Acceleration of Large Margin Metric Learning for Nearest Neighbor Classification Using Triplet Mining and Stratified Sampling

Parisa Abdolrahim Poorheravi* — Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada

Benyamin Ghojogh* — Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada

Vincent Gaudet — Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada

Fakhri Karray — Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada

Mark Crowley — Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada

*The first two authors contributed equally to this work.
Email: {pabdolra, bghojogh, vcgaudet, karray, mcrowley}@uwaterloo.ca

Abstract

Metric learning is a technique in manifold learning to find a projection subspace for increasing and decreasing the inter- and intra-class variances, respectively. Some metric learning methods are based on triplet learning with anchor-positive-negative triplets. Large margin metric learning for nearest neighbor classification is one of the fundamental methods to do this. Recently, Siamese networks have been introduced with the triplet loss. Many triplet mining methods have been developed for Siamese nets; however, these techniques have not been applied on the triplets of large margin metric learning. In this work, inspired by the mining methods for Siamese nets, we propose several triplet mining techniques for large margin metric learning. Moreover, a hierarchical approach is proposed, for acceleration and scalability of optimization, where triplets are selected by stratified sampling in hierarchical hyperspheres. We analyze the proposed methods on three publicly available datasets.

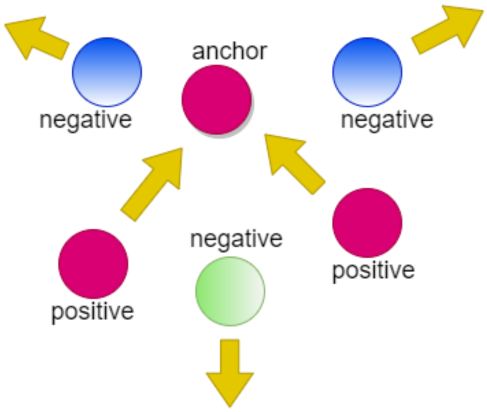

Fig. 1: Metric learning for decreasing and increasing the intra- and inter-class variances, respectively, by pulling the positives (same class instances) toward the anchor but pushing the negatives (other class instances) away.

1 Introduction

Distance metric learning is one of the fundamental and most competitive techniques in machine and manifold learning [1]. The goal of metric learning is to find a proper metric whose subspace discriminates the classes by increasing and decreasing the inter- and intra-class variances, respectively [2, 3] (e.g., see Fig. 1). This goal was first introduced by Fisher Discriminant Analysis (FDA) [2, 4].

Some metric learning methods make use of anchor-positive-negative *triplets* where the positive and negative instances are the data points having the same and different class labels with respect to an anchor instance, respectively. One of the first metric learning methods based on triplets was large margin metric learning for nearest neighbor classification [5, 6]. This method uses Semi-Definite Programming (SDP) optimization [7] as SDP has been found to be useful for metric learning [5, 6, 8, 9]. Later, the concept of a triplet cost function was proposed in the field of neural networks by introducing Siamese networks [10–12]. The triplet loss can be either in the form of Hinge loss [11] or softmax [13]. The examples of the former and latter are [5, 6, 11] and [14–16], respectively.

Solving SDP problems requires the interior point method [17], which is iterative and slow especially for big data. This can be improved and accelerated by selecting the most important data points for embedding [18]. For example, we rather care about the nearest or farthest positives and negatives than selecting all the data points. This technique is referred to as *triplet mining* in the literature where the positive and negative instances with respect to an anchor make a triplet [11].

After the introduction of Siamese networks in the literature, different triplet mining techniques were developed for Siamese training using triplets. However, these mining methods have not been implemented or proposed for the previously developed concept of large margin metric learning for nearest neighbor classification. In this work, inspired by the mining techniques for Siamese networks, we propose different triplet mining methods for large margin metric learning. By only considering the most valuable points of the dataset with respect to anchors, the SDP optimization speeds up while preserving an acceptable classification accuracy in large margin metric learning.

