Learning Hierarchical Similarity Metrics

Nakul Verma[*] UC San Diego naverma@cs.ucsd.edu

Dhruv Mahajan[†] Sundararajan Sellamanickam[†] Vinod Nair[†]
[†] Yahoo! Labs, Bangalore
{dkm,ssrajan,vnair}@yahoo-inc.com

Abstract

Categories in multi-class data are often part of an underlying semantic taxonomy. Recent work in object classification has found interesting ways to use this taxonomy structure to develop better recognition algorithms. Here we propose a novel framework to learn similarity metrics *using the class taxonomy. We show that a nearest neighbor classifier using the learned metrics gets improved performance over the best discriminative methods. Moreover, by incorporating the taxonomy, our learned metrics can also help in some taxonomy specific applications. We show that the metrics can help determine the correct placement of a new category that was not part of the original taxonomy, and can provide effective classification amongst categories local to specific subtrees of the taxonomy.*

1. Introduction

The performance of many classification algorithms relies heavily on having a good notion of similarity or a *metric* on the input space. Consider, for instance, an object recognition task with Support Vector Machines (SVMs): the right similarity kernel can significantly improve SVM's classification accuracy. Such similarity metrics can not only help in classification but are also shown to be effective in retrieval and embedding tasks [17, 11]. It thus comes as no surprise that a considerable amount of work focuses on learning representations that capture the necessary similarity information in data [22, 20, 1].

For the particular task of image categorization, one also has to deal with unique challenges specific to image data – image datasets tend to have 10s to 100s and now even up to 1000s of object categories, making classification especially hard. Computer vision practitioners are finding novel ways to improve the classification performance on such data. One fruitful line of work tries to exploit the inherent structure in the object categories. It turns out that object categories are often part of an underlying semantic taxonomy. Having access to this taxonomy structure has been shown to benefit both the accuracy as well as the scalability of learning algorithms [15, 16, 10].

[*]Work done while visiting Yahoo! Labs, Bangalore.

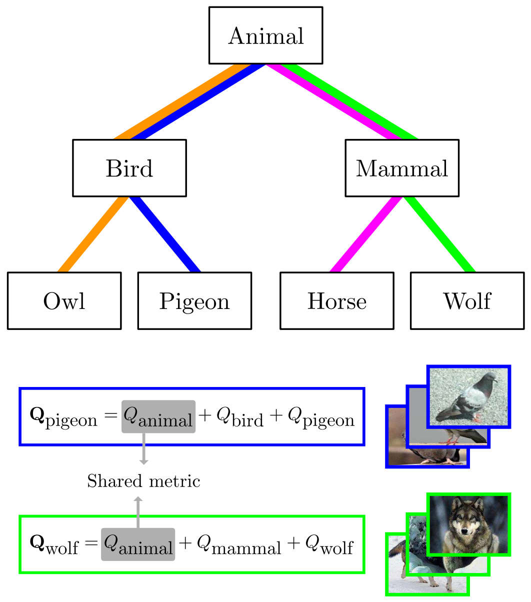

Animal
Bird
Mammal
Owl
Pigeon
Horse
Wolf
$\mathbf{Q}_{\text{pigeon}} = Q_{\text{animal}} + Q_{\text{bird}} + Q_{\text{pigeon}}$
Shared metric
$\mathbf{Q}_{\text{wolf}} = Q_{\text{animal}} + Q_{\text{mammal}} + Q_{\text{wolf}}$

Figure 1. Learning similarity metrics in a hierarchy. Each node t in the taxonomy is associated with its local metric Q_t. These local metrics are combined together ($\mathbf{Q}_t$) along the paths of the taxonomy for effective object categorization.

Our goal in this work is to enhance our learned similarity metrics when the taxonomy structure in the categories is known. We provide a probabilistic nearest-neighbor classification based framework for learning a set of *hierarchical* metrics that reflects the underlying class taxonomy. We associate a separate metric with each node of the taxonomy. The key to learning good metrics is to *share* the discriminating information among them. This is where the parent-child links in the taxonomy help. We train the metrics in such a way that the burden of discriminating between the various categories is shared between a node and its parent (see Figure 1): the metric associated with the root node contains the discrimination information among the categories associated with each of its children, while the metrics associated with the individual children contain discrimination information about the categories belonging to their children.

We show that learning these metrics in this way not only helps in representing our multi-category data such that a simple nearest neighbor classification provides good results, but also provides good results on more taxonomy specific tasks such as finding the correct placement of an unseen category in the taxonomy. In particular, we show that our our methodology benefits in all the following learning tasks.

Improved classification accuracy. We show state-of-the-art classification performance on several datasets where taxonomy information is available. We use various subtrees of the ImageNet dataset [6] and show that a nearest neighbor classifier using the learned metrics yields better performance than the best SVMs and embedding methods available in the literature.

Correct placement of unseen categories. We show that our learned metrics can also help determine the correct location of a new category that was not part of the original taxonomy. This is especially useful in augmenting large taxonomies with new categories, where manually inserting a node is cumbersome. Our method is straightforward, and can suggest a likely set of candidate locations for placing the new category. A human expert can then place the category at the appropriate location, reducing the overall manual effort.

