keep a reasonable complexity.

4 Experimental evaluation

4.1 Dataset and settings

We evaluated the performance of k-NN classification over the ILSVCR65 dataset and the corresponding semantic hierarchy[3][1]. The dataset contains images of *animals* and *vehicles* further classified in *birds*, *cats* and *dogs*, and *boats* and *cars*, respectively (see Fig. 1b). Note that there is no *mammal* category in this taxonomy, so *birds* are as similar to *cats* and *dogs* as *cats* are to *dogs*. Finally these categories are further classified in 7, 5, 31, 5 and 9 fine-grain categories (57 leaf nodes), respectively. Although leaf nodes are balanced (each flat classifier is trained with the same number of images) the semantic hierarchy is not, resulting in a strong bias towards some parent nodes, such as *animal*, and particularly *dog*. Labels are assigned only at leaf nodes, represented each with 100/50/150 images in the training/validation/test sets.

To represent the images we used the LLC[12] features provided with the dataset. The original features include two spatial pyramid levels (1x1 and 3x3), for a total of 100K dimensions. As k-NN is not practical in such high dimensional space, we only kept the first level of the spatial pyramid (10K dimensions) and reduced the features to 50 dimensions using PCA.

4.2 Results

In our experiments we study the performance of different accuracy-specificity strategies, including flat k-NN classification (FLAT), flat with rejection (FLAT-REJ) and the DARTS method, evaluated for different values of α. We learned

[1] The ILSVCR65 dataset, hierarchy and the DARTS source code are available at http://www.image-net.org/projects/hedging/

metrics using the LMNN, BoostMetric and ITML algorithms using the implementations provided by their authors[2][3][4]. Due to the large number of samples, considering all the possible pairs/triplets constraints is extremely costly, so we sampled a significant number (i.e. still millions of constraints). We used grid search to adjust the corresponding parameters, measuring the k-NN classification accuracy (k=20) in the validation set to prevent overfitting.

The average classification accuracy in the test set is shown in Table 1. We see that, unfortunately, using the metrics learned with LMNN and BoostMetric lead to worse performance than simply using the Euclidean distance. In contrast, the metric learned with ITML improves significantly the accuracy. This may suggest that the large margin framework is not suitable for this particular problem, due to the huge number of triplet constraints, of which only a small fraction are considered in practice. This small fraction (still millions of constraints) may not be large enough to learn a metric properly. Typically, these algorithms are evaluated with very successful results for smaller datasets with relative low dimensional feature spaces and few classes, when all or a significant fraction of triplet constraints can be considered. Our case is more challenging and in practice we had to discard a large amount of triplet constraints to keep the training time reasonable. In contrast, ITML considers pairwise constraints, which scale better with the number of samples and classes, and results in an improved accuracy.

We also measured the semantic similarity between the prediction and the ground truth (leaf node) as[13]

$$s\left(v,y\right)=\frac{\left|\pi'\left(v\right)\bigcap\pi'\left(y\right)\right|}{\max\left(\left|\pi'\left(v\right)\right|,\left|\pi'\left(y\right)\right|\right)}$$

where v is the predicted node, y is the ground truth leaf node and $\pi'\left(v\right)$ indicates the path from v to the root node (excluding v) and $\left|\pi'\left(v\right)\right|$ indicates the length of that path. In contrast to accuracy, that only considers a binary outcome for a test sample (either correct or wrong classification), the semantic similarity gives a graded score which may be a more suitable measure when including internal nodes as predictions (e.g. a *dog* should be more similar to a *cat* than to a boat, because both are *animals*). In fact, the flat classification accuracy of SVM[5] is higher than k-NN with ITML, but the semantic similarity is lower, and as we will see this is related with a poorer accuracy-specificity performance. This suggest that, after metric learning, the feature space is structured in a more semantically meaningful way, in which leaf node misclassifications are likely to be predicted as a still relatively similar leaf node class. In contrast, SVM does not change the structure of the feature space but finds nonlinear decision boundaries.

[2] http://www.cse.wustl.edu/~kilian/code/lmnn/lmnn.html
[3] http://code.google.com/p/boosting/
[4] http://www.cs.utexas.edu/~pjain/itml/
[5] In [3] SVM achieves higher classification accuracy using spatial pyramid and 100K-dim features, in contrast to the 50-dim features (no spatial pyramid) used in our experiments.