In addition to proposing triplet mining techniques for the optimization, we propose a hierarchical approach for further acceleration of metric learning. This approach includes iterative selection of data subsets by hierarchical stratified sampling [19] to train the embedding subspace. Not only does this approach accelerate the SDP optimization by reducing time complexity, but also it improves performance in some cases due to effectiveness of model averaging [20] and reduction of estimation variance by stratified sampling [21]. We also used the proposed triplet mining techniques in combination with the proposed hierarchical approach for the sake of acceleration.

The remainder of paper is as follows. In Section 2, we review the foundations of large margin metric learning, triplet loss, and Siamese networks. We discuss the triplet mining methods that have already been proposed for Siamese triplet training, i.e., batch all [22], batch hard [23], batch semi-hard [11], easiest/hardest positives and easiest/hardest negatives, and negative sampling [18]. Section 3 proposes how to use the triplet mining techniques in SDP optimization of large margin metric learning. The hierarchical approach is proposed in Section 4. We report the experimental results in Section 5. Finally, Section 6 concludes the paper and provides the possible future direction.

2 Background

2.1 Large Margin Metric Learning for Nearest Neighbor Classification

k-Nearest Neighbor (k-NN) classification is highly impacted by the distance metric utilized for measuring the differences between data points. Euclidean distance does not weight the points and it values them equally. A general distance metric can be viewed as the Euclidean distance after projection of points onto a discriminative subspace. This projection can be viewed as a linear transformation with a projection matrix denoted by $\boldsymbol{L}$ [24]. We call this general metric the Mahalanobis distance [1, 2]:

$$\mathcal{D} := \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 := \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)\|_2^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{M}(\boldsymbol{x}_i - \boldsymbol{x}_j), \quad (1)$$

where $\boldsymbol{M} := \boldsymbol{L}\boldsymbol{L}^\top$. The matrix $\boldsymbol{M}$ must be positive semi-definite, i.e. $\boldsymbol{M} \succeq 0$, for the metric to satisfy convexity and the triangle inequality [17].

In order to improve the k-NN classification performance, we should decrease and increase the intra- and inter-class variances of data, respectively [3]. As can be seen in Fig. 1, one way to achieve this goal is to pull the data points of the same class toward one another while pushing the points of different classes away.

Let y_{il} be one (zero) if the data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$ are (are not) from the same class. Moreover, let η_{ij} be one if $\boldsymbol{x}_j$ is amongst the k-nearest neighbors of $\boldsymbol{x}_i$ with the same class label; otherwise, it is zero. For tackling the goal of pushing together the points of a class and pulling different classes away, the following cost function can be minimized [5]:

$$\sum_{i,j} \eta_{ij} \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)\|_2^2 + c \sum_{i,j,l} \eta_{ij}(1-y_{il}) \Big[1 + \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)\|_2^2 - \|\boldsymbol{L}^\top(\boldsymbol{x}_i - \boldsymbol{x}_l)\|_2^2\Big]_+, \tag{2}$$

where $[.]_+ := \max(.,0)$ is the standard Hinge loss. The first term in Eq. (2) pushes the same-class points towards each other. The second term, on the other hand, is a triplet loss [11] which increases and decreases the inter- and intra-class variances, respectively.

Inspired by support vector machines, the cost function (2) can be restated using slack variables:

$$\begin{aligned} \underset{\boldsymbol{M},\,\xi_{ijl}}{\text{minimize}} \quad & \mathcal{L} := \sum_{i,j} \eta_{ij}\, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 + c \sum_{i,j} \eta_{ij}\,(1-y_{il})\,\xi_{ijl}, \quad \forall l \\ \text{subject to} \quad & \|\boldsymbol{x}_i - \boldsymbol{x}_l\|_{\boldsymbol{M}}^2 - \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 \geq 1 - \xi_{ijl}, \\ & \xi_{ijl} \geq 0, \\ & \boldsymbol{M} \succeq \boldsymbol{0}, \end{aligned} \tag{3}$$

which is a SDP problem [7]. The first term in the objective functions of Eqs. (2) and (3) are equivalent because of Eq. (1). The Hinge loss in Eq. (2) can be approximated using non-negative slack variables, denoted by ξ_{ijl}. The second term of objective function in Eq. (3), in addition to the first and second constraints, play the role of Hinge loss.