Good locally consistent classification. Since we learn a hierarchy of metrics, our metrics have the unique ability to not only discriminate well amongst the given categories in the taxonomy, but to also discriminate well among categories in any *subtree* of the taxonomy. That is, in order to do well just on the categories belonging to a subtree of the taxonomy, one does not have to re-train a special set of similarity metrics on the subtree. The corresponding subset of metrics learned from the global taxonomy would do just as well. In that sense, our framework has an advantage of performing well on a specific categorization task when all it knows is the general taxonomy.

The rest of the paper is organized as follows. In the next section we discuss the related work. We formulate our framework in Section 3, giving specific instantiations that are helpful for image categorization. We then provide detailed experiments on the efficacy of our proposed model in Section 4, and conclude in Section 5.

2. Related work

With a few notable exceptions, most works in the literature either focus on developing good classifiers for taxonomy data without learning a metric [4, 3, 15, 16, 10, 7], or focus on developing good metrics without exploiting the taxonomy structure [22, 8, 12, 9, 1, 18]. For instance, Hierarchical SVM [4]—an example of the first kind—adapts the basic SVM classifier to learn a linear hyperplane for each category by accumulating contributions from each node along the path from the root to a leaf, but does not yield a metric. Neighborhood Component Analysis [8]—an example of the second kind—adopts a probabilistic approach to learn a nearest neighbor respecting low dimensional embedding, but is oblivious to the taxonomy structure. Our work explores how to *combine* the two paradigms and benefit from their respective strengths.

Taxonomy Embedding [21] and Label Embedding Trees [2] are two approaches that leverage the taxonomy structure while learning an embedding. Taxonomy Embedding jointly learns a *prototype* for each category such that nearby categories in the taxonomy have similar prototypes, while learning a single linear embedding that maps the data points close to their corresponding prototype. Label Tree Embedding, a slightly different approach, tries to learn the taxonomy itself from the empirical confusion matrix of a one-vs-rest classification to find a good embedding. Both these approaches are rather restrictive as they commit to a single metric. We, on the other hand, learn multiple metrics that are essentially parameterized by the taxonomy. The way we combine the metrics for the classification task make them more expressive than a single linear metric.

Some works go beyond the traditional classification task and demonstrate interesting applications using the taxonomy. Li *et al*. [14], for instance, learn a taxonomy from images and the associated tag information to perform classification, annotation, and automatic hierarchical organization of a photo collection. Deng *et al*. [5] use a taxonomy to learn an image similarity function for improved retrieval.

Salakhutdinov *et al*. [19] show that learning in a hierarchy can improve recognition performance on classes with a small number of training cases. Whereas, Zweig and Weinshall [23] train a recognition model on images for an object category close to a unseen object in a hierarchy and show improved accuracy for that new object given its correct location in the hierarchy. Here we explore a slightly different application where we *find* the correct placement of a related, but previously unseen, category in a given taxonomy.

3. Distance metric learning for taxonomies

Given an underlying class taxonomy, we associate a separate metric with each node. The individual metrics serve as *local viewpoints* for representing the data according to the different nodes. Having local representations of the data has two main advantages:

1. Having separate metrics for separate categories (leaf nodes) can significantly improve the classification accuracy compared to having just a single metric across categories [22].

2. In a hierarchical setting, local representations make it possible for the children to *share* the representation (or

the metric) of their parent. This sharing helps distribute the burden of category recognition in such a way that a metric associated with a node is mainly responsible for discriminating among the categories associated with its siblings and children. Hence, each metric is responsible to discriminate among only a few categories, making the overall classification easier.

Here we suggest two sharing mechanisms through which a metric associated with a parent can share information with the metrics associated with its children: (i) distance metric sharing, and (ii) transformation sharing.

Notation. For a given node t in the taxonomy, let p_t denote its parent, and A_t denote the set of nodes in the path from t to the root node (that is, the set of all its ancestors including itself). The variables $\mathbf{x}$ and y shall represent the D-dimensional feature vector and the corresponding class label respectively. We assume that the class labels are associated only with the leaf node categories in the taxonomy.

Distance metric sharing (DMS) model. In this model, we associate a local distance metric Q_t with each node, that is, each Q_t is a $D \times D$ positive semidefinite (PSD) matrix. In order to get a globally consistent node representation, these local metrics are combined together with their ancestor representations by simple aggregation. We define a node's *aggregate* metric as $\mathbf{Q}_t := \sum_{i \in A_t} Q_i$.

Such a combination implies that a child's aggregate distance metric can be written as the sum of its parent's aggregate metric and its own *local* metric, *i.e.* $\mathbf{Q}_t = \mathbf{Q}_{p_t} + Q_t$. As an example, in Figure 1 observe that nodes *Owl* and *Pigeon* have *Bird* as the parent. The corresponding metrics $\mathbf{Q}_{\text{owl}} = \mathbf{Q}_{\text{bird}} + Q_{\text{owl}}$ and $\mathbf{Q}_{\text{pigeon}} = \mathbf{Q}_{\text{bird}} + Q_{\text{pigeon}}$ share the parent's metric $\mathbf{Q}_{\text{bird}}$.

