Towards Self-Adaptive Metric Learning On the Fly

Yang Gao
The University of Texas at Dallas
Richardson, Texas
yxg122530@utdallas.edu

Yi-Fan Li
The University of Texas at Dallas
Richardson, Texas
yli@utdallas.edu

Swarup Chandra
The University of Texas at Dallas
Richardson, Texas
swarup.chandra@utdallas.edu

Latifur Khan
The University of Texas at Dallas
Richardson, Texas
lkhan@utdallas.edu

Bhavani Thuraisingham
The University of Texas at Dallas
Richardson, Texas
bhavani.thuraisingham@utdallas.edu

ABSTRACT

Good quality similarity metrics can significantly facilitate the performance of many large-scale, real-world applications. Existing studies have proposed various solutions to learn a Mahalanobis or bilinear metric in an online fashion by either restricting distances between similar (dissimilar) pairs to be smaller (larger) than a given lower (upper) bound or requiring similar instances to be separated from dissimilar instances with a given margin. However, these linear metrics learned by leveraging fixed bounds or margins may not perform well in real-world applications, especially when data distributions are complex. We aim to address the open challenge of "Online Adaptive Metric Learning" (OAML) for learning adaptive metric functions on-the-fly. Unlike traditional online metric learning methods, OAML is significantly more challenging since the learned metric could be non-linear and the model has to be self-adaptive as more instances are observed. In this paper, we present a new online metric learning framework that attempts to tackle the challenge by learning a ANN-based metric with adaptive model complexity from a stream of constraints. In particular, we propose a novel Adaptive-Bound Triplet Loss (ABTL) to effectively utilize the input constraints, and present a novel Adaptive Hedge Update (AHU) method for online updating the model parameters. We empirically validates the effectiveness and efficacy of our framework on various applications such as real-world image classification, facial verification, and image retrieval.

CCS CONCEPTS

• **Information systems** → **Similarity measures**; • **Theory of computation** → **Online learning algorithms**; • **Computing methodologies** → *Online learning settings*.

KEYWORDS

Adaptive-Bound Triplet Loss; Adaptive Metric Complexity; Online Metric Learning

1 INTRODUCTION

Learning a good quality similarity metric for images is critical to many large-scale web applications such as image classification, face verification, and image retrieval. These applications are widely used for user identity verification in mobile apps and image recommendation in multi-media search engines. Therefore, having a scalable and high-performance technique for learning a good image similarity metric is becoming increasingly important. Various online metric learning (OML) solutions applying neural networks [4, 6, 12, 14, 19] have been proposed to learn a similarity metric from a stream of constraints. These constraints are typically collected and constructed from user clicks, indicating the similarity and dissimilarity relations among images. Two kinds of constraints, i.e., pairwise and triplet constraints, are widely-accepted in previous studies. A pairwise constraint consists of two similar or dissimilar images, while a triplet constraint is of the form (A, B, C), where image A is similar to image B, but is dissimilar to image C. Compared to offline metric learning approaches, OML algorithms do not require the entire training data to be made available prior to the learning task and hence are more suitable for web applications.

Despite the success of existing OML approaches in many applications, some intrinsic drawbacks have been recently observed. First, most of the existing OML algorithms [6, 12, 14, 19] are designed to learn a pre-selected linear metric (e.g., Mahalanobis distance) by solving a convex optimization problem. These simple metrics are unable to capture the non-linear semantic similarities among instances in complex application scenarios such as image retrieval with fine-grained sub-classes. Second, the performance of existing OML solutions depend heavily on the quality of input constraints, i.e., the true similarity or dissimilarity among example images (training data) provided by the user. These algorithms learn the metric model only from the "hard" constraints within the input and are biased towards them, thereby weakening the model robustness. Here, a constraint is "hard" if the dissimilar images in it is computed as similar by current metric and vice-versa. Moreover, filtering out "hard" constraints from a large volume of candidates is computationally expensive, and is impossible to perform in a real-time setting. In

addition, a real-world input stream usually contains limited number of "hard" constraints because we have little control on the source, i.e., the users. As a result, existing OML approaches may suffer from low constraint utilization in practical applications, leading to poor performance.

These intrinsic drawbacks result in the open challenge of "Online Adaptive Metric Learning" (OAML), i.e., how to learn a metric function with adaptive complexity as more constraints are observed in the stream, while maintaining high constraint utilization during the continuous learning process. This metric function should have a dynamic hypothesis space, ranging from simple linear functions to highly non-linear functions.

In this paper, we propose a novel framework that is able to learn a ANN-based metric from a stream of constraints with full constraint utilization, and more importantly, is able to dynamically adapt its model complexity when necessary. To achieve this, we first amend the common ANN architecture so that the output of every hidden layer is given as input to an independent metric embedding layer (MEL). Each MEL represents an embedding space where similar instances are closer to each other and dissimilar instances are further away from each other. A metric weight is then assigned to each MEL, measuring its importance in the entire metric model. We propose a novel Adaptive-Bound Triplet Loss (ABTL) to evaluate the similarity and dissimilarity errors introduced by the arriving constraint at each online round, and present a novel Adaptive Hedge Update (AHU) method to update the parameters of the metric model along with the metric weights based on these errors. Note that the proposed ABTL eliminates the unique dependency on "hard" constraints in existing methods, thereby improving both the constraint utilization and the quality of learned metrics. We refer to the proposed approach as Online metric learning with Adaptive Hedge Update (OAHU).

