LEARNING METRICS BY LEARNING CONSTRAINED EMBEDDINGS OF OBJECTS TO $\mathbb{R}^n$

Henry Gouk & Bernhard Pfahringer
Department of Computer Science
University of Waikato
Hamilton, New Zealand
hgrg1@students.waikato.ac.nz, bernhard@cs.waikato.ac.nz

Michael Cree
School of Engineering
University of Waikato
Hamilton, New Zealand
cree@waikato.ac.nz

Abstract

Similarity metrics are a core component of many information retrieval and machine learning systems. In this work we propose a method capable of learning a similarity metric from data equipped with a binary relation. By modelling the adjacency matrix of the relation we are able to learn target vectors for each instance. A regression model can then be constructed that maps instances to these learned targets, resulting in a feature extractor that computes vectors for which Euclidean distance is a meaningful measure of relatedness. We present results on several image recognition datasets.

1 Introduction

Similarity metrics play an important role in information retrieval, face verification and recognition, recommender systems, and also many machine learning algorithms. The problem of constructing a similarity metric through the use of machine learning is well studied and several recent advances have taken advantage of the power of deep neural networks, resulting in impressive accuracy gains on common face verification and recognition benchmark tasks (Taigman et al., 2014; Sun et al., 2014; Schroff et al., 2015). Although these results are a promising indicator that deep learning can be used to learn similarity metrics, there are still aspects that can be improved.

A common trend among similarity metric learning algorithms is to train on pairs of instances that are considered similar, resulting in a huge increase in the computational cost of constructing the metric compared to other learning tasks where most algorithms only consider a single instance at a time. A second problem lies in the benchmark tasks often used to evaluate new methods. Many techniques are evaluated on classification benchmark datasets, where each instance is labelled with a class and two instances are considered similar if and only if they share the same class. By evaluating metrics in this way there is an implicit assumption that transitivity is desired—which for some tasks is justified—but this is not always the case. For example, consider three images: A, B, and C. It could be the case the that A and B are considered similar because they have similar backgrounds. It could also be the case that B and C are similar because the same object appears in the foreground of both, despite the rest of the scene being vastly different. If transitivity were to be enforced then A and C would be considered similar, even though there is no reason for that to be the case.

We are in both transitive and nontransitive relations, such as multi-label classification datasets. Elaborating on the previous example: images A and B would be given a label indicating what type of background they have, while B and C would be given a label indicating what foreground object appears in the image. In this case B has two labels, and hence the task fits into the multi-label classification paradigm. However, in this work we take an even more general view of the problem.
definition. We assume only that the information provided consists of pairwise similarity or dissimilarity constraints. This generalised view enables the use of more diverse data collection strategies, such as the relevance feedback methods commonly used in information retrieval systems.

In our approach to similarity metric learning we acknowledge that there are latent classes within the data, however no explicit knowledge of these classes is required. By taking advantages of existence of these latent classes, we first learn the structure of a target vector space for an embedding function, and subsequently learn a model that performs the embedding. Our algorithm does not operate on pairs of feature vectors, and hence results in a computationally cheaper approach to learning instance similarity.

2 RELATED WORK

Metric learning is a well established area, and much research has been put into developing sophisticated algorithms that can learn instance similarity. Metric learning techniques can be roughly divided into two categories: Mahalanobis based methods, and neural network based methods. In this section we briefly outline previous work in both of these areas.

2.1 MAHALANOBIS BASED METHODS

The general form of a Mahalanobis distance metric over a set, \( X \), is given in Equation 1, where \( \vec{x}_i, \vec{x}_j \in X \). The algorithms based on this model primarily differ on how the linear transform is optimised.

\[
D(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^\top M (\vec{x}_i - \vec{x}_j)
\]  

The large margin nearest neighbours (Weinberger et al., 2005) algorithm employs semidefinite programming to optimise a loss function composed of two terms: one that draws very similar instances together, and another that encourages a margin to be formed between dissimilar instances. As the name suggests, the motivation for the development of this algorithm was to improve the accuracy of \( k \)-Nearest Neighbours (\( k \)-NN) classifiers.