Thus the accuracy may be higher, but misclassifications tend to be less related semantically to the true class than in k-NN with metric learning.

Evaluation metric	k-NN				SVM
	Euclidean	LMNN	BoostMetric	ITML	
Accuracy (%)	18.02	16.34	9.87	21.19	**24.01**
Semantic similarity(%)	60.94	59.79	54.15	**64.16**	62.01

Table 1. Flat classification results.

The corresponding accuracy-specificity curves were obtained varying the value of α, from 0 to 0.99 (note that FLAT-REJ with α=0 corresponds to the flat classifier). Specificity is measured as normalized information gain. Fig. 3 shows the curves for different decision, metric learning methods and $k = 30$. We can observe that, in terms of accuracy-specificity trade-off, DARTS consistently outperforms FLAT-REJ, increasing the accuracy and still keeping a higher average specificity. In addition, an appropriate metric improves significantly the performance, not only in flat k-NN classification (compare Table 1) but also in accuracy-specificity curves.

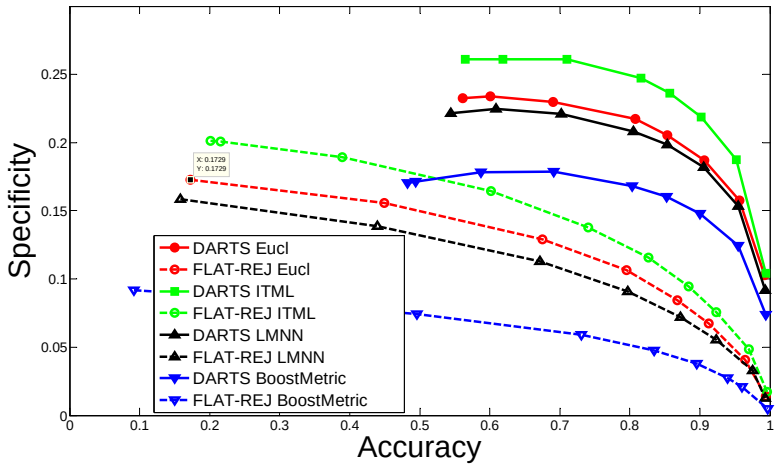


Fig. 3. Effect of metric learning over the accuracy-specificity curve ($k = 30$).

As shown in Fig. 4, the choice of the method used to estimate posterior probabilities is not so critical. We also compare the curves for several values of k, with larger neighborhoods performing better (we show more results later). In-

terestingly, Atiya's method[9] improves slightly the performance for large neighborhoods, while not being useful in smaller ones.

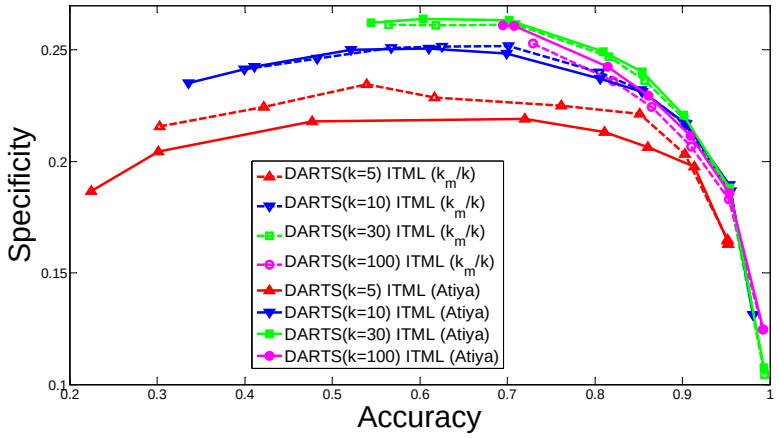


Fig. 4. Effect of the probability estimation method over the accuracy-specificity curve.

