Deep Metric Learning via Lifted Structured Feature Embedding

Hyun Oh Song
Stanford University
hsong@cs.stanford.edu

Yu Xiang
Stanford University
yuxiang@cs.stanford.edu

Stefanie Jegelka
MIT
stefje@csail.mit.edu

Silvio Savarese
Stanford University
ssilvio@stanford.edu

Abstract

Learning the distance metric between pairs of examples is of great importance for learning and visual recognition. With the remarkable success from the state of the art convolutional neural networks, recent works [1, 31] have shown promising results on discriminatively training the networks to learn semantic feature embeddings where similar examples are mapped close to each other and dissimilar examples are mapped farther apart. In this paper, we describe an algorithm for taking full advantage of the training batches in the neural network training by lifting the vector of pairwise distances within the batch to the matrix of pairwise distances. This step enables the algorithm to learn the state of the art feature embedding by optimizing a novel structured prediction objective on the lifted problem. Additionally, we collected Online Products dataset: 120k images of 23k classes of online products for metric learning. Our experiments on the CUB-200-2011 [37], CARS196 [19], and Online Products datasets demonstrate significant improvement over existing deep feature embedding methods on all experimented embedding sizes with the GoogLeNet [33] network.

1. Introduction

Comparing and measuring similarities between pairs of examples is a core requirement for learning and visual competence. Being able to first measure how similar a given pair of examples are makes the following learning problems a lot simpler. Given such a similarity function, classification tasks could be simply reduced to the nearest neighbor problem with the given similarity measure, and clustering tasks would be made easier given the similarity matrix. In this regard, metric learning [13, 39, 34] and dimensionality reduction [18, 7, 29, 2] techniques aim at learning semantic distance measures and embeddings such that similar input objects are mapped to nearby points on a manifold and dissimilar objects are mapped apart from each other.

Furthermore, the problem of extreme classification [6, 26] with enormous number of categories has recently attracted a lot of attention in the learning community. In this setting, two major problems arise which renders conventional classification approaches practically obsolete. First, algorithms with the learning and inference complexity linear in the number of classes become impractical. Second, the availability of training data per class becomes very scarce. In contrast to conventional classification approaches, metric learning becomes a very appealing technique in this regime because of its ability to learn the general concept of distance metrics (as opposed to category specific concepts) and its compatibility with efficient nearest neighbor inference on the learned metric space.

With the remarkable success from the state of the art convolutional neural networks [20, 33], recent works [1, 31] discriminatively train neural network to directly learn the non-linear mapping function from the input image to a lower dimensional embedding given the input label annotations. In high level, these embeddings are optimized to pull examples with different class labels apart from each other and push examples from the same classes closer to each other. One of the main advantages of these discriminatively trained network models is that the network jointly learns the feature representation and semantically meaningful embed-
Deep feature embedding with state of the art convolutional neural networks: Bell et al. [1] learn embedding for visual search in interior design using contrastive [14] embedding. FaceNet [31] uses triplet [39] embedding to learn embedding on faces for face verification and recognition. Li et al. [22] learn a joint embedding shared by both 3D shapes and 2D images of objects. In contrast to the existing approaches above, our method computes a novel structured loss and the gradient on the lifted dense pairwise distance matrix to take full advantage of batches in SGD.

Zero shot learning and ranking: Frome et al., Socher et al., and Weston et al. [12, 32, 40] leverage text data to train visual ranking models and to constrain the visual predictions for zero shot learning. Wang et al. [38] learns to rank input triplet of data given human rater’s rank ratings on each triplets and also released a triplet ranking dataset with 5,033 triplet examples [8]. However, the approach is not scalable with the size of the training data because it’s very costly to obtain ranking annotations in contrast to multi-class labels (i.e., product name) and because the approach is limited to ranking the data in triplet form. Lampert et al. [21] does zero shot learning but with attributes (such as object’s color or shape) provided for both the train and the test data. On a related note, [24, 25, 28] do zero-shot learning for visual recognition but rely on the WordNet hierarchy for semantic information of the labels.

The paper is organized as follows. In section 3, we start with a brief review of recent state of the art deep learning based embedding methods [14, 31]. In section 4, we describe how we lift the problem and define a novel structured loss. In section 5 and 6, we describe the implementation details and the evaluation metrics. We present the experimental results and visualizations in section 7.

3. Review

In this section, we briefly review recent works on discriminatively training neural networks to learn semantic embeddings.