Note that such kind of sharing is meaningful since one can now define the distance with respect to a new test example $\mathbf{x}_{\text{t}}$ in a consistent way. Let $\mathbf{x}_{\text{o}}$ and $\mathbf{x}_{\text{p}}$ be examples belonging to categories *Owl* and *Pigeon* respectively. Then comparing the distances

$$\rho(\mathbf{x}_{\text{t}}, \mathbf{x}_{\text{o}}; \mathbf{Q}_{\text{owl}}) := (\mathbf{x}_{\text{t}} - \mathbf{x}_{\text{o}})^\top \mathbf{Q}_{\text{owl}} (\mathbf{x}_{\text{t}} - \mathbf{x}_{\text{o}}), \text{ and}$$
$$\rho(\mathbf{x}_{\text{t}}, \mathbf{x}_{\text{p}}; \mathbf{Q}_{\text{pigeon}}) := (\mathbf{x}_{\text{t}} - \mathbf{x}_{\text{p}})^\top \mathbf{Q}_{\text{pigeon}} (\mathbf{x}_{\text{t}} - \mathbf{x}_{\text{p}})$$

becomes a comparison in the respective local viewpoints and in the shared metric $\mathbf{Q}_{\text{bird}}$ at the parent level.

If one is interested in transforming the feature space, one can factorize these metrics to obtain node specific low-dimensional visualizations for the data.

Transformation sharing (TS) model. It is also possible to directly associate a local $d \times D$ *transformation* matrix L_t with each node. These matrices represent d-dimensional linear transformations of the data in the local viewpoint of the corresponding nodes. Sharing the representation across nodes can be done in a similar way by defining a node's *aggregate* transformation matrix $\mathbf{L}_t := \sum_{i \in A_t} L_i = \mathbf{L}_{p_t} + L_t$.

The corresponding distance metric can be compared using the quadratic form

$$\mathbf{L}_t^\top \mathbf{L}_t = L_t^\top L_t + \mathbf{L}_{p_t}^\top \mathbf{L}_{p_t} + \mathbf{L}_{p_t}^\top L_t + L_t^\top \mathbf{L}_{p_t} .$$

Comparison between the two models. Though the TS model has an advantage of explicitly incorporating low-dimensional representations (by the choice of d), it is generally unclear how to pick d for a given application. Moreover, even if one knows the right d, explicitly optimizing the objective function for that d is hard.

If one enforces orthogonality between $\mathbf{L}_t$ and its parent $\mathbf{L}_{p_t}$ then the cross terms in the quadratic form $\mathbf{L}_t^\top \mathbf{L}_t$ disappear, making it equivalent to the DMS model. Due to ease of implementation, we shall use the DMS model for our experiments.

3.1. Learning and inference

We present a novel discriminative probabilistic nearest neighbor classification approach to learn in the 'distance metric sharing' model.

First define the distance between any two input datapoints $\mathbf{x}_1$ and $\mathbf{x}_2$ with respect to the distance metric $\mathbf{Q}_t$ as $\rho(\mathbf{x}_1, \mathbf{x}_2; \mathbf{Q}_t) := (\mathbf{x}_1 - \mathbf{x}_2)^\top \mathbf{Q}_t (\mathbf{x}_1 - \mathbf{x}_2)$. Now given an input query $\mathbf{x}$, we define the *score* $f(\mathbf{x}; y)$ with respect to an arbitrary class y, by combining the distances between $\mathbf{x}$ and the closest examples to $\mathbf{x}$ in the class y (in the metric associated with class y), that is, $f(\mathbf{x}; y) := \sum_{\tilde{\mathbf{x}} \in \mathcal{N}_y(\mathbf{x})} \rho(\mathbf{x}, \tilde{\mathbf{x}}; \mathbf{Q}_y)$, where $\mathcal{N}_y(\mathbf{x})$ denotes the set of k closest neighbors in the class y. This score gives an estimate of how *close* a given example is to the examples of class y. Using the scores for all the classes, we can define the class probability distribution of the example $\mathbf{x}$ as:

$$p(y|\,\mathbf{x}; \mathbb{Q}) := \frac{\exp(-f(\mathbf{x}; y))}{\sum_{\bar{y}} \exp(-f(\mathbf{x}, \bar{y}))}, \qquad (1)$$

where $\mathbb{Q}$ is simply the set of all the distance metrics $\{Q_t\}$.

Now, given the training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, we learn the distance metrics by maximizing a regularized likelihood function:

$$\mathcal{L}(\mathbb{Q}) = \frac{1}{n} \sum_{i=1}^{n} \log p(y_i|\,\mathbf{x}_i; \mathbb{Q}) - \frac{\lambda}{2} \sum_t \text{trace}(Q_t^\top Q_t) \quad (2)$$

subject to positive semidefinite (PSD) constraints on the local distance metrics, $Q_t \succeq 0$. Here λ is the regularization constant. Note that satisfying the PSD constraints on the local Q_t's makes the aggregate metrics $\mathbf{Q}_t$'s PSD as well.

Observe that the equivalent minimization problem along with the PSD constraints is a convex optimization problem and can be solved efficiently.

We explain the role of the two terms in the optimization. For the first term, observe that Eq. (1) can be rewritten as:

$$p(y|\,\mathbf{x};\mathbb{Q}) = \frac{1}{1+\sum_{\bar{y}\neq y}\exp\big(f(\mathbf{x},y)-f(\mathbf{x},\bar{y})\big)}.$$

Hence, the likelihood is maximized by (i) minimizing $f(\mathbf{x}_i, y_i)$, and (ii) maximizing $f(\mathbf{x}_i, \bar{y})$ (for $\bar{y} \neq y_i$) for each training point $\mathbf{x}_i$. Geometrically these terms can be thought as *pulling together* the same class neighbors, while *pushing away* different class neighbors. (In this sense, our optimization can be thought as an efficient hierarchical generalization of LMNN [22] or NCA [8].)