The contribution of this paper are as follows:

- We present a novel online metric learning framework - called OAHU - that addresses the challenges in "Online Adaptive Metric Learning" by learning a ANN-based metric with adaptive model complexity and achieving full constraint utilization.
- We theoretically show the regret bound of OAHU and demonstrate the optimal range for parameter selection.
- We empirically evaluate OAHU on real-world image classification, face verification, and image retrieval tasks, and compare its results with existing state-of-the-art online metric learning algorithms.

Rest of this paper is organized as follows. We first review related works in Section 2. Then, we formalize the online metric learning problem and discuss our proposed solution in Section 3. Next, we discuss the empirical evaluation of OAHU in Section 4, and finally conclude in Section 5.

2 RELATED WORK

Existing online metric learning solutions can be broadly classified into two categories: bilinear similarity-based or Mahalanobis distance-based methods.

The bilinear similarity-based methods learn a distance metric given by $d(x_1, x_2) = x_1^T A x_2$. A popular bilinear-based method is OASIS [6]. It is an online dual approach with a passive-aggressive algorithm which computes a similarity metric to solve the image retrieval task. Sparse Online Metric Learning (SOML) [5] is another technique which learns a metric by simplifying the full matrix A by a diagnoal matrix w, i.e., $A = diag(w)$, to handle very high-dimensional cases. To reduce the memory and computational cost, the Sparse Online Relative Similarity (SORS) [1] method adopts an off-diagonal l_1 norm of A as the sparse regularization to pursue a sparse model during the learning process. These three methods are based on triplet constraints where relative similarity/dissimilarity among images are specified.

On the other hand, Mahalanobis distance-based methods focus on learning a distance metric given by $d(x_1, x_2) = (x_1 - x_2)^T A (x_1 - x_2)$. Several algorithms have been proposed. Pseudo-metric Online Learning Algorithm (POLA) [22] introduces successive projection operations to learn the optimal metric. Whereas, LEGO [12] learns a Manalanobis metric based on LogDet regularization and gradient descent. RDML [14] uses a regularized online metric learning algorithm. Recently, Li et al. [19] introduced a closed-form solution called OPML, which adopts a one-pass triplet construction strategy to learn the optimal metric with a low computational cost. All these methods, except OPML, are based on pairwise constraints where the similarity or dissimilarity among pairs of images are specified.

In general, triplet constraints are more effective than pairwise constraints [6, 23, 25] because they allow simultaneous learning of similar and dissimilar relations. Therefore, similar to OASIS and OPML, we focus on triplets to learn a high-quality metric. However, in contrast to all the previous online metric learning algorithms, our approach does not depend on the Mahalanobis or bilinear distance since their linearity property [2] may limit expressiveness when exposed to complex data distributions in practical applications. Instead, we introduce a self-adaptive ANN-based metric that flexibly selects an appropriate metric function according to the scenario on which it is deployed.

3 OAHU

3.1 Problem Setting

Given a sequence of triplet constraints $S = \{(x_t, x_t^+, x_t^-)\}_{t=1}^T$ that arrive sequentially, where $\{x_t, x_t^+, x_t^-\} \in \mathcal{R}^d$, and x_t (anchor) is similar to x_t^+ (positive) but is dissimilar to x_t^- (negative), the goal of online adaptive metric learning is to learn a model $F : \mathcal{R}^d \mapsto \mathcal{R}^{d'}$ such that $||F(x_t) - F(x_t^+)||_2 \ll ||F(x_t) - F(x_t^-)||_2$.

The overall aim of the task is to learn a metric model with adaptive complexity while maintaining a high constraint utilization rate. Here, the complexity of F needs to be adaptive so that its hypothesis space is automatically enlarged or shrinked as necessary.

In practical applications, the input triplet stream can be generated as follows. First, some seed triplets can be constructed from user clicks. Then, more triplets are formed by taking transitive closures[1] over these seed triplets. The generated triplets are inserted into the stream in chronological order of creation. In this paper, we assume

---

[1]If (x_1, x_2) and (x_1, x_3) are two similar pairs, then (x_2, x_3) is a similar pair. If (x_1, x_2) and (x_2, x_3) are two similar pairs, then (x_1, x_3) is a similar pair. If (x_1, x_2) is a similar pair and (x_1, x_3) is a dissimilar pair, then (x_2, x_3) is a dissimilar pair. If (x_1, x_2) is a similar pair and (x_2, x_3) is a dissimilar pair, then (x_1, x_3) is a dissimilar pair.