Information-theoretic metric learning is another Mahalanobis technique, and was introduced by Davis et al. (2007). This criteria aims to minimise the Kullback-Leibler divergence between two multivariate Gaussians, of which the inverse covariance matrices are used to define Mahalanobis distance metrics. One of these Gaussians is defined in advance and acts as a regulariser, while the other is treated as a free parameter and optimised subject to constraints derived from similarity information.

Neighbourhood Components Analysis (NCA) (Goldberger et al., 2004) is another method developed to be used in conjunction with \( k \)-NN classifiers. This technique attempts to find the matrix, \( M \), by minimising a differentiable loss function that approximates the behaviour of \( k \)-NN in the transformed feature space.

2.2 NEURAL NETWORK METHODS

Siamese neural networks were first introduced for solving the task of hand written signature verification (Bromley et al., 1993), and since then one of the primary motivations for advancing their capabilities has been face verification (Taigman et al., 2014; Chopra et al., 2005). They are composed of two identical feature extraction networks, the output of which are compared using cosine similarity.

An important development for Siamese networks was the contrastive loss function (Hadsell et al., 2006; Chopra et al., 2005). The loss is based on Euclidean distance, rather than cosine similarity, and aims only to create a margin between dissimilar instances as opposed to naively maximising the distance between them. Applications are presented, including face verification and data visualisation.

Related to Siamese networks is the recent trend of using triplet loss functions (Gomez-Ojeda et al., 2015; Schroff et al., 2015) that attempt to reduce the distance between a target instance and a positive
example, while simultaneously increasing the distance between the target and a negative example. This type of loss function allows one to specify relative similarity as ground truth, rather than absolute similarity. These networks can be thought of as Siamese networks with three extractors.

There is also an extension of NCA to nonlinear transformations of the input data [Salakhutdinov & Hinton, 2007]. This method can be viewed as a probabilistic variant of Siamese networks. The nonlinear transformation models used in the original exposition of this method were stacked Restricted Boltzmann Machines, initialised using unsupervised pretraining and subsequently finetuned using the NCA loss function.

3 Method

3.1 Metric Learning with Binary Relations

Many similarity metric learning algorithms are trained on pairs of instances sampled from a set of labelled data, rather than individual instances. For each of these pairs, a new label is generated; positive if the two instances are seen as being similar, or negative otherwise. Although this problem transformation is convenient, training on pairs greatly increases the computational cost of building models, as the number of attributes is doubled and the effective training set size is squared. When the task is viewed as learning a similarity metric it is not easy to see a way around this drawback, however it is also possible to look at many similarity learning tasks as the problem of learning a binary relation. Under this interpretation, the binary classification tasks performed by algorithms that operate on pairs of instances can be thought of as methods for constructing models that determine whether two examples are related according to some binary relation.

A key advantage of posing similarity learning as the task of inferring a binary relation is the ability to decompose the training data into classes. This idea is most often studied in conjunction with equivalence relations, where the equivalence classes form a disjoint partition over the original training set. In this context the relation must exhibit reflexivity, symmetry, and transitivity—the last of which is somewhat limiting. One can drop the requirement for transitivity by instead considering binary tolerance relations, which are only reflexive and symmetric. As with equivalence relations, it is possible to decompose the training data into a set of classes (termed tolerance classes), however these classes need not be disjoint.

If the relation on the training data is a binary equivalence relation, then the task to be solved is simply that of building a classifier. If the relation is not transitive, but still reflexive and symmetric and hence a tolerance relation, the problem can be seen as a multi-label classification task. In this situation each label corresponds to a tolerance class. Thus, many similarity metric learning problems can be described as multi-label classification tasks. Partitioning the training set into tolerance classes given only the relation matrix is an instance of the NP-complete problem of listing all maximal cliques in a graph [Karp, 1972], so an alternative approach must be taken.