Contrastive embedding [14] is trained on the paired data \( \{(x_i, x_j, y_{ij})\} \). Intuitively, the contrastive training minimizes the distance between a pair of examples with the same class label and penalizes the negative pair distances for being smaller than the margin parameter \( \alpha \). Concretely, the cost function is defined as,

\[
J = \frac{1}{m} \sum_{(i,j)} \frac{m/2}{y_{ij}} D_{i,j}^2 + (1 - y_{ij}) [\alpha - D_{i,j}]_+^2 , \tag{1}
\]

where \( m \) stands for the number of images in the batch, \( f(\cdot) \) is the feature embedding output from the network, \( D_{i,j} = ||f(x_i) - f(x_j)||_2 \), and the label \( y_{ij} \in \{0, 1\} \) indicates
whether a pair \((x_i, x_j)\) is from the same class or not. The \([\cdot]_+\) operation indicates the hinge function \(\max(0, \cdot)\). We direct the interested readers to refer [14, 1] for the details.

**Triplet embedding** [39, 31] is trained on the triplet data \(\{x_a^{(i)}, x_p^{(i)}, x_n^{(i)}\}\) where \(x_a^{(i)}, x_p^{(i)}\) have the same class labels and \(x_a^{(i)}, x_n^{(i)}\) have different class labels. The \(x_a^{(i)}\) term is referred to as an anchor of a triplet. Intuitively, the training process encourages the network to find an embedding where the distance between \(x_a^{(i)}\) and \(x_n^{(i)}\) is larger than the distance between \(x_a^{(i)}\) and \(x_p^{(i)}\) plus the margin parameter \(\alpha\). The cost function is defined as,

\[
J = \frac{3}{2m} \sum_i \left[ D_{i,a,i,p}^2 - D_{i,a,i,n}^2 + \alpha \right]_+ ,
\]

where \(D_{i,a,i,p} = \|f(x_a^{(i)}) - f(x_p^{(i)})\|\) and \(D_{i,a,i,n} = \|f(x_a^{(i)}) - f(x_n^{(i)})\|\). Please refer to [31, 39] for the complete details.

![Contrastive embedding](image1)

(a) Contrastive embedding

![Triplet embedding](image2)

(b) Triplet embedding

![Lifted structured embedding](image3)

(c) Lifted structured embedding

Figure 2: Illustration for a training batch with six examples. Red edges and blue edges represent similar and dissimilar examples respectively. In contrast, our method explicitly takes into account all pair wise edges within the batch.

**4. Deep metric learning via lifted structured feature embedding**

We define a structured loss function based on all positive and negative pairs of samples in the training set:

\[
J = \frac{1}{2|\hat{P}|} \sum_{(i,j) \in \hat{P}} \max(0, J_{i,j})^2 ,
\]

\[
J_{i,j} = \max \left( \max_{(i,k) \in \hat{N}} \alpha - D_{i,k}, \max_{(j,l) \in \hat{N}} \alpha - D_{j,l} \right) + D_{i,j}
\]

where \(\hat{P}\) is the set of positive pairs and \(\hat{N}\) is the set of negative pairs in the training set. This function poses two computational challenges: (1) it is non-smooth, and (2) both evaluating it and computing the subgradient requires mining all pairs of examples several times.

We address these challenges in two ways: First, we optimize a smooth upper bound on the function instead. Second, as is common for large data sets, we use a stochastic approach. However, while previous work implements a stochastic gradient descent by drawing pairs or triplets of points uniformly at random [14, 1, 22], our approach deviates from those methods in two ways: (1) it biases the sample towards including “difficult” pairs, just like a subgradient of \(J_{i,j}\) would use the close negative pairs \(^1\); (2) it makes use of the full information of the mini-batch that

\(^1\)Strictly speaking, this would be a subgradient replacing the nested max by a plus.
is sampled at a time, and not only the individual pairs.

Figures 2a and 2b illustrate a sample batch of size $m = 6$ for the contrastive and triplet embedding. Red edges in the illustration represent positive pairs (same class) and the blue edges represent negative pairs (different class) in the batch. In this illustration, it is important to note that adding extra vertices to the graph is a lot more costly than adding extra edges because adding vertices to the graph incurs extra I/O time and/or storage overhead.

To make full use of the batch, one key idea is to enhance the mini-batch optimization to use all $O(m^2)$ pairs in the batch, instead of $O(m)$ separate pairs. Figure 2c illustrates the concept of of transforming a training batch of examples to a fully connected dense matrix of pairwise distances. Given a batch of $c$-dimensional embedded features $X \in \mathbb{R}^{m \times c}$ and the column vector of squared norm of individual batch elements $\tilde{x} = [\|f(x_1)\|^2, \ldots, \|f(x_m)\|^2]^T$, the dense pairwise squared distance matrix can be efficiently constructed by computing, $D^2 = \tilde{x}1^T + 1\tilde{x}^T - 2XX^T$, where $D^2_{ij} = \|f(x_i) - f(x_j)\|^2$. However, it is important to note that the negative edges induced between randomly sampled pairs carry limited information. Most likely, they are different from the much sharper, close (“difficult”) neighbors that a full subgradient method would focus on.