x_t^*: Any anchor, positive or negative instance in a triplet.	E_l: l^{th} metric model in OAHU.
$f^{(l)}(x_t^*)$: Metric embedding of x_t^* generated by E_l.	$\mathcal{L}^{(l)}(x_t, x_t^+, x_t^-)$: The local loss of (x_t, x_t^+, x_t^-) evaluated by E_l.
$\mathcal{L}_{overall}(x_t, x_t^+, x_t^-)$: The total loss of (x_t, x_t^+, x_t^-) evaluated in OAHU.	$\alpha^{(l)}$: Weight of metric model E_l.
$D^{(l)}(x_i, x_j)$: Distance between x_i and x_j measured in E_l based on $f^{(l)}(x)$.	$d_{sim}^{(l)}(x_i, x_j)$ / $d_{dis}^{(l)}(x_i, x_j)$: Similarity/Dissimilarity threshold in E_l for a pair (x_i, x_j).
$\mathcal{T}_{sim}^{(l)}$ / $\mathcal{T}_{dis}^{(l)}$: Upper bound of $d_{sim}^{(l)}(x_i, x_j)$ / Lower bound of $d_{dis}^{(l)}(x_i, x_j)$.	$\mathcal{L}_{attr}^{(l)}(x_t, x_t^+)$ / $\mathcal{L}_{rep}^{(l)}(x_t, x_t^-)$: Attractive/Repulsive loss of (x_t, x_t^+) / (x_t, x_t^-) evaluated in E_l.
L: Number of hidden layers in OAHU.	d: Dimensionality of input data.
S_{hidden}: Number of units in each hidden layer in OAHU.	S_{emb}: Dimensionality of learned metric embedding.
β: Discount factor for the update of $\alpha^{(l)}$.	s: Smooth factor for the update of $\alpha^{(l)}$.
η: Learning rate.	$W^{(l)}$ / $\Theta^{(l)}$: Parameter of lth hidden layer/Parameter of lth metric embedding layer.

Table 1: Commonly used symbols and terms

the existence of such triplet stream and focus on addressing the challenges of online metric learning in such scenarios.

3.2 Overview

To learn a metric model with adaptive complexity, we need to address the following question: *when* and *how* to change the "complexity" of metric model in an *online* setting? In this section, we will discuss the details of the proposed framework OAHU that addresses the question in a data-driven manner.

Figure 1 illustrates the structure of our OAHU framework. Inspired by recent works [11, 24], we use an over-complete network and automatically adapt its effective depth to learn a metric function with an appropriate complexity based on input constraints. Consider a neural network with L hidden layers: we connect an independent embedding layer to the network input layer and each of these hidden layers. Every embedding layer represents a space where similar instances are closer to each other and vice-versa. Therefore, unlike conventional online metric learning solutions that usually learns a linear model, OAHU is an ensemble of models with varying complexities that share low-level knowledge with each other. For convenience, let $E_l \in \{E_0, E_1, E_2, \ldots, E_L\}$ denote the lth metric model in OAHU, i.e., the network branch starting from input layer to the lth metric embedding layer, as illustrated in Figure 1. The simplest model in OAHU is E_0, which represents a linear transformation from the input feature space to the metric embedding space. A weight $\alpha^{(l)} \in [0, 1]$ is assigned to E_l, measuring its importance in OAHU.

For a triplet constraint (x_t, x_t^+, x_t^-) that arrives at time t, its metric embedding $f^{(l)}(x_t^*)$ generated by E_l is

$$f^{(l)}(x_t^*) = h^{(l)}\Theta^{(l)} \tag{1}$$

where $h^{(l)} = \sigma(W^{(l)}h^{(l-1)})$, with $l \geq 1, l \in \mathbb{N}$, and $h^{(0)} = x_t^*$.

Here x_t^* denotes any anchor (x_t), positive (x_t^+) or negative (x_t^-) instance based on the definition in Section 3.1, and $h^{(l)}$ represents the activation of lth hidden layer. Note that we also explicitly limit the learned metric embedding $f^{(l)}(x_t^*)$ to reside on a unit sphere, i.e., $||f^{(l)}(x_t^*)||_2 = 1$, to reduce the potential model search space and accelerate the training of OAHU.

During the training phase, for every arriving triplet (x_t, x_t^+, x_t^-), OAHU first retrieves the metric embedding $f^{(l)}(x_t^*)$ from l^{th} metric model according to Eq. 1, and then produces a local loss $\mathcal{L}^{(l)}$ for E_l by evaluating the similarity and dissimilarity errors based on $f^{(l)}(x_t^*)$. Thus, the overall loss introduced by this triplet is given

by

$$\mathcal{L}_{overall}(x_t, x_t^+, x_t^-) = \sum_{l=0}^{L} \alpha^{(l)} \cdot \mathcal{L}^{(l)}(x_t, x_t^+, x_t^-) \tag{2}$$