Rather than partitioning the training set into—potentially overlapping—subsets, each corresponding to a tolerance class, we find a target vector for each instance such that the targets of instances that are related will be close in the target vector space, and unrelated instances will be far apart. The rationale behind this is that each tolerance class will be mapped to a cluster in the target vector space, however we can employ a technique that does not have to explicitly determine to which tolerance classes each instance belongs.

3.2 Learning Targets

Instead of training a large Siamese network on the—potentially quite large—feature vectors in the training set, we build a network that takes instance identifiers as input. That is, each instance in the training set is given a unique nominal identifier. A Siamese network with a single hidden layer in each extractor can then be trained on pairs of these identifiers, with the task being to determine whether the instances are related or not. Once the network has converged, the columns of the weight matrix (or rows, depending on the convention used) can be used as targets for a regression network. This process for learning the target representations is conceptually similar to the skip-gram with negative sampling model used for learning word representations [Mikolov et al., 2013], although the formalisation provided in their work is given in terms of a standard multi-layer perceptron. A
notable difference is that the two extractors of the Siamese network in the skip-gram model do not share the same weight matrix, giving their network the ability to model binary relations that do not exhibit symmetry and reflexivity.

More formally: say we have a training set, $X$, of objects equipped with a binary relation, $R$. The adjacency matrix, $M$, of $R$ is defined as follows:

$$
M_{i,j} = \begin{cases} 
1, & (\vec{x}_i, \vec{x}_j) \in R \\
0, & \text{otherwise}
\end{cases}
$$

In practice, we do not always know all entries in $M$, so the relation can instead be represented as a set of triples, $Z$, of the form $(i, j, y)$, where $\vec{x}_i, \vec{x}_j \in X$ and $y$ is the known value of $M_{i,j}$. By building a model that can accurately estimate the known values of $M_{i,j}$ given only the indices, $i$ and $j$, it is possible to learn the structure of a vector space where a predefined distance metric is a good indicator of similarity.

We use the following model for estimating entries in $M$:

$$
\hat{M}_{i,j} = \begin{cases} 
1, & \text{if } p(i, j) \geq \text{threshold} \\
0, & \text{otherwise}
\end{cases}
$$

The value of $\text{threshold}$ is only important if one actually wishes to reconstruct the relation matrix or fill in missing entries. High quality target vectors will be found as long as a value for $\text{threshold}$ exists such that a large percentage of entries are correctly reconstructed, however this value need not be computed. We consider two different possibilities for $p(\cdot, \cdot)$:

$$
p(i, j) = e^{-\|\vec{t}_i - \vec{t}_j\|^2_2}
$$

and

$$
p(i, j) = \frac{1}{1 + e^{b - \vec{t}_i^\top \vec{t}_{i,j}}}
$$

Where $\vec{t}_i$ is a target vector from the weight matrix and corresponds to $\vec{x}_i \in X$ and $b$ is a bias term. Equation 4 is based on the squared Euclidean distance, whereas Equation 5 is based on the cosine similarity. The target vectors are found by optimising the negative log likelihood:

$$
\arg \min_{b, \vec{t}_i} - \sum_{(i,j,y) \in Z} y \log p(i, j) + (1 - y) \log(1 - p(i, j))
$$

Stochastic gradient descent with momentum is used to find a sufficiently good local optima in a short amount of time—typically in roughly a minute when running on a CPU. When Equation 5 is used the target vectors are normalised to have a length of one each time they are updated, much like the max-norm regularisation introduced by Srebro et al. (2004) except we use an equality constraint rather than an inequality constraint. Once the target vectors have been found a regression model can be trained to minimise the squared error between the embedding of each instance and the corresponding target vector. Because the target vectors have unit length when trained with the cosine based loss function, squared error is still a sensible objective for the regression network. This is because we can take advantage of the relation between cosine distance and squared Euclidean distance when operating on points lying on a hypersphere with unit radius:

$$
\|\vec{x}_i - \vec{x}_j\|^2_2 = 2(1 - \vec{x}_i^\top \vec{x}_j)
$$

We find that optimising the squared error works well in practice, even though only one of vectors in the distance calculation is guaranteed to have unit length.
Under review as a conference paper at ICLR 2016

<table>
<thead>
<tr>
<th>DATASET</th>
<th>INSTANCES</th>
<th>CLASSES</th>
<th>Known values of M</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>70,000</td>
<td>10</td>
<td>0.033%</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>60,000</td>
<td>10</td>
<td>0.04%</td>
</tr>
<tr>
<td>PubFig83</td>
<td>13,840</td>
<td>83</td>
<td>1%</td>
</tr>
</tbody>
</table>

Figure 1: The datasets used in our experiments.