2.2 Triplet loss and Siamese Network

As explained for Eq. (2), the second term in that equation is the triplet loss which pushes the classes away and pulls the points of a class together. In Eq. (2), $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, and $\boldsymbol{x}_l$ are anchor, positive, and negative instances, respectively. The goal of triplet loss is to make anchor and positive instances closer and push the negative instances away as also seen in Fig. 1.

Recently, the triplet loss has been used for training neural networks which are called Siamese or triplet networks [11]. A Siamese network is composed of three sub-networks which share their weights. The anchor, positive, and negative instances are fed to these sub-networks and the triplet loss is used to tune their weights. Siamese networks are usually used for learning a discriminative embedding space. In this work, we propose several triplet mining methods inspired by the triplet mining techniques already existing in the literature for the Siamese nets.

3 Proposed Triplet Mining

The optimization problem in Eq. (3) considers all the negative instances even in large datasets. The SDP for solving Problem (3) is very time-consuming and slow [7]. Hence, Problem (3) becomes intractable for large datasets, as has been noted in [5]. This motivated us to use triplet mining on the data for further improvement upon [5, 6]. There exist several triplet mining methods which are proposed for Siamese network training. Inspired by those, we propose here the triplet mining techniques in the objective function of Eq. (3) to facilitate the optimization process. There can be different ways for triplet mining. In the following, we propose k-batch all, k-batch hard, k-batch semi-hard, extreme distances, and negative sampling for large margin metric learning.

3.1 k-Batch All

One of the mining methods to be considered is *batch all* which takes all the positives and negatives of the data batch into account for Siamese neural network [22]. The proposed method in [5, 6] is a batch-all version which takes only k nearest positives and all the negatives. This makes sense because the SDP is slow and cannot handle all possible permutations of positive and negative instances. Here, we call this method *k-batch all (k-BA)* where the objective in equation:

$$\mathcal{L} = \sum_{i,j} \eta_{ij}\, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 + c \sum_{i,j} \eta_{ij}\,(1-y_{il})\,\xi_{ijl}, \quad \forall l. \tag{4}$$

3.2 k-Batch Hard

Another mining method for Siamese networks is *batch hard* in which the farthest positive and nearest negative with respect to the anchor are considered [23]. The farthest positive is the hardest one to be classified as a neighbor of the anchor. Likewise, the nearest negative is the hardest one to be separated from the anchor's class. In this work, we consider k positive and k negative instances and we call this *k-batch hard (k-BH)* where the objective in Eq. (3) becomes:

$$\mathcal{L} = \sum_{i,j} \gamma_{ij}^+\, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 + c \sum_{i,j,l} \gamma_{ij}^+\, \eta_{il}^-\,(1-y_{il})\,\xi_{ijl}, \tag{5}$$

where γ_{ij}^+ is one (zero) if $\boldsymbol{x}_j$ is (is not) amongst the k-farthest neighbors of $\boldsymbol{x}_i$ with the same class label. Similarly, η_{il}^- is one (zero) if $\boldsymbol{x}_l$ is (is not) amongst the k-nearest neighbors of $\boldsymbol{x}_i$ with different class label.

3.3 k-Batch Semi-Hard

Batch semi-hard is another method, for Siamese networks, in which the hardest negatives (closest to the anchor) that are farther from the positive are taken into account [11]. In our work, we have k positive instances and for each, we consider k negatives. This method we call *k-batch semi-hard (k-BSH)* in which the cost in Eq. (3) can be modeled as:

$$\mathcal{L} = \sum_{i,j} \eta_{ij}\, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 + c \sum_{i,j} \eta_{ij} \sum_{l} \eta_{il}^{\mp}\,(1-y_{il})\,\xi_{ijl}, \tag{6}$$

where η_{ij}, as defined before, is one (zero) if $\boldsymbol{x}_j$ is (is not) amongst the k-nearest neighbors of $\boldsymbol{x}_i$ with the same class label and $\eta_{il}^{\mp}$ is one (zero) if $\boldsymbol{x}_l$ is (is not) amongst the k-nearest neighbors of $\boldsymbol{x}_i$, with different class label, and farther from $\boldsymbol{x}_j$ to $\boldsymbol{x}_i$.