The regularization term in the likelihood can be written as: $\text{trace}(Q_t^{\mathsf{T}}Q_t) = \text{trace}\big((\mathbf{Q}_{p_t}-\mathbf{Q}_t)^{\mathsf{T}}(\mathbf{Q}_{p_t}-\mathbf{Q}_t)\big)$. As a result, it attempts to keep the distance metrics of the children $\mathbf{Q}_t$ to be closer to that of the parent $\mathbf{Q}_{p_t}$.

Incorporating context sensitive loss. Sometimes it is favorable to penalize misclassification between different categories differently. Consider, for instance, mispredicting an image of *Wolf* as a *Horse* versus mispredicting it as a *Pigeon* (*c.f.* Figure 1). The latter misclassification seems more severe and our optimization should try to minimize it.

Such a requirement can be addressed by using a *context sensitive loss* (CSL) function[1]: $\Delta(y,\bar{y}) \geq 0$ (with the requirement that $\Delta(y,y) = 0$, for all y) denoting the penalty assigned to predict $\bar{y}$ when the true label was y. We can easily incorporate this in our formulation by redefining our probability distribution during training (*c.f.* Eq. (1)) as

$$p_{\text{csl}}(y|\,\mathbf{x};\mathbb{Q}) \propto \frac{\exp(-f(\mathbf{x};y))}{\sum_{\bar{y}}\exp(-f(\mathbf{x},\bar{y})+\Delta(y,\bar{y}))}, \tag{3}$$

Observe that this has the effect of *pushing* the neighbors associated with severely penalized categories farther than those with less penalty.

Optimization details. For experiments, we optimize our objective function as shown in Algorithm 1. We used a simple gradient ascent procedure with an *adaptive* step-size: we start by setting the step (η) as 0.001 and decrease it by half whenever the function value decreases (worsens), or increase it by 1.01 if the function value increases (improves). The function value was computed after the PSD projection.

We chose the number of neighbors (k) as 5. We found that varying k did not have a significant effect on the performance. We also found that running the optimization of about four rounds (R) was sufficient.

3.2. Using the learned metrics

We can use the trained set of metrics $\mathbb{Q}$ for the example applications (as discussed in the Introduction) as follows:

[1]There are several ways to define a CSL. One possibility is to define the loss as the shortest path distance between the two classes. Another possibility is to define it using the tree height difference.

Algorithm 1 Hierarchical Metric Learning

```
Input: The training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$,
       Taxonomy $\mathcal{T}$ with total $T$ nodes, and
       $C$ is the set of leaf nodes from $\mathcal{T}$.
       Parameters: $R$ – max. number of rounds,
                   $k$ – number of nearest neighbors,
                   $\eta$ – gradient step-size.
 1: for $t = 1$ to $T$ do
 2:   Initialize $Q_t := \begin{cases} I & \text{for } t \text{ corresponding to leaf node} \\ \mathbf{0} & \text{otherwise} \end{cases}$
 3: end for
 4: Define $\mathbf{Q}_t := \sum_{i\in A_t} Q_i$, for $t = 1, \ldots, T$.
 5: for $r = 1$ to $R$ do
 6:   for $i = 1$ to $n$, and $y \in C$ do
 7:     Compute the set $\mathcal{N}_y(\mathbf{x}_i)$ as the $k$ closest neighbors to the
        point $\mathbf{x}_i$ in the class $y$ using the metric $\mathbf{Q}_y$.
 8:   end for
 9:   repeat
10:     for $t = 1$ to $T$ do
11:       $Q_t = Q_t + \eta\,\frac{\partial\mathcal{L}(\mathbb{Q})}{\partial Q_t}$
12:       $Q_t = \text{Project\_to\_PSD}(Q_t)$
13:     end for
14:   until convergence
15: end for
Output: The learned metrics $Q_1, \ldots, Q_T$.
```

Categorizing new data. Given a test example $\mathbf{x}_t$, we can predict its most likely class in a straightforward way. We simply predict the most likely class as $y^* := \text{argmax}_y\, p(y|\,\mathbf{x}_t;\mathbb{Q}) = \text{argmin}_y\, f(\mathbf{x}_t; y)$.

Placing unseen categories in the taxonomy. Consider a scenario where one receives a collection of examples from an unseen category whose location is not known in the taxonomy. We assume that this new category is related to the taxonomy, but its placement is unknown. Our goal is to systematically identify its correct placement or suggest a likely set of candidate locations in the taxonomy.

We predict the likely location for the new category via a simple discrimination centric approach. We predict the class label for each new example according to the categories in the current taxonomy. Then, we find the majority class, and place the *new class* as a sibling to the majority class.

The basic intuition behind this procedure is that our metrics are tuned towards localizing the discrimination task towards the siblings and parents of a category (see our discussion in Section 4.2.2). Then, given that the examples belonging to the new category share similar discriminating features as its true siblings, our metrics would be able to pick-up on them.