![MNIST Accuracy Graph](image)

Figure 2: Plots of the target vector size versus k-NN accuracy for the MNIST hand written digits dataset (left) and the CIFAR-10 dataset (right). The same legend applies to both plots. The dashed line indicated the performance of a classification network with the same architecture, but trained with the cross entropy loss function on the original labels.

4 EVALUATION

We primarily concern ourselves with extrinsically evaluating how well the proposed methods perform in various scenarios. Our extrinsic evaluation covers several use cases for learned similarity metrics:

- A domain specific distance metric for k-NN classifiers;
- Constructing embeddings when dealing with a large number of classes;
- Data visualisation.

The datasets used are summarised in Figure 1. For each of our experiments the algorithms used to model the adjacency matrices are only supplied with a very small subsample of the possible instance pairs.

4.1 MNIST

The first dataset considered is MNIST [LeCun et al. 1998], a collection of hand written digits. We use the network from Scherer et al. (2010) with one modification: Rectified Linear Units are used for the convolutional layer, as opposed to the logistic activation function. This is a relatively shallow architecture consisting of only a single convolutional layer with 112 feature maps and 9 × 9 filters, followed by 5 × 5 max pooling, and then a fully connected output layer. Figure 2 shows the performance of a k-NN classifier for different target vector computation configurations. An interesting application of this technique, and embedding based similarity learning techniques in general, is the ability to compute two dimensional representations of instances. This gives one the ability to create visualisations of datasets, such as those given in Figure 3 for MNIST. It should be noted that because this is a multi-class classification dataset, and hence equipped with an equivalence relation, the target learning algorithm is not provided with any information about the similarity between classes, as they are disjoint.

5
4.2 CIFAR-10

We now consider the CIFAR-10 dataset (Krizhevsky, 2009). This dataset was chosen to increase the confidence that data equipped with equivalence relations containing only a small number of classes will still exhibit good performance, despite the lack of interclass similarity information. The model used for this dataset, summarised in Figure 4, is fairly simplistic and of moderate size. We horizontally flip each input image with a probability of 0.5 while training. When the network is trained as a standard classifier with a softmax output layer and the cross entropy loss function an accuracy of 79.5% is achieved on the test set. Figure 2 shows the performance of this network when trained to predict the targets found using our methods. As with MNIST, the increase in target vector size improves performance. However, in this case we observe a degradation in performance when the target vector size is too high.

4.3 PubFig83

PubFig83 (Pinto et al., 2011) is a face recognition dataset consisting of at least 100 photos of 83 different public figures. Specifically, we use the aligned version of this dataset developed by Chiachia et al. (2014) and only use the central 60 x 60 pixels of each image. The motivation for including this dataset is to examine to what extent classification performance is impacted when a larger number of equivalence classes are present. We once again use a fairly modestly sized network, considering current trends, and vary the output size to see how k-NN classification performance is impacted. The network architecture is given in Figure 5 and the accuracy for different target vectors sizes is demonstrated in Figure 2.

5 SUMMARY AND FUTURE WORK

This paper presented a new framework for learning instance similarity based on decomposing the adjacency matrix of a relation on the training data into target vectors for each example. We have
shown empirically that these target vectors form clusters corresponding to classes present within the data. By training regression models to map the original feature vectors to these targets it is possible to learn an embedding function, such that Euclidean distance is a meaningful measure of similarity when applied to the target feature space.

A interesting direction for future research could be to apply binary matrix factorisation methods to find binary target vectors. These vectors could then be used to train multi-label classifiers to compute hash codes for complex data, such as the image datasets used herein.

REFERENCES