Since the new architecture of OAHU introduces two sets of new parameters, $\Theta^{(l)}$ (parameters of $f^{(l)}$) and $\alpha^{(l)}$, we need to learn $\Theta^{(l)}, \alpha^{(l)}$ and $W^{(l)}$ during the online learning phase. Therefore, the final optimization problem to solve in OAHU at time t is given by

$$\begin{aligned} \underset{\Theta^{(l)}, W^{(l)}, \alpha^{(l)}}{\text{minimize}} \quad & \mathcal{L}_{overall} \\ \text{subject to} \quad & ||f^{(l)}(x_t^*)||_2 = 1, \forall l = 0, \ldots, L. \end{aligned} \tag{3}$$

To solve the above optimization problem, we propose a novel Adaptive-Bound Triplet Loss (ABTL) in Section 3.3 to estimate $\mathcal{L}^{(l)}$ and introduce a novel Adaptive Hedge Update (AHU) method in Section 3.4 for online updating $\Theta^{(l)}, W^{(l)}$ and $\alpha^{(l)}$.

3.3 Adaptive-Bound Triplet Loss

3.3.1 Limitations of Existing Pairwise/Triplet Loss. Existing online metric learning solutions typically optimize a pairwise or triplet loss in order to learn a similarity metric. A widely-accepted pairwise loss [22] and a common triplet loss [19] are shown in Equation 4 and 5 respectively.

$$L(x_t, x_t') = \max\{0, y_t(d^2(x_t, x_t') - b) + 1\} \tag{4}$$

$$L(x_t, x_t^+, x_t^-) = \max\{0, b + d(x_t, x_t^+) - d(x_t, x_t^-)\} \tag{5}$$

Here $y_t \in \{+1, -1\}$ denotes whether x_t is similar ($+1$) or dissimilar (-1) to x_t' and $b \in \mathcal{R}$ is a user-specified fixed margin. Compared to the triplet loss that simultaneously evaluates both similarity and dissimilarity relations, the pairwise loss can only focus on one of the relations at a time, which leads to a poor metric quality. On the other hand, without a proper margin, the loose restriction of triplet loss can result in many failure cases. Here, a failure case is a triplet that produces a zero loss value, where dissimilar instances are closer than similar instances. As the example shown in Figure 2, for an input triplet (x_t, x_t^+, x_t^-), although x_t^+ is closer to x_t^- and further from x_t, models optimizing triplet loss would incorrectly ignore this constraint due to its zero loss value. In addition, selecting an appropriate margin is highly data-dependent, and requires extensive domain knowledge. Furthermore, as the model improves performance over time, a larger margin is required to avoid failure cases.

Next, we describe an adaptive-bound triplet loss that attempts to solve these drawbacks.

3.3.2 Adaptive-Bound Triplet Loss (ABTL). We define that instances of the same classes are mutually attractive to each other, while instances of different classes are mutually repulsive to each other. Therefore, for any input triplet constraint (x_t, x_t^+, x_t^-), there is an attractive loss $\mathcal{L}_{attr} \in [0, 1]$ between x_t and x_t^+, and a repulsive
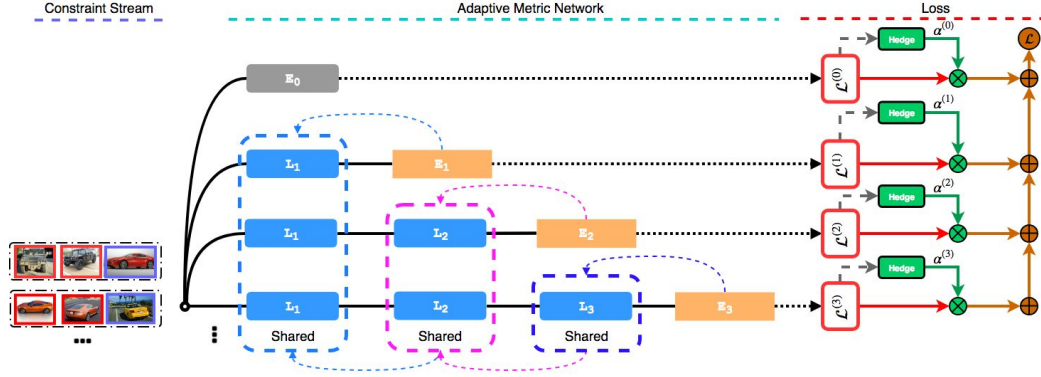
Figure 1: Overview of the OAHU. The chromatic dashed arrows represent update contributions and directions. Each $L_i \in \{L_1, L_2, \ldots\}$ represents a linear transformation layer followed by a ReLU activation. $\{E_0, E_1, \ldots\}$ are the embedding layers connected to corresponding input or hidden layers. Note that E_0 here represents a linear metric model, i.e., a linear transformation from the input feature space to the embedding space.
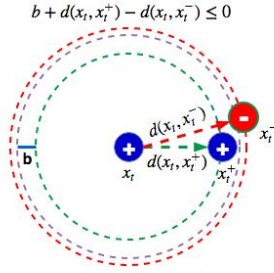

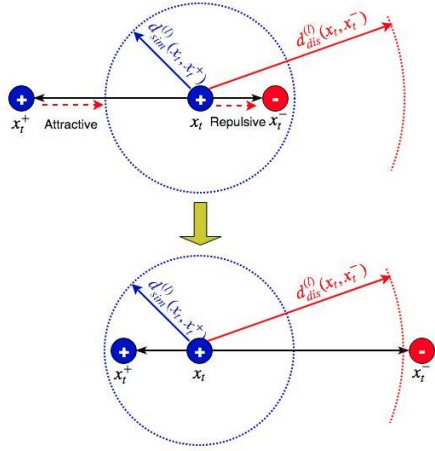$$b + d(x_t, x_t^+) - d(x_t, x_t^-) \leq 0$$

Figure 2: A failure case of triplet loss.