Local subtree classification. In this example application, we are considering a scenario where the end goal is to get good discrimination amongst a *subset* of categories of all the categories in a taxonomy. Since, typically, it is hardest to discriminate amongst the neighboring categories (as the corresponding images share several features), we focus on categories that are part of a *subtree* of the taxonomy. To get the best possible set of metrics for discrimination, one should ideally train on the subtree taxonomy. However, this is cumbersome since one has to train the metrics repeatedly for each user for their specific subset categorization.

In our experiments, we show that the metrics learned on the full taxonomy would yield performance similar to the metrics that have been specifically trained on a subtree in recognizing the categories from that subtree.

4. Experiments

We now assess the effectiveness of our hierarchical metric model on a wide array of image datasets. We show that our learned metrics consistently yield better classification results compared to several state-of-the-art classification techniques across all datasets. An empirical analysis of our learned metrics elucidates several interesting observations. Lastly, we demonstrate that our metrics are useful for the taxonomy specific applications as well.

4.1. Setup

4.1.1 Datasets

ImageNet subtrees. ImageNet consists of images collected from the Web that are organized according to the WordNet hierarchy [6]. We report results on nine subtrees of ImageNet that are mentioned in [6]. The following subtrees were used: Amphibian, Fish, Fruit, Furniture, Geo (geological formation), Music (musical instrument), Reptile, Tool and Vehicle. The training, validation and test sets for these subtrees were taken from the 2010 ImageNet Large Scale Visual Recognition Challenge dataset[2], which is a subset of the ImageNet database. The subtrees vary in training set sizes (8,800 to 54,000), height (3 to 6), and the number of classes (8 to 40).

We used the SIFT-based bag-of-words representation given as part of the Challenge dataset. The vocabulary size is 1000, so each image is represented as a 1000 dimensional word count vector. We reduced the dimensionality to 250 with PCA, and normalized each vector to unit L_2-norm.

Animals with Attributes (AwA)[3]. This dataset contains images of 50 animal classes, but without a taxonomy. 17 of the 50 classes in AwA are present in the Mammals subtree of ImageNet. We use the part of the Mammals subtree that contains the AwA classes as the hierarchy for the 17 classes, as shown in Figure 4. We use the bag-of-words representation based on color-SIFT features supplied with the dataset. The original dimensionality of this representation is 2000, which we reduced to 500 dimensions with PCA, and then normalize to unit L_2-norm. The dataset is split into approx. 6,700 training, 1,000 validation, and 2,600 test cases.

4.1.2 Methods for comparison

We compare the performance of our method with state-of-the-art distance metric learning methods for classification. We take Large Margin Nearest Neighbor (LMNN) [22] and Taxonomy Embedding (TAXEMB) [21] methods for comparison. In LMNN, a linear transformation is learned such that the k nearest neighbor classification in the embedded space is improved. It does not use a taxonomy to learn the embedding. At test time, the class label of a new point is predicted using k nearest neighbor classification. In TAXEMB, two sets of parameters are learned: (i) a *prototype* vector for each class such that classes that are nearby in the taxonomy have similar prototypes, and (ii) a linear embedding such that examples belonging to the same class are placed close to its prototype. At test time, the classification is done by embedding the test point and selecting the nearest prototype's label.

Since SVMs are widely used for classification, we also compared with non-linear support vector machines (NLSVM). We fixed the degree of the polynomial kernel from the range $[4, 9]$ and the regularization constant from the range $[10^{-4}, 10^4]$, using the validation data.

As a baseline, we took the untrained version of our model with *trivial* initialization: Euclidean metric for the root node (*i.e.*, the identity matrix) and the zero metric for rest of the nodes (*i.e.*, the zero matrix). This is equivalent to representing the data in the standard Euclidean space. We used a simple variant of nearest neighbor classifier, where instead of picking the majority class of k closest neighbors, we pick k closest neighbors from each class. We sum these distances and pick the class with the smallest total distance. We call this variant as the aggregate nearest neighbor (AGGKNN)[4]. Since this baseline uses the Euclidean metric, it would help demonstrate the importance of *learning* the metrics.

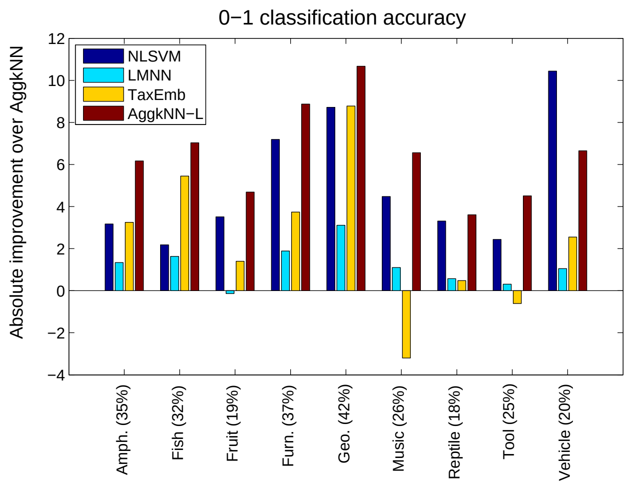
We used two main variants of our *hierarchical distance metrics learning* method. The first method directly optimizes Eq. (2); we refer to this method as AGGKNN-L. The second optimizes the CSL variant of Eq. (2) (details below); we refer to this method as AGGKNN-L-CSL.