3.4 Extreme Distances

Considering that every instance could be chosen based on their distance to the anchor (whether they are nearest or farthest), we have four different cases [25]. Easy and hard positives correspond to the nearest and farthest positives, respectively; easy and hard negatives correspond to the farthest and nearest negatives, respectively. Easy Positive-Easy Negative (EPEN), Easy Positive-Hard Negative (EPHN), Hard Positive-Easy Negative (HPEN), and Hard Positive-Hard Negative (HPHN) are the four possible cases. HPHN is equivalent to the batch hard method explained in Section 3.2. Since we are taking k instances from both positive and negative sets, the cost in Eq. (3) for the other three cases are as follows:

$$k\text{-EPEN:} \quad \mathcal{L} = \sum_{i,j} \eta_{ij}\, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 + c \sum_{i,j,l} \eta_{ij}\, \gamma_{il}^-\,(1-y_{il})\,\xi_{ijl}, \tag{7}$$

$$k\text{-EPHN:} \quad \mathcal{L} = \sum_{i,j} \eta_{ij}\, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 + c \sum_{i,j,l} \eta_{ij}\, \eta_{il}^-\,(1-y_{il})\,\xi_{ijl}, \tag{8}$$

$$k\text{-HPEN:} \quad \mathcal{L} = \sum_{i,j} \gamma_{ij}^+\, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 + c \sum_{i,j,l} \gamma_{ij}^+\, \gamma_{il}^-\,(1-y_{il})\,\xi_{ijl}, \tag{9}$$

where γ_{il}^- is one (zero) if $\boldsymbol{x}_l$ is (is not) amongst the k-farthest neighbors of $\boldsymbol{x}_i$ with different class label. The hardest cases are useful due to the concept of opposition learning [26] and the fact that more difficult separable data points are better to be emphasized. Moreover, the easiest cases are found to be effective in the literature [16].

3.5 k-Negative Sampling

In *negative sampling*, as another mining method proposed for Siamese networks, for every positive instance, each negative's probability of occurrence is calculated using a stochastic probability distribution. The distribution of pairwise distances, denoted by $q(\mathcal{D})$, of two points can be estimated as [18]:

$$q(\mathcal{D}) \propto \mathcal{D}^{d-2}(1-0.25\mathcal{D}^2)^{\frac{d-3}{2}}, \tag{10}$$

where d is the dimensionality of data and $\mathcal{D}$ is defined by Eq. (1). For an anchor $\boldsymbol{x}_i$, the probability of a negative instance $\boldsymbol{x}_l$, with distance $\mathcal{D}$ from $\boldsymbol{x}_i$ can be calculated as [18]:

$$\mathbb{P}(\boldsymbol{x}_l \,|\, \boldsymbol{x}_i) \propto \min(\lambda, q^{-1}(\mathcal{D})), \tag{11}$$

where λ (e.g., 1.4) is for giving all the negatives a minimum chance of selection. One can use a roulette wheel strategy for selecting negative instances using the probability in Eq. (11) [27].

In this work, we select the k-nearest positives and sample k negatives for every anchor-positive pair. We call this method *k-negative sampling (k-NS)* and its cost function in Eq. (3) is:

$$\mathcal{L} = \sum_{i,j} \eta_{ij}\, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\boldsymbol{M}}^2 + c \sum_{i,j,l} \eta_{ij}\, \rho_{il}^-\, (1-y_{il})\, \xi_{ijl}, \tag{12}$$

where ρ_{il}^- is one (zero) if $\boldsymbol{x}_l$ is (is not) a sampled negative for the $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ anchor-positive pair.