Figure 3: Schematic illustration of ABTL.

loss $\mathcal{L}_{rep} \in [0, 1]$ between x_t and x_t^-. In the lth metric model $E_l \in \{E_0, \ldots, E_L\}$, we focus on a distance measure $D^{(l)}(x_i, x_j)$ defined as follows:

$$D^{(l)}(x_i, x_j) = ||f^{(l)}(x_i) - f^{(l)}(x_j)||_2, \quad \forall l = 0, 1, 2, \ldots, L \quad (6)$$

where $f^{(l)}(x)$ denotes the embedding generated by E_l. Since $||f^{(l)}(x)||_2$ is restricted to be 1, we have $D^{(l)}(x_i, x_j) \in [0, 2]$. In each iteration,

let $D_{orig}^{(l)}(x_1, x_2)$ denote the distance between x_1 and x_2 before applying the update to $f^{(l)}$ (i.e., updating $\Theta^{(l)}$ and $\{W^{(j)}\}_{j=0}^l$) and $D_{update}^{(l)}(x_1, x_2)$ denote the distance after applying the update.

Figure 3 illustrates the main idea of ABTL. The objective is to have the distance $D_{update}^{(l)}(x_t, x_t^+)$ of two similar instances x_t and x_t^+ to be less than or equal to a similarity threshold $d_{sim}^{(l)}(x_t, x_t^+)$ so that the attractive loss $\mathcal{L}_{attr}^{(l)}(x_t, x_t^+)$ drops to zero; on the other hand, for two dissimilar instances x_t and x_t^-, we desire their distance $D_{update}^{(l)}(x_t, x_t^-)$ to be greater than or equal to a dissimilarity threshold $d_{dis}^{(l)}(x_t, x_t^-)$, thereby reducing the repulsive loss $\mathcal{L}_{rep}^{(l)}(x_t, x_t^-)$ to zero. Equation 7 presents these constraints mathematically.

$$\begin{cases} D_{update}^{(l)}(x_t, x_t^+) \leq d_{sim}^{(l)}(x_t, x_t^+) \\ \\ D_{update}^{(l)}(x_t, x_t^-) \geq d_{dis}^{(l)}(x_t, x_t^-) \end{cases} \quad (7)$$

Unfortunately, directly finding an appropriate $d_{sim}^{(l)}(x_t, x_t^+)$ or $d_{dis}^{(l)}(x_t, x_t^-)$ is difficult, since it varies with different input constraints. Therefore, we introduce a user-specified hyper-parameter $\tau > 0$ to explicitly restrict the range of both thresholds. Let $\mathcal{T}_{sim}^{(l)}$ denote the upper bound of $d_{sim}^{(l)}(x_t, x_t^+)$, and $\mathcal{T}_{dis}^{(l)}$ denote the lower bound of $d_{dis}^{(l)}(x_t, x_t^-)$,

$$\begin{cases} \mathcal{T}_{sim}^{(l)} = \tau \geq d_{sim}^{(l)}(x_t, x_t^+) \\ \\ \mathcal{T}_{dis}^{(l)} = 2 - \tau \leq d_{dis}^{(l)}(x_t, x_t^-) \end{cases} \quad (8)$$

Without loss of generality, we propose

$$\begin{cases} d_{sim}^{(l)}(x_t, x_t^+) = a_1 e^{D_{orig}^{(l)}(x_t, x_t^+)} + b_1 \\ \\ d_{dis}^{(l)}(x_t, x_t^-) = -a_2 e^{-D_{orig}^{(l)}(x_t, x_t^-)} + b_2 \end{cases} \quad (9)$$

where a_1, b_1, a_2 and b_2 are variables need to be determined. Note that $d_{sim}^{(l)}(x_t, x_t^+)$ and $d_{dis}^{(l)}(x_t, x_t^-)$ are monotonically increasing

and decreasing, respectively. Therefore, $d_{sim}^{(l)}$ preserves the information of original relative similarity, i.e., $d_{sim}^{(l)}(x_1, x_2) < d_{sim}^{(l)}(x_1, x_3)$ if $D_{orig}^{(l)}(x_1, x_2) < D_{orig}^{(l)}(x_1, x_3)$. This property is critical to many real-world applications that require a fine-grained similarity comparison, such as image retrieval.