4.1.3 Evaluation metrics

We report the classification accuracy in the categorization experiments. The *accuracy* function is defined as: $1 - \frac{1}{mH}\sum_{t=1}^{m} \Delta(y_t, \hat{y}_t)$ where y_t and $\hat{y}_t$ denote the true

[2]www.image-net.org/challenges/LSVRC/2010/
[3]http://attributes.kyb.tuebingen.mpg.de/

[4]We found that this simple variant consistently yields better performance over regular k-NN.



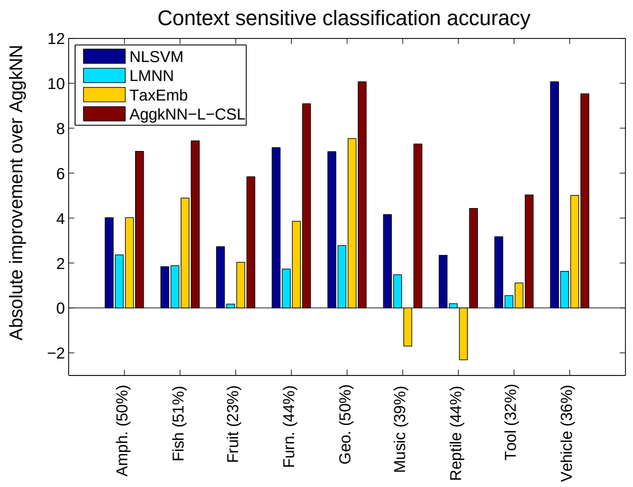


Figure 2. Classification accuracy for the various datasets. The horizontal axis shows the dataset name along with the absolute accuracy achieved by AGGKNN method. The vertical axis shows the absolute improvement gained over AGGKNN. **Left:** The standard 0-1 classification accuracy. **Right:** Context sensitive accuracy using the common ancestor CSL.

and predicted class labels of the t^{th} test example; m denotes the total number of test examples. Each correct prediction has zero loss (*i.e.*, $\Delta(y_t, \hat{y}_t) = 0$ when $\hat{y}_t = y_t$).

We evaluate the performance on two measures of accuracy (each measure treats the misclassification differently):

1. The *conventional* accuracy measure (also referred to as 0-1 accuracy) treats each misclassification equally (so $\Delta(y_t, \hat{y}_t) = 1$ when $\hat{y}_t \neq y_t$ and $H = 1$).

2. The *context sensitive* accuracy measure treats each misclassification according to the CSL function. We use $\Delta(y_t, \hat{y}_t)$ as the height of the lowest common ancestor for the pair $(y_t, \hat{y}_t)$ when $\hat{y}_t \neq y_t$. H is set as the maximum tree height to normalize the misclassification error to lie in $(0, 1]$.

4.2. Results

4.2.1 Classification

Figure 2 (Left) shows 0-1 accuracy results for the different datasets using various classification methodologies. Observe that LMNN performs only slightly better than our baseline AGGKNN method. Except on Music and Tool datasets, TAXEMB performs better than LMNN. The performance improvement on the Geo and Fish datasets is significant (4%-6%), indicating that using hierarchical information in training is useful for 0-1 classification. TAXEMB, however, performs significantly worse than our AGGKNN-L method, especially on the larger datasets. This is expected since learning a single prototype per class may not be sufficient when classifying among large number of classes. A comparison with NLSVM shows that our method typically yields better results than a strong discriminative classifier. Our method outperforms all the other methods on all the datasets except Vehicle where it is the second best. The performance difference between the AGGKNN-L and AGGKNN is significant on all the datasets, emphasizing the need for learning the metrics.

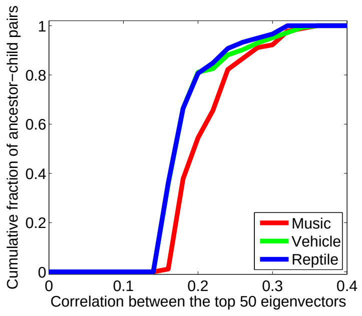
Figure 2 (Right) shows the context sensitive accuracy results with our AGGKNN-L-CSL method. Observe that TAXEMB works well for CSL. We can make similar observations regarding the benefits of learning hierarchical metrics. We also compared our AGGKNN-L and AGGKNN-L-CSL methods. AGGKNN-L typically performs similar (sometimes slightly worse) to AGGKNN-L-CSL across the datasets. We believe that this is because AGGKNN-L already uses the hierarchical information effectively and does not need an explicit taxonomy based loss.

4.2.2 Analysis of the learned metrics

We now do a detailed analysis of the learned metrics on two properties that we believe are the key for their success: (i) orthogonality, and (ii) localized discrimination.

Orthogonality. We believe that one of the reasons for good performance of our metrics is: the individual local metrics *share* the burden of classification in such a way that different metrics focus on classifying amongst different subsets of categories. We can characterize this by stating that different metrics emphasize on *different sets of features* for the discrimination. Equivalently, one can say that top principal components of different metrics are approximately *orthogonal* to each other.