Hence, we change our batch to be not completely random, but integrate elements of importance sampling. We sample a few positive pairs at random, and then actively add their difficult neighbors to the training mini-batch. This augmentation adds relevant information that a subgradient would use. Figure 4 illustrates the mining process for one positive pair in the batch, where for each image in a positive pair we find its close (hard) negative images. Note that our method allows mining the hard negatives from both the left and right image of a pair in contrast to the rigid triplet structure [31] where the negative is defined only with respect to the predefined anchor point. Indeed, the procedure of mining hard negative edges is equivalent to computing the loss augmented inference in structured prediction setting [35, 17]. Our loss augmented inference can be efficiently processed by first precomputing the pairwise batch squared distance matrix $D^2$. Figure 3 presents the comprehensive visual comparison of different training structures (i.e. batch, label, network layout) with different loss functions. In contrast to other approaches (Fig. 3a and 3b), our method greatly simplifies the network structure (Fig. 3c) and requires only one branch of the CNN.

Furthermore, mining the single hardest negative with nested max functions (eqn. 4) in practice causes the network to converge to a bad local optimum. Hence we opti-
derived as shown in algorithm 1, where the gradients with respect to the distances are,

\[
\frac{\partial J_i}{\partial D_{i,j}} = \frac{1}{|P|} \mathbb{1}[J_{i,j} > 0] \frac{\exp(\alpha - D_{i,j})}{\exp(J_{i,j} - D_{i,j})} \tag{5}
\]

\[
\frac{\partial J_i}{\partial D_{i,k}} = \frac{1}{|P|} \mathbb{1}[J_{i,j} > 0] \frac{-\exp(\alpha - D_{i,k})}{\exp(J_{i,j} - D_{i,k})} \tag{6}
\]

\[
\frac{\partial J_i}{\partial D_{j,i}} = \frac{1}{|P|} \mathbb{1}[J_{i,j} > 0] \frac{-\exp(\alpha - D_{j,i})}{\exp(J_{i,j} - D_{j,i})}, \tag{7}
\]

where \(\mathbb{1}[\cdot]\) is the indicator function which outputs 1 if the expression evaluates to true and outputs 0 otherwise. As shown in algorithm 1 and equations 5, 6, and 7, our method provides informative gradient signals for all negative pairs as long as they are within the margin of any positive pairs (in contrast to only updating the hardest negative) which makes the optimization much more stable.

Having stated the formal objective, we now illustrate and discuss some of the failure modes of the contrastive [14] and triplet [31, 39] embedding in which the proposed embedding learns successfully. Figure 5 illustrates the failure cases in 2D with examples from three different classes. Contrastive embedding (Fig. 5a) can fail if the randomly sampled negative \(x_j\) is collinear with the examples from another class (purple examples in the figure). Triplet embedding (Fig. 5b) can also fail if such sampled negative \(x_n\) is within the margin bound with respect to the sampled positive example \(x_i\) and the anchor \(x_a\). In this case, both contrastive and triplet embedding incorrectly pushes the positive \((x_i/x_a)\) towards the cluster of examples from the third class. However, in the proposed embedding (Fig. 5c), given sufficiently large random samples \(m\), the hard negative examples \((x_k\)’s in Fig. 5c) within the margin bound pushes the positive \(x_i\) towards the correct direction.

5. Implementation details

We used the Caffe [16] package for training and testing the embedding with contrastive [14], triplet [31, 39], and ours. Maximum training iteration was set to 20,000 for all the experiments. The margin parameter \(\alpha\) was set to 1.0. The batch size was set to 128 for contrastive and our method and to 120 for triplet. For training, all the convolutional layers were initialized from the network pretrained on ImageNet ILSVRC [30] dataset and the fully connected layer (the last layer) was initialized with random weights. We also multiplied the learning rate for the randomly initialized fully connected layers by 10.0 for faster convergence. All the train and test images are normalized to 256 by 256. For training data augmentation, all images are randomly croppped at 227 by 227 and randomly mirrored horizontally. For training, we exhaustively use all the positive pairs of examples and randomly subsample approximately equal number of negative pairs of examples as positives.