The values of a_1, b_1, a_2 and b_2 are then determined by the following boundary conditions:

$$\begin{cases} d_{sim}^{(l)}\big|_{D_{orig}^{(l)}(x_t, x_t^+)=2} \leq \mathcal{T}_{sim}^{(l)} = \tau & d_{sim}^{(l)}\big|_{D_{orig}^{(l)}(x_t, x_t^+)=0} = 0 \\ d_{dis}^{(l)}\big|_{D_{orig}^{(l)}(x_t, x_t^-)=0} \geq \mathcal{T}_{dis}^{(l)} = 2-\tau & d_{dis}^{(l)}\big|_{D_{orig}^{(l)}(x_t, x_t^-)=2} = 2 \end{cases} \tag{10}$$

Thus we have

$$\begin{cases} d_{sim}^{(l)}\big(x_t, x_t^+\big) = \frac{\tau}{e^2-1}\big(e^{D_{orig}^{(l)}(x_t, x_t^+)} - 1\big) \\ d_{dis}^{(l)}\big(x_t, x_t^-\big) = -\frac{2-(2-\tau)}{1-e^{-2}}\big(e^{-D_{orig}^{(l)}(x_t, x_t^-)} - 1\big) + (2-\tau) \end{cases} \tag{11}$$

Provided with $d_{sim}^{(l)}$ and $d_{dis}^{(l)}$, the attractive loss $\mathcal{L}_{attr}$ and repulsive loss $\mathcal{L}_{rep}$ are determined by

$$\begin{cases} \mathcal{L}_{attr}^{(l)}(x_t, x_t^+) = \max\left\{0, \frac{1}{2-c_1}D_{orig}^{(l)}(x_t, x_t^+) - \frac{c_1}{2-c_1}\right\} \\ \mathcal{L}_{rep}^{(l)}(x_t, x_t^-) = \max\left\{0, \frac{-1}{c_2}D_{orig}^{(l)}(x_t, x_t^-) + 1\right\} \end{cases} \tag{12}$$

where $c_1 = d_{sim}^{(l)}(x_t, x_t^+)$ and $c_2 = d_{dis}^{(l)}(x_t, x_t^-)$.

Therefore, the local loss $\mathcal{L}^{(l)}(x_t, x_t^+, x_t^-)$, which measures the similarity and dissimilarity errors of the input triplet (x_t, x_t^+, x_t^-) in E_l, is the average of $\mathcal{L}_{attr}^{(l)}$ and $\mathcal{L}_{rep}^{(l)}$:

$$\mathcal{L}^{(l)}(x_t, x_t^+, x_t^-) = \frac{1}{2}\Big(\mathcal{L}_{attr}^{(l)}(x_t, x_t^+) + \mathcal{L}_{rep}^{(l)}(x_t, x_t^-)\Big) \tag{13}$$

Next, we answer an important question: What is the best value of τ? Here, we demonstrate that the optimal range of τ is $(0, \frac{2}{3})$. The best value of τ is thus empirically selected within this range via cross-validation.

THEOREM 3.1. *With $\tau \in (0, \frac{2}{3})$, by optimizing the proposed adaptive-bound triplet loss, different classes are separated in the metric embedding space.*

PROOF. Let $D(c_1, c_2)$ denotes the minimal distance between classes c_1 and c_2, i.e., the distance between two closest instances from c_1 and c_2 respectively. Consider an arbitrary quadrupole $(x_1, x_2, x_3, x_4) \in Q$ where $\{x_1, x_2\} \in c_1$, $\{x_3, x_4\} \in c_2$, and Q is the set of all possible quadrupoles generated from class c_1 and c_2. Suppose (x_2, x_3) is the closest dissimilar pair among all possible dissimilar pairs that can be extracted from (x_1, x_2, x_3, x_4) [2]. We first prove that the lower bound of $D(c_1, c_2)$ is given by $\min_{(x_1, x_2, x_3, x_4) \in Q} D^{(l)}(x_1, x_4) - D^{(l)}(x_1, x_2) - D^{(l)}(x_3, x_4)$. Due to the triangle inequality property of p-norm distance, we have

$$\begin{aligned} D^{(l)}(x_1, x_4) &\leq D^{(l)}(x_1, x_2) + D^{(l)}(x_2, x_4) \\ &\leq D^{(l)}(x_1, x_2) + D^{(l)}(x_2, x_3) + D^{(l)}(x_3, x_4) \end{aligned} \tag{14}$$

---

[2]it can always be achieved by re-arranging symbols of x_1, x_2, x_3 and x_4

Therefore,

$$\begin{aligned} D(c_1, c_2) &= \min_{(x_1, x_2, x_3, x_4) \in Q} D^{(l)}(x_2, x_3) \\ &\geq \min_{(x_1, x_2, x_3, x_4) \in Q} D^{(l)}(x_1, x_4) - D^{(l)}(x_1, x_2) - D^{(l)}(x_3, x_4) \end{aligned} \tag{15}$$