4 Proposed Hierarchical Large Margin Metric Learning with Stratified Sampling

The triplet mining methods, introduced in the previous section, are promising techniques for better and faster performance of large margin metric learning; however, they can be further improved as explained here. We propose a hierarchical approach for accelerating the large margin metric learning. The main idea is to consider portions of data for training for solving the optimization in order to tackle the slow pace of SDP. However, for taking into account the whole training data, portions of data should be introduced to the optimization problem hierarchically. This technique has a divide and conquer manner to accelerate the training phase [28]. It also can improve the performance of embedding model due to model averaging [20, 29] and reduction of estimation variance by stratified sampling [21].

The procedure of this hierarchical approach can be found in Algorithm 1. As can be seen in this algorithm, this approach is iterative. In every iteration, several hyper-spheres are considered in the space of data and the triplets are sampled from inside of the hyper-spheres (see Line 10 in Algorithm 1). We employ stratified sampling [19] where classes of data are considered to be strata. The SDP optimization, Eq. (3), is solved at every iteration using merely the sampled triplets rather than the whole data (see Line 11 in Algorithm 1). We factorize the matrix $\boldsymbol{M}$ in Eq. (1) into $\boldsymbol{L}\boldsymbol{L}^\top$ using eigenvalue decomposition:

$$\boldsymbol{M} = \boldsymbol{\Psi}\boldsymbol{\Sigma}\boldsymbol{\Psi}^\top = \boldsymbol{\Psi}\boldsymbol{\Sigma}^{(1/2)}\boldsymbol{\Sigma}^{(1/2)}\boldsymbol{\Psi}^\top = \boldsymbol{L}\boldsymbol{L}^\top, \tag{13}$$

which can be done because $\boldsymbol{M} \succeq 0$. As Eq. (1) shows, metric learning can be viewed as Euclidean distance after projection onto a subspace spanned by the columns of $\boldsymbol{L}$, i.e., the column space of $\boldsymbol{L}$. Hence, the whole data are projected into the metric subspace trained by the sampled triplets (see Line 13 in Algorithm 1). This is effectively akin to a principal component analysis [24] in the space of triplets. Note that for not having data being collapsed in subspaces with low ranks, one can slightly strengthen the diagonal of $\boldsymbol{M}$ which results in larger eigenvalues without effecting the projection directions [30].

At every iteration, the number of hyper-spheres, denoted by n_s, and the radius of them, denoted by r, are determined by a function decreasing and increasing with respect to the iteration index, respectively. This is because by the progress of algorithm, we want to make the hyper-spheres coarser to see more of data but at the same time, the number of them should become less not to have much overlap between the sampling areas. The size of stratified sampling, denoted by n_τ, in every hyper-sphere can also alter by the iteration index τ because in the late iterations, there is no need to consider the whole data in the hyper-sphere but a part of them. For the stratified sampling size, we sample a portion of each available class (i.e., strata) within the hyper-sphere, where the sampling portion in iteration τ is denoted by p_τ.

```
1  Procedure: Hierarchical Metric Learning($\boldsymbol{X}$, $k$)
2  Input: $\boldsymbol{X}$: dataset, $k$: number of neighbors
3  Initialize $r$, $n_s$, and $p_\tau$
4  for $\tau$ from $1$ to $T$ do
5      $r :=$ increasing function of $\tau$
6      $n_s :=$ decreasing function of $\tau$
7      $p_\tau :=$ decreasing function of $\tau$
8      for $s$ from $1$ to $n_s$ do
9          $\boldsymbol{c}_s \sim$ range($\boldsymbol{X}$)
10         $\{\boldsymbol{x}_{i,a}, \boldsymbol{x}_{i,p}, \boldsymbol{x}_{i,n}\}_{i=1}^{n_\tau} \leftarrow$ draw a stratified triplet sample
             with sampling portion $p_\tau$ within the $s$-th
             hypersphere
11         Solve optimization (3)
12         Decompose $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^\top$ using Eq. (13)
13         Project $\boldsymbol{X}$ onto $\mathbb{C}$ol($\boldsymbol{L}$): $\boldsymbol{X} \leftarrow \boldsymbol{L}^\top\boldsymbol{X}$
```

Algorithm 1: Hierarchical Large Margin Metric Learning

We initialize the radius, number of hyper-spheres, and the portion of sampling by $r := 0.1\sigma$, $n_s := \lfloor 0.01 \times n \rfloor$ (clipped to $10 \leq n_s \leq 20$), and $p_\tau := 1$. The updates of these variables are performed as $r := r + \Delta r$, $n_s := \max(n_s - \lceil 0.2 \times n_s \rceil, 1)$, and $p_\tau := \max(p_\tau - 0.05, 0.2)$, where $\Delta r := 0.3\sigma$ and σ is the average standard deviation along features.