In order to further study the effect of the parameter k in the performance, we also use a variation of the F-score to evaluate the accuracy-specificity performance

$$F(f) = \max_{\lambda \in [0,1)} \left(\frac{2\,A(f_\lambda)\,G(f_\lambda)}{A(f_\lambda) + G(f_\lambda)} \right)$$

where f_λ is the classifier that maximizes (2) for λ. As mentioned earlier, small neighborhoods are not suitable in this problem. The size of the neighborhood trades off accuracy in the estimation and locality. A more accurate estimation of the posterior probability requires more neighbors, but then the neighborhood in the feature space is more spread, leading to less localized prediction. The accuracy-specificity F score is a compact way to compare different methods (see Fig. 5, using Atiya's estimator) and their dependency with k. First, we can see that the performance gain of both DARTS and metric learning over the flat classifiers and the Euclidean distance is consistent and very robust to the particular choice of k. Actually, for neighborhoods large enough (say $k \geq 20$) F scores are almost constant. For smaller values the performance decreases significantly. We can also observe a slow but steady decay in the performance due to a less localized prediction. For that reason we set $k = 30$ in the remaining experiments.

Finally, Fig. 6 compares the performance of DARTS with SVM classification and DARTS with k-NN. The performance of SVM is comparable to k-NN with Euclidean distance. Note that in this variation there is no training at all if we use (3) to estimate posterior probabilities. Using ITML k-NN outperforms SVM

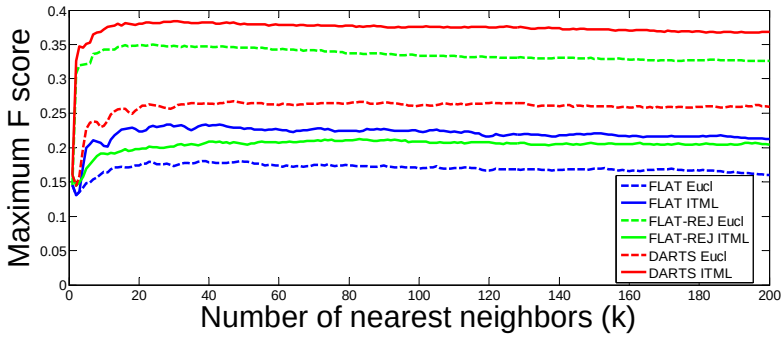


Fig. 5. Best accuracy-specificity F score for different k.

for this feature space. As we mentioned earlier, a better metric not only helps to increase the flat accuracy, but also helps to structure the feature space in such a way that semantic relations are better preserved. The resulting nearest neighbors are also more semantically related, which provides a better way to estimate the posterior probability at different levels of the hierarchy, so the DARTS method can make better decisions. As we discussed earlier, the average semantic similarity of predictions in the flat classifier is a good indicator of how suitable the feature space is for this framework.

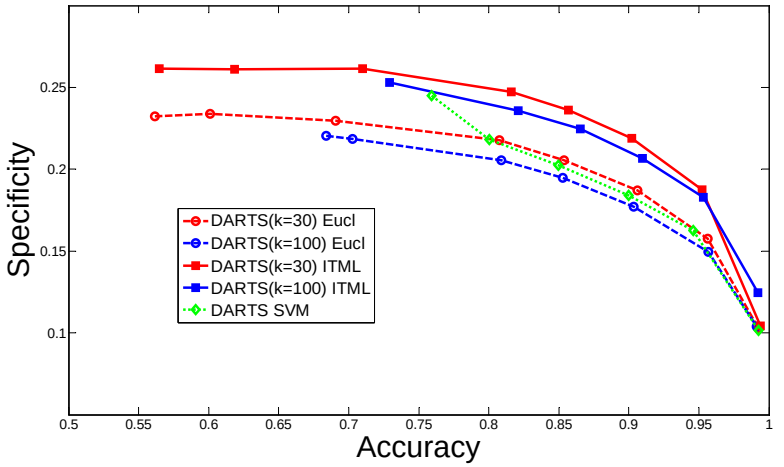
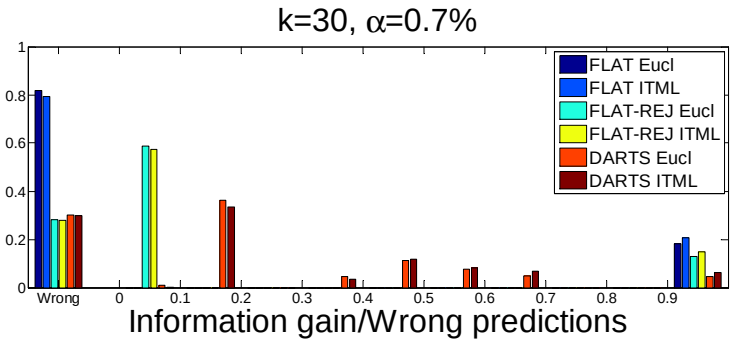


Fig. 6. Comparison of k-NN and SVM approaches.

Fig. 7 shows the fraction of predictions and their associated information gain (or predicted incorrectly). The flat classifier always ventures a prediction so the amount of both correct and incorrect is relatively high. With a rejection option, many wrong predictions are rejected and labeled as *entity* (root node) but also some correct predictions. DARTS further reduces the amount of very specific correct predictions, and keeps a similar rate of wrong predictions as FLAT-REJ. However, the amount of predictions assigned to the root node is reduced considerably, and they are assigned to more specific intermediate nodes, which is more useful. Higher values of α reduce the number of wrong predictions but also increase the number of uncertain predictions (i.e. *entity*) and also reduce the number of correct predictions with highest specificity.



(a)

Fig. 7. Information gain and wrong predictions.

Fig. 8 shows some examples of how wrong predictions in the flat classifier can be recovered as less specific labels, but still more useful than a wrong prediction. FLAT-REJ can sometimes anticipate a wrong prediction but predicts it as *entity*. In contrast, DARTS provides more specific labels. Some examples of the k-nearest neighbors for specific images are shown in Fig. 9. Although neighbors of the same class are not as many as desirable, the selected neighbors often belong to related superclasses (e.g. *dog*, *car*), so it seems reasonable to use them to infer a useful label.

5 Conclusion

The k-NN framework is simple but powerful. It can achieve very competitive performance, and even outperform SVM using the same features. As a non-parametric method, it can incorporate new samples to the model without need for retraining (other than metric learning and/or weights in (4) if desired to update them, but not strictly necessary).

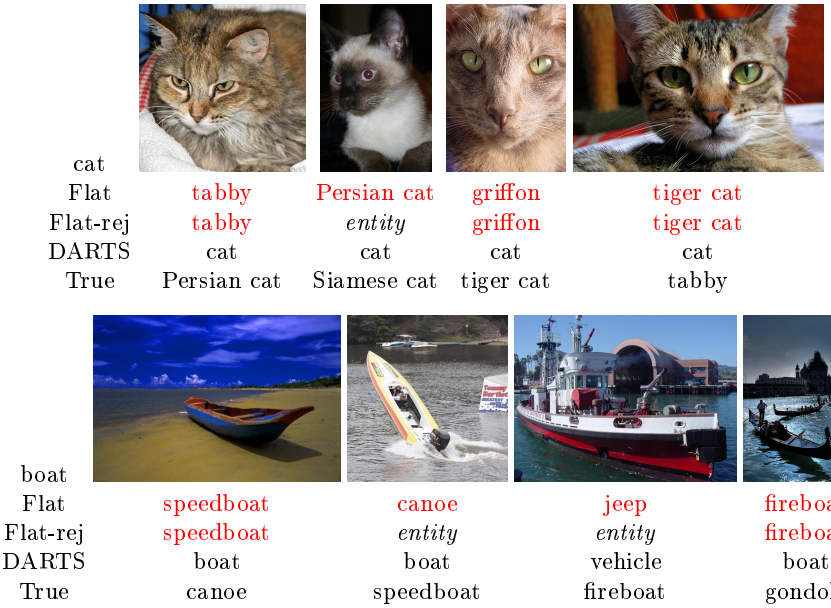
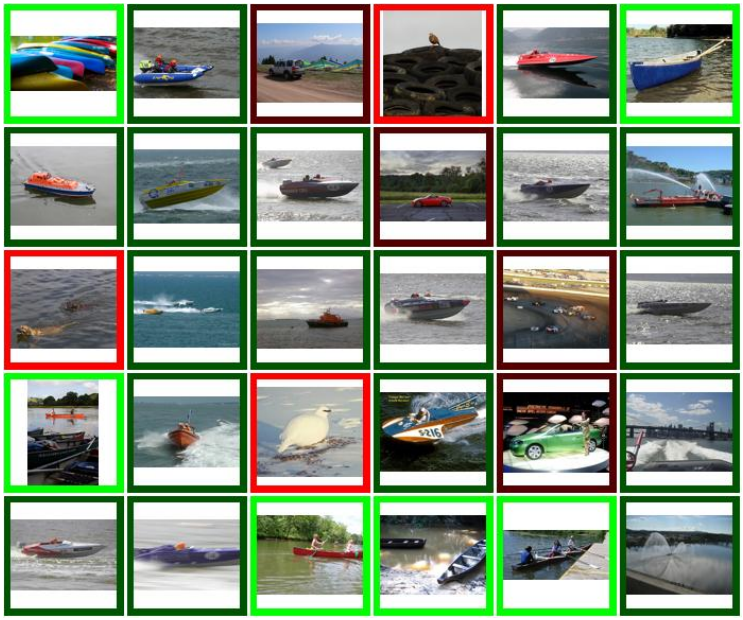


Fig. 8. Examples of wrong predictions in flat classification recovered to less specific but correct predictions ($k = 30$, $\alpha = 0.7$). Rejected cases in FLAT-REJ are labeled as *entity* (non-informative).

We observed that the structure of the feature space after metric learning reflects a more suitable structure to deal with flat misclassifications, which is exploited in k-NN classification to effectively trade off accuracy and specificity. This results in even better accuracity-specificity trade-off than classifiers with higher flat classification accuracy, such as SVM. In this case, semantic similarity is more suitable than flat accuracy to measure the performance. We also noticed that a wrong choice of metric learning method may result in disappointing results, particularly in the experiments performed with methods using triplet constraints.

Experiments show promising results with relatively low dimensionality (50 dimensions). SVM can still achieve better performance resorting to very high dimensional features (100K dimensions)[3]. In future work we would like to target larger scale datasets and also higher dimensional features. However, scalability in k-NN classification is the main obstacle, so approximate search methods and structures may be necessary. Search in very high dimensional features spaces is also very demanding. Moreover, current metric learning methods do not scale well either, due to the use of paiwise and triplet constraints. Thus, further fundamental research in k-NN classification tools and metric learning methods seems necessary to make it practical for larger datasets and higher dimensional spaces.

cat (Persian cat)



boat (canoe)

Fig. 9. Nearest neighbors (by increasing distance) to the first image of each broad category in Fig. 8. Green (red) frames indicate the level of semantic similarity (dissimilarity).

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China: 61322212, 61035001 and 61350110237, in part by the Key Technologies R&D Program of China: 2012BAH18B02, in part by National Hi-Tech Development Program (863 Program) of China: 2014AA015202, and in part by the Chinese Academy of Sciences Fellowships for Young International Scientists: 2011Y1GB05.

References

1. Fergus, R., Bernal, H., Weiss, Y., Torralba, A.: Semantic label sharing for learning with many categories. In: ECCV. (2010) 762–775
2. Griffin, G., Perona, P.: Learning and using taxonomies for fast visual categorization. In: CVPR. (2008)
3. Deng, J., Krause, J., Berg, A.C., Li, F.F.: Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In: CVPR. (2012) 3450–3457
4. Hwang, S.J., Grauman, K., Sha, F.: Learning a tree of metrics with disjoint visual features. In: NIPS. (2011) 621–629
5. Weinberger, K.Q., Saul, L.K.: Distance Metric Learning for Large Margin Nearest Neighbor Classification. JMLR **10** (2009) 207–244
6. Shen, C., Kim, J., Wang, L., van den Hengel, A.: Positive semidefinite metric learning using boosting-like algorithms. JMLR **13** (2012) 1007–1036
7. Kulis, B.: Metric learning: A survey. Foundations and Trends in Machine Learning **5** (2013) 287–364
8. Fukunaga, K., Hostetler, L.: k-nearest-neighbor bayes-risk estimation. Information Theory, IEEE Transactions on **21** (1975) 285–293
9. Atiya, A.F.: Estimating the posterior probabilities using the k-nearest neighbor rule. Neural Comput. **17** (2005) 731–740
10. Platt, J.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Advances in Large Margin Classifiers. MIT Press (1999) 61–74
11. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: ITML. (2007) 209–216
12. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T.S., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR. (2010) 3360–3367
13. Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Computational Linguistics **32** (2006) 13–47