To verify this empirically, we chose different pairs of nodes on a path from root to some leaf node. We computed a correlation score for each such pair by averaging



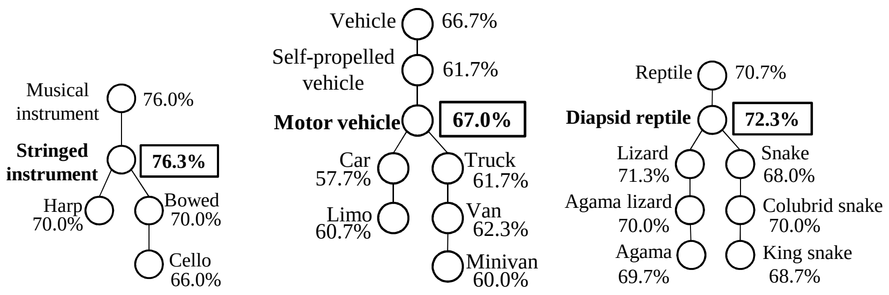


Figure 3. Analysis of the learned metrics. **Left:** A plot showing how correlated the top eigenvectors of metrics are. **Right:** Ancestor path of some non-sibling pairs of nodes in Music, Vehicle, and Reptile. The metric associated with the lowest common ancestor yields the best classification result when discriminating between the pairs.

the dot product between the most correlated eigenvectors from the top 50 eigenvectors. Figure 3 (Left) shows the cumulative distribution of the correlation score of different pairs for Music, Vehicle and Reptile datasets. Note that over 90% of the pairs have correlation less than 0.3. This shows that eigenvectors, and hence the local metrics of the nodes along a path, are focused on *different sets of features* – thereby sharing the overall classification burden.

Localized discrimination. The second key reason for good performance of our metrics is that each metric is specialized to discriminate between categories belonging to its children.

We can see this by measuring the classification accuracy using an individual metric associated with a node (instead of aggregating it along a path). Figure 3 (Right) shows the 0-1 classification accuracy on two non-sibling categories by using individual local metrics along the path for Music (*Harp* and *Cello*), Vehicle (*Limo* and *Minivan*) and Reptile (*Agama* and *King snake*). Observe that the nodes most helpful in classification are *String Instrument*, *Motor-vehicle* and *Diapsid reptile* respectively, which also happen to be the lowest common ancestors for the selected nodes. There were similar results for other pairs of classes, and datasets. This shows that the least common ancestor plays a major role in discriminating between its children categories.

These properties together make our methodology powerful: the learned metrics share the classification burden in a locally consistent manner.

4.2.3 Placing unseen categories in taxonomy

We conducted systematic experiments with Music, Furniture and Reptile, wherein (i) we removed data belonging to a category from the taxonomy, (ii) learn the metrics, and (iii) find the "class" placement using the learned metrics in the taxonomy. We did this experiment with four different classes (removing them one at a time) for each dataset. Our method identified the correct location in 50% cases, and one level up in 25% cases. For the remaining cases (one per dataset) the confusion was with semantically similar objects: *Wardrobe* got confused with *Chinese cabinet* in Furniture, or visually similar objects: *Snake* got confused with *Lizard* in Reptile.

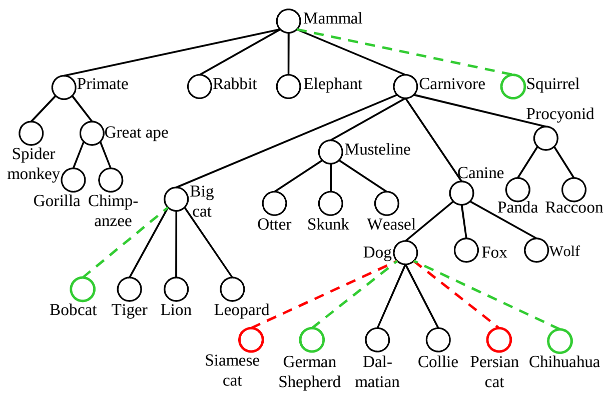
As an example application, we mapped the categories available in Animals with Attribute (AwA) dataset [13] (a dataset that does not have a pre-specified taxonomy) to the taxonomy structure available for Mammals from ImageNet. Note that only some (17 of 50) categories of AwA can be easily mapped, others do not have a clear placement. Figure 4 (Left) depicts the 17 classes along with the derived taxonomy structure in solid lines. We trained our metrics on these 17 classes. We then used our proposed procedure to place six new classes: *Bobcat*, *German Shepherd*, *Siamese Cat*, *Squirrel*, *Chihuahua*, and *Persian Cat* (these were chosen due to their semantic or visual similarity to the existing nodes at different parts of the taxonomy).

Figure 4 (Left) shows the proposed locations of these previously unseen categories in dashed lines. The green dashed lines show the categories there were placed correctly – *Bobcat*, *German Shepherd*, *Chihuahua* and *Squirrel*. The class *Squirrel* was assigned to root due to its visual similarity with *Rabbit* and background mostly consisting of gardens and parks. See Figure 4 (Right).

The red dashed lines show the categories that were placed incorrectly – *Persian Cat* and *Siamese Cat*. These categories seemed to be placed under the *Dog* node as they resemble more with *Collie* than *Leopard*, *Tiger* or *Lion*. See Figure 4 (Right). These results suggest that our learned metrics do a reasonable job in predicting the locations for unseen categories and can assist an expert in enriching existing taxonomies with new categories.