5 Experimental Results and Analysis

5.1 Datasets and Setup

In this work, we use three publicly available datasets. The first dataset is the Fisher Iris data [31] which includes 150 data points in three classes with dimensionality of 4. The second dataset which we used was ORL faces data [32] with 40 classes each having 10 subjects. The size of facial images are 112×92 pixels. The third dataset was the MNIST digits data [33] with 28×28-pixel images.

The Iris dataset was randomly split into train-validation-test sets with portions 70%-15%-15%. In the ORL dataset, the first six faces of every subject made the training data and the rest of images were split to test and validation sets. A subset of MNIST with 400-100-100 images was also taken for train-validation-test. Note that the SDP in large margin metric learning cannot handle very large datasets due to the slow pacing of optimization. The ORL dataset was further projected onto the 15 leading eigenfaces [34] as preprocessing [24]. The validation set was used for determining the optimal values of k and c. The MNIST data were also projected onto the principal component analysis subspace with dimensionality 30.

5.2 Comparison of Triplet Mining Methods in the Non-Hierarchical and Hierarchical Approaches

For each dataset, we returned the accuracy of the k-nearest classification using the Mahalanobis distance for the different triplet mining methods. Table 1 represents the accuracies and run-time for Iris, ORL faces, and MNIST datasets, respectively.

In all datasets, k-BH has obtained the highest accuracy in non-hierarchical approach. However, in hierarchical approaches, k-BSH has obtained a top accuracy. The reason for k-BH and k-BSH to have acceptable performance is using the hard (near) negative instances in the training. This helps avoiding overfitting to the training data. In ORL faces data, the best accuracy is for k-BH and k-EPHN. This is because in both of these methods, the hardest negative instances are used for training, helping to avoid overfitting again. For the same reason, k-BSH has the second best performance in this dataset. Moreover, we see that the results of k-NS is acceptable in this data which is due to the effectiveness of the probability distribution used for sampling from the negative instances. This distribution was recently proposed for Siamese training [18]; however, the results show that it is also effective for triplet mining in the large margin metric learning.

Table 1: Comparing accuracies and run-time of the proposed triplet mining methods in both non-hierarchical and hierarchical metric learning for nearest neighbor classification.

Dataset			k-BA	k-BH	k-BSH	k-HPEN	k-EPEN	k-EPHN	k-NS
Iris	Non-Hierarchical	Accuracy (%)	72.73	100	86.36	95.45	81.82	95.45	72.73
		Time (sec)	832.85	5.51	6.62	4.77	5.34	5.11	5.06
	Hierarchical	Accuracy (%)	100	100	100	100	100	100	100
		Time (sec)	23.73	9.72	4.54	7.25	4.73	5.05	4.64
ORL Faces	Non-Hierarchical	Accuracy (%)	–	85.00	78.75	72.50	75.00	85.00	77.50
		Time (sec)	–	16.13	18.61	19.59	19.19	16.31	19.05
	Hierarchical	Accuracy	76.25	76.25	81.25	78.75	78.75	81.25	63.75
		Time (sec)	0.39	0.93	0.79	4.36	1.07	0.95	0.39
MNIST	Non-Hierarchical	Accuracy (%)	–	82.00	79.00	82.00	78.00	82.00	78.00
		Time (sec)	–	122.21	182.13	152.89	173.18	135.64	170.33
	Hierarchical	Accuracy (%)	71.00	77.00	79.00	81.00	75.00	78.00	79.00
		Time (sec)	27.17	1.56	1.55	0.49	1.02	1.70	1.55

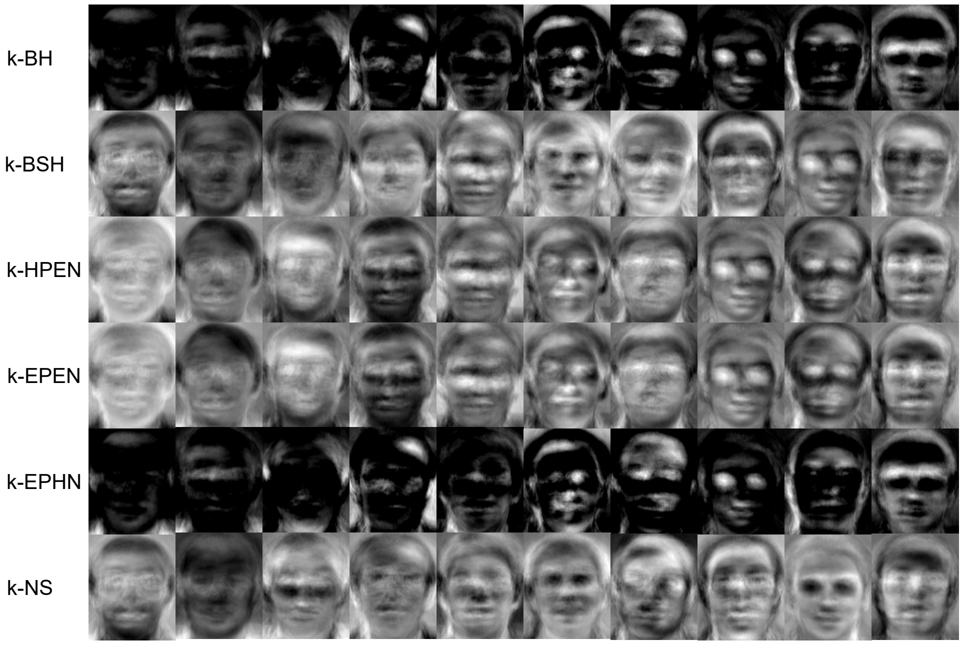

Fig. 2: The top ten ghost faces in different triplet mining methods.

In the case of Iris data, due to the small size and simplicity of dataset, the accuracies are all perfect in the hierarchical approach. In this approach, for the ORL and MNIST datasets, the highest accuracies are for k-BSH which can be interpreted as explained above. As obvious in table, the hierarchical approach either outperforms the non-hierarchical approach (due to model averaging) or has comparable result with much less consumed time. The cases, where non-hierarchical approach has slightly better accuracy, are due to the random sampling from hyper-spheres rather than using all data.

In the non-hierarchical approach, we tested the k-BA merely on the Iris dataset because the two other datasets are too large for k-BA as it considers all the negative instances. For the same reason, it is very time consuming; hence, the longest time belongs to k-BA in Table 1. For the ORL and MNIST datasets, the longest time belongs to k-HPEN and k-BSH, respectively, mainly due to handling the hard cases in optimization. As the table shows, the hierarchical approach is scalable and much faster because of sampling. For this reason, we could run k-BA efficiently for all three datasets in this approach. Note that the characteristic of computer used for simulations was Intel Core-i7, 1.80GHz, with 16GB RAM.

5.3 Comparison of Triplet Mining Methods By Ghost Faces

As Eq. (13) shows, metric learning can be viewed as Euclidean distance after projection onto a subspace spanned by the columns of $\boldsymbol{L}$. In the eigenvalue decomposition, the eigenvectors and eigenvalues are sorted from the leading to trailing.

Inspired by eigenfaces [34] and Fisherfaces [35], for the large margin metric learning, we can visualize the eigen-subspaces (column spaces of $\boldsymbol{L}$) for the facial dataset in order to display the ghost faces. Here, we consider the top ten columns of $\boldsymbol{L}$. The ghost faces of the ORL face dataset are depicted in Fig. 2. As seen in this figure, k-NS features are more discriminative which distinguish the different classes using various extracted features including eye, eyebrow, cheeks (for eye glasses), chin, hair, and nose. In second place after k-NS, the k-BSH, k-HPEN, and k-EPEN features are diverse enough (including eye, cheek, nose, and hair) for discriminating the classes. The k-BH and k-EPHN have mostly concentrated on the eye and eye-brow. This makes sense because many of the subjects in the ORL face dataset wear eye-glasses.