By optimizing the adaptive-bound triplet loss, the following constraints are satisfied

$$\begin{cases} D^{(l)}(x_1, x_2) \leq d_{sim}^{(l)}(x_1, x_2) \leq \mathcal{T}_{sim}^{(l)} \\ D^{(l)}(x_3, x_4) \leq d_{sim}^{(l)}(x_3, x_4) \leq \mathcal{T}_{sim}^{(l)} \\ D^{(l)}(x_1, x_4) \geq d_{dis}^{(l)}(x_1, x_4) \geq \mathcal{T}_{dis}^{(l)} \end{cases} \tag{16}$$

Thus

$$\begin{aligned} D(c_1, c_2) &\geq \min_{(x_1, x_2, x_3, x_4) \in Q} D^{(l)}(x_1, x_4) - D^{(l)}(x_1, x_2) - D^{(l)}(x_3, x_4) \\ &\geq \mathcal{T}_{dis}^{(l)} - 2\mathcal{T}_{sim}^{(l)} \\ &= 2 - \tau - 2\tau \\ &= 2 - 3\tau \end{aligned} \tag{17}$$

if $\tau \in (0, \frac{2}{3})$, we have $3\tau < 2$. Therefore,

$$D(c_1, c_2) \geq 2 - 3\tau > 0 \tag{18}$$

Equation 18 indicates that the minimal distance between class c_1 and c_2 is always positive so that these two classes are separated.

□

3.3.3 Discussion. Compared to pairwise and triplet loss, the proposed ABTL has several advantages:

- Similar to triplet loss, ABTL simultaneously evaluates both similarity and dissimilarity errors to obtain a high-quality metric.
- The similarity threshold $d_{sim}^{(l)}(x_t, x_t^+)$ is always less than or equal to $D_{orig}^{(l)}(x_t, x_t^+)$ and the dissimilarity threshold $d_{dis}^{(l)}(x_t, x_t^-)$ is always greater than or equal to $D_{orig}^{(l)}(x_t, x_t^-)$. Thus, every input constraint contributes to model update, which eliminates the failure cases of triplet loss and leads to a higher constraint utilization rate.
- The optimal range of τ is provided with a theoretical proof.

3.4 Adaptive Hedge Update (AHU)
With adaptive-bound triplet loss, the overall loss in OAHU becomes

$$\begin{aligned} \mathcal{L}_{overall}(x_t, x_t^+, x_t^-) &= \sum_{l=0}^{L} \alpha^{(l)} \cdot \mathcal{L}^{(l)}(x_t, x_t^+, x_t^-) \\ &= \sum_{l=0}^{L} \frac{\alpha^{(l)}}{2}\Big(\mathcal{L}_{attr}^{(l)}(x_t, x_t^+) + \mathcal{L}_{rep}^{(l)}(x_t, x_t^-)\Big) \end{aligned} \tag{19}$$

We propose to learn $\alpha^{(l)}$ using the Hedge Algorithm [9]. Initially, all weights are uniformly distributed, i.e., $\alpha^{(l)} = \frac{1}{L+1}, l = 0, 1, 2, \ldots, L$. At every iteration, for each metric model E_l, OAHU transforms the input triplet constraint into E_l's metric embedding space, where the constraint similarity and dissimilarity errors are

evaluated. Thus $\alpha^{(l)}$ is updated based on the local loss suffered by E_l as:

$$\alpha_{t+1}^{(l)} \leftarrow \begin{cases} \alpha_t^{(l)} \beta^{\mathcal{L}^{(l)}} & \beta^{\min_l \mathcal{L}^{(l)}} \log \mathcal{L}^{(l)} > \beta - 1 \\ \alpha_t^{(l)}[1 - (1 - \beta)\mathcal{L}^{(l)}] & otherwise \end{cases} \quad (20)$$

where $\beta \in (0, 1)$ is the discount factor. We choose to update $\alpha^{(l)}$ based on Equation 20, to maximize the update of $\alpha^{(l)}$ at each step. In other words, if a metric model produces a low/high $\mathcal{L}^{(l)}$ at time t, its associated weight $\alpha^{(l)}$ will gain as much increment/decrement as possible. Therefore, in OAHU, metric models with good performance are highlighted while models with poor performance are eliminated. Moreover, following Sahoo et.al [21], to avoid the model bias issue [7, 10], i.e., OAHU may unfairly prefer shallow models due to their faster convergence rate and excessively reduces weights of deeper models, we introduce a smooth factor s used to set a minimal weight for each metric model. Thus, after updating weights according to Equation 20, the weights are set as: $\alpha_{t+1}^{(l)} = \max(\alpha_{t+1}^{(l)}, \frac{s}{L+1})$. Finally, at the end of every round, the weights α are normalized such that $\sum_{l=0}^{L} \alpha_{t+1}^{(l)} = 1$.

Learning the parameters $\Theta^{(l)}$ and $W^{(l)}$ is tricky but can be achieved via gradient descent. By applying Online Gradient Descent (OGD) algorithm, the update rules for $\Theta^{(l)}$ and $W^{(l)}$ are given by:

$$\begin{aligned} \Theta_{t+1}^{(l)} &\leftarrow \Theta_t^{(l)} - \eta \nabla_{\Theta_t^{(l)}} \mathcal{L}_{overall}(x_t, x_t^+, x_t^-) \\ &= \Theta_t^{(l)} - \eta \frac{\alpha^{(l)}}{2} \left(\nabla_{\Theta_t^{(l)}} \mathcal{L}_{attr}^{(l)}(x_t, x_t^+) + \nabla_{\Theta_t^{(l)}} \mathcal{L}_{rep}^{(l)}(x_t, x_t^-) \right) \end{aligned} \quad (21)$$

$$\begin{aligned} W_{t+1}^{(l)} &\leftarrow W_t^{(l)} - \eta \nabla_{W_t^{(l)}} \mathcal{L}_{overall}(x_t, x_t^+, x_t^-) \\ &= W_t^{(l)} - \eta \sum_{j=l}^{L} \frac{\alpha^{(j)}}{2} \left(\nabla_{W_t^{(l)}} \mathcal{L}_{attr}^{(j)}(x_t, x_t^+) + \nabla_{W_t^{(l)}} \mathcal{L}_{rep}^{(j)}(x_t, x_t^-) \right) \end{aligned} \quad (22)$$

Updating $\Theta^{(l)}$ (Equation 21) is simple since $\Theta^{(l)}$ is specific to E_l, and is not shared across different metric models. In contrast, because deeper models share low-level knowledge with shallower models in OAHU, all models deeper than l layers contribute to the update of $W^{(l)}$, as expressed in Equation 22. Algorithm 1 outlines the training process of OAHU.

3.5 Regret Bound

Overall, hedge algorithm enjoys a regret bound of $R_T \le \sqrt{T \ln N}$ [9] where N is the number of experts and T is the number of trials. In our case, N is the number of metric models (i.e., $L + 1$) and T is the number of input triplet constraints. The average worst-case convergence rate of OAHU is then given by $O(\sqrt{\ln(N)/T})$. Therefore, OAHU is guaranteed to converge after learning from sufficiently many triplets.

3.6 Application of OAHU

In this part, we discuss the deployment of OAHU in different practical applications. Although OAHU can be applied in many applications, we mainly focus on the three most common tasks: classification, similarity comparison, and analogue retrieval. Note that the

---

Algorithm 1 OAHU: Online Metric Learning with Adaptive Hedge Update

Require: Discount Factor $\beta \in (0, 1)$; Smooth Factor s; Control Parameter $\tau \in (0, \frac{2}{3})$; Learning rate η; A randomly initialized ANN with L hidden layers that is parameterized by $\Theta^{(l)}, W^{(l)}$ and $\alpha^{(l)}$.

Ensure: $\alpha^{(l)}, \Theta^{(l)}$ and $W^{(l)}$

1: Initialize $\alpha^{(l)} = \frac{1}{L+1}, \forall l = 0, 1, \ldots, L$
2: **for** $t = 1, 2, \ldots, T$ **do**
3:     Receive a triplet constraint (x_t, x_t^+, x_t^-)
4:     Transform and retrieve the metric embedding $f^{(l)}(x_t)$, $f^{(l)}(x_t^+)$ and $f^{(l)}(x_t^-)$ from each model E_l.
5:     Evaluate $\mathcal{L}_{attr}^{(l)}(x_t, x_t^+)$ and $\mathcal{L}_{rep}^{(l)}(x_t, x_t^-)$ (Eq. 12), $\forall l = 0, 1, \ldots, L$.
6:     Compute $L^{(l)}(x_t, x_t^+, x_t^-), \forall l = 0, 1, \ldots, L$ as per Eq. 13.
7:     Update $\Theta_{t+1}^{(l)}, \forall l = 0, 1, 2, \ldots, L$ as per Eq. 21.
8:     Update $W_{t+1}^{(l)}, \forall l = 0, 1, 2, \ldots, L$ as per Eq. 22.
9:     Update $\alpha_{t+1}^{(l)}, \forall l = 0, 1, 2, \ldots, L$ as per Eq. 20.
10:     $\alpha_{t+1}^{(l)} = \max(\alpha_{t+1}^{(l)}, \frac{s}{L+1}), \forall l = 0, 1, 2, \ldots, L$.
11:     Normalize $\alpha_{t+1}^{(l)}$, i.e., $\alpha_{t+1}^{(l)} = \frac{\alpha_{t+1}^{(l)}}{\sum_{j=0}^{L} \alpha_{t+1}^{(j)}}, \forall l = 0, 1, 2, \ldots, L$.
12: **end for**

---

following discussion describes only one possible way of deployment for each task, and other deployment methods can exist.

3.6.1 Classification. Given a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathcal{R}^d$ is an training instance and $y_i \in \{1, 2, \ldots, c\}$ is its associated label, we determine the label $y \in \{1, 2, \ldots, c\}$ of a test instance $x \in \mathcal{R