6. Evaluation

In this section, we briefly introduce the evaluation metrics used in the experiments. For the clustering task, we use the F\(_1\) and NMI metrics. F\(_1\) metric computes the harmonic mean of precision and recall. \(F_1 = \frac{2PR}{P+R}\). The normalized mutual information (NMI) metric takes as input a set of clusters \(\Omega = \{\omega_1, \ldots, \omega_K\}\) and a set of ground truth classes \(C = \{c_1, \ldots, c_K\}\). \(\omega_i\) indicates the set of examples with cluster assignment \(i\). \(c_j\) indicates the set of examples with the ground truth class label \(j\). Normalized mutual information is defined as the ratio of mutual information and the average entropy of clusters and the entropy of labels.
NMI \((\Omega, C) = \frac{I(\Omega; C)}{2(H(\Omega) + H(C))}\). We direct interested readers to refer \[23\] for complete details. For the retrieval task, we use the Recall@K \[15\] metric. Each test image (query) first retrieves K nearest neighbors from the test set and receives score 1 if an image of the same class is retrieved among the K nearest neighbors and 0 otherwise. Recall@K averages this score over all the images.

7. Experiments

We show experiments on CUB200-2011 \[37\], CARS196 \[19\], and our Online Products datasets where we use the first half of classes for training and the rest half classes for testing. For testing, we first compute the embedding on all the test images at varying embedding sizes \{64, 128, 256, 512\} following the practice in \[1, 31\]. For clustering evaluation, we run affinity propagation clustering \[11\] with bisection method \[10\] for the desired number of clusters set equal to the number of classes in the test set. The clustering quality is measured in the standard \(F_1\) and NMI metrics. For the retrieval evaluation, we report the result on the standard Recall@K metric \[15\] in log space of K. The experiments are performed with GoogLeNet \[33\].

7.1. CUB-200-2011

The CUB-200-2011 dataset \[37\] has 200 classes of birds with 11,788 images. We split the first 100 classes for training (5,864 images) and the rest of the classes for testing (5,924 images). Figure 6 shows the quantitative clustering quality for the contrastive \[14\], triplet \[39, 31\], and using pool5 activation from the pretrained GoogLeNet \[33\] network on ImageNet \[30\], and our method on both \(F_1\), NMI, and Recall@K metrics. Our embedding shows significant performance margin both on the standard \(F_1\), NMI, and Recall@K metrics on all the embedding sizes. Figure 8 shows some example query and nearest neighbors on the test split of CUB200-2011 \[37\] dataset. Figure 9 shows the Barnes-Hut t-SNE visualization \[36\] on our 64 dimensional embedding. Although t-SNE embedding does not directly translate to the high dimensional embedding, it is clear that similar types of birds are quite clustered together and are apart from other species.

7.2. CARS196

The CARS196 data set \[19\] has 198 classes of cars with 16,185 images. We split the first 98 classes for training (8,054 images) and the other 98 classes for testing (8,131 images). Figure 7 shows the quantitative clustering quality for the contrastive \[14\], triplet \[39, 31\], and using pool5
Figure 9: Barnes-Hut t-SNE visualization [36] of our embedding on the test split (class 101 to 200; 5,924 images) of CUB-200-2011. Best viewed on a monitor when zoomed in.

Figure 11: Barnes-Hut t-SNE visualization [36] of our embedding on the test split (class 99 to 196; 8,131 images) of CARS196. Best viewed on a monitor when zoomed in.
7.3. Online Products dataset

We used the web crawling API from eBay.com [9] to download images and filtered duplicate and irrelevant images (i.e. photos of contact phone numbers, logos, etc). The preprocessed dataset has 120,053 images of 22,634 online products (classes) from eBay.com. Each product has approximately 5.3 images. For the experiments, we split 59,551 images of 11,318 classes for training and 60,502 images of 11,316 classes for testing. Figure 12 shows the quantitative clustering and retrieval results on $F_1$, NMI, and Recall@K metric with GoogLeNet. Figures 13 and 14 show some example queries and nearest neighbors on the
Figure 15: Barnes-Hut t-SNE visualization [36] of our embedding on the test split (class 11,319 to 22,634; 60,502 images) of Online Products.
Figure 13: Examples of successful queries on our Online Products dataset using our embedding (size 512). Images in the first column are query images and the rest are five nearest neighbors. Best viewed on a monitor zoomed in.

dataset for both successful and failure cases. Despite the huge changes in the viewpoint, configuration, and illumination, our method can successfully retrieve examples from the same class and most retrieval failures come from fine grained subtle differences among similar products. Figure 15 shows the t-SNE visualization of the learned embedding on our Online Products dataset.

8. Conclusion

We described a deep feature embedding and metric learning algorithm which defines a novel structured prediction objective on the lifted dense pairwise distance matrix within the batch during the neural network training. The experimental results on CUB-200-2011 [37], CARS196 [19], and our Online Products datasets show state of the art performance on all the experimented embedding dimensions.

References

[19] J. Krause, M. Stark, J. Deng, and F.-F. Li. 3d object representations for fine-grained categorization. ICCV 3dRR-13, 2013. 1, 2, 3, 4, 5, 6