6 Conclusion and Future Direction

Large margin metric learning for for nearest neighbor classification makes use of SDP optimization which is very slow and computationally expensive, because of the interior point optimization method, especially when the data scale up. In this paper, inspired by the state-of-the-art triplet mining techniques for Siamese network training, we proposed and analyzed several triplet mining methods for large margin metric learning. These triplet mining methods make the set of triplets smaller by limiting the instances to the most important ones. This speeds up the optimization and makes it more efficient. The proposed triplet mining techniques were k-BA, k-BH, k-BSH, k-HPEN, k-EPEN, k-EPHN, and k-NS. Moreover, We suggested a new hierarchical approach which, in combination with the triplet mining methods, reduces the time of training considerably and makes the method scalable. Our experiments on three public available datasets verified the effectiveness of the proposed approaches. A possible future direction is to try the proposed hierarchical approach using stratified sampling on other subspace learning methods.

References

[1] B. Kulis *et al.*, "Metric learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2013.

[2] A. Globerson and S. T. Roweis, "Metric learning by collapsing classes," in *Advances in Neural Information Processing Systems*, 2006, pp. 451–458.

[3] B. Ghojogh, M. Sikaroudi, S. Shafiei, H. R. Tizhoosh, F. Karray, and M. Crowley, "Fisher discriminant triplet and contrastive losses for training siamese networks," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2020.

[4] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer Series in Statistics New York, 2001, vol. 1, no. 10.

[5] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems*, 2006, pp. 1473–1480.

[6] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[7] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[8] B. Alipanahi, M. Biggs, and A. Ghodsi, "Distance metric learning vs. Fisher discriminant analysis," in *Proceedings of the 23rd National Conference on Artificial Intelligence*, vol. 2, 2008, pp. 598–603.

[9] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 77–90, 2006.

[10] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1735–1742.

[11] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[12] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[13] M. Ye, X. Zhang, P. C. Yuen, and S.-F. Chang, "Unsupervised embedding learning via invariant and spreading instance feature," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6210–6219.

[14] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems*, 2005, pp. 513–520.

[15] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 360–368.

[16] H. Xuan, A. Stylianou, and R. Pless, "Improved embeddings with easy positive triplet mining," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 2474–2482.

[17] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[18] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2840–2848.

[19] V. Barnett, *Elements of sampling theory*. English Universities Press, London, 1974.

[20] G. Claeskens, N. L. Hjort *et al.*, "Model selection and model averaging," *Cambridge Books*, 2008.

[21] B. Ghojogh and M. Crowley, "The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial," *arXiv preprint arXiv:1905.12787*, 2019.

[22] S. Ding, L. Lin, G. Wang, and H. Chao, "Deep feature learning with relative distance comparison for person re-identification," *Pattern Recognition*, vol. 48, no. 10, pp. 2993–3003, 2015.

[23] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[24] I. Jolliffe, *Principal component analysis*. Springer, 2011.

[25] M. Sikaroudi, B. Ghojogh, A. Safarpoor, F. Karray, M. Crowley, and H. Tizhoosh, "Offline versus online triplet mining based on extreme distances of histopathology patches," in *International Symposium on Visual Computing*. Springer, 2020.

[26] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, vol. 1. IEEE, 2005, pp. 695–701.

[27] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009, vol. 74.

[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT Press, 2009.

[29] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[30] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher discriminant analysis with kernels," in *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop*. IEEE, 1999, pp. 41–48.

[31] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[32] AT&T Laboratories Cambridge, "Orl face dataset," 2001. [Online]. Available: http://cam-orl.co.uk/facedatabase.html

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[34] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1991, pp. 586–587.

[35] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 711–720, 1997.