4.2.4 Local subtree classification

To demonstrate the effectiveness of our learned set of metrics for a specialized task, we considered two subtrees each from Music and Tool taxonomies. The subtrees for the Music dataset are rooted at *Stringed Instrument* (with depth 3, and 6 leaf nodes) and *Wood-wind Instrument* (with depth



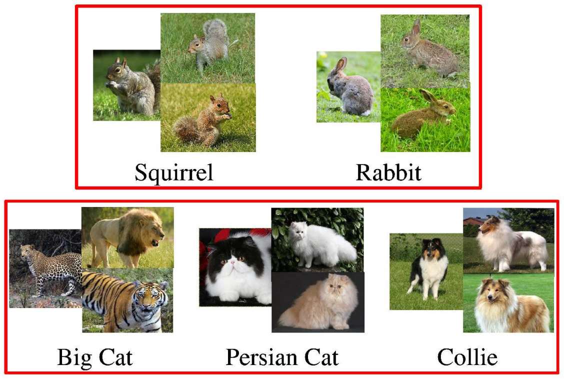


Figure 4. **Left:** The 17 categories of AwA dataset mapped onto the Mammals taxonomy. The original 17 categories are connected with solid lines. The nodes connected with dashed lines show the predicted locations of entirely new categories in the taxonomy. Nodes shaded in green indicate the *correct* placement of the corresponding categories, while the nodes shaded in red indicate an *incorrect* placement. **Right:** Example images from *Squirrel* (top) and *Persian cat* (bottom). The *Squirrel* category is visually similar to the *Rabbit* category. The *Persian cat* is visually similar to *Collie* than to the categories that are part of *Big cat*.

4, and 4 leaf nodes). Similarly, the subtrees chosen for Tool dataset were *Cutting Tool* (with depth 3, and 8 leaf nodes), and *Opener* (with depth 3, and 3 leaf nodes).

Metric Used	*String Instrument*	*Wood-wind Instrument*	*Opener*	*Cutting Tool*
Specialized	62.70	58.66	63.20	57.10
General	60.40	55.00	59.60	55.30

Table 1. Local classification accuracy (%) results on subtrees of Music and Tool datasets.

Table 1 shows how well our learned metrics (on the full taxonomy) fare when compared with learning a separate special set for each subtree. We show the 0-1 classification accuracy achieved when classifying amongst the categories local to the subtrees. Note that in each case, there was only a small degradation in accuracy (2.8% on average) when using the general set of metrics.

5. Conclusion

We presented a novel framework that leverages the taxonomy to learn a set of hierarchical similarity metrics. Our metrics yield improved classification performance over several methods available in the literature. Since our metrics are taxonomy centric, they also benefit some interesting taxonomy specific applications. A fruitful future direction is to incorporate unlabelled data for metric learning and taxonomy enrichment.

Acknowledgements. The authors would like to thank Kai Wang for helpful discussions.

References

[1] B. Babenko, S. Branson, and S. Belongie. Similarity metrics for categorization: from mono. to category specific. *ECCV*, 2009. 1, 2

[2] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. *NIPS*, 2010. 2

[3] A. Beygelzimer, J. Langford, Y. Lifshits, G. Sorkin, and A. Strehl. Cond. prob. tree estimation analysis and algorithms. *UAI*, 2009. 2

[4] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. *CIKM*, 2004. 2

[5] J. Deng, A. Berg, and L. Fei-Fei. Hierarchical semantic indexing for large scale image retrieval. *CVPR*, 2011. 2

[6] J. Deng, W. Dong, R. Socher, L. Li, K. Ki, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. *CVPR*, 2009. 2, 5

[7] R. Fergus, H. Bernal, Y. Weiss, and A. Torralba. Semantic label sharing for learning with many categories. *ECCV*, 2010. 2

[8] A. Globerson and S. T. Roweis. Metric learning by collapsing classes. *NIPS*, 2005. 2, 4

[9] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. *NIPS*, 2004. 2

[10] G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. *CVPR*, 2008. 1, 2

[11] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. *CVPR*, 2006. 1

[12] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. *CVPR*, 2008. 2

[13] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen classes by between-class attrib. transfer. *CVPR*, 2009. 7

[14] L. Li, C. Wang, Y. Lim, D. Blei, and L. Fei-Fei. Building and using a semantivisual image hierarchy. *CVPR*, 2010. 2

[15] M. Marszałek and C. Schmid. Semantic hierarchies for visual object recognition. *CVPR*, 2007. 1, 2

[16] M. Marszałek and C. Schmid. Constructing category hierarchies for visual recognition. *ECCV*, 2008. 1, 2

[17] B. Mcfee and G. Lanckriet. Metric learning to rank. *ICML*, 2010. 1

[18] D. Ramanan and S. Baker. Local distance functions: A taxonomy, new algorithms, and an evaluation. *ICCV*, 2009. 2

[19] R. Salakhutdinov, A. Torralba, and J. Tenenbaum. Learning to share visual appearance for multiclass object detection. *CVPR*, 2011. 2

[20] G. Shakhnarovich. *Learning Task-Specific Similarity*. PhD thesis, Massachusetts Institute of Technology, 2006. 1

[21] K. Weinberger and O. Chapelle. Large margin taxonomy embedding with an application to document categorization. *NIPS*, 2009. 2, 5

[22] K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 2009. 1, 2, 4, 5

[23] A. Zweig and D. Weinshall. Exploiting object hierarchy: Combining models from different category levels. *ICCV*, 2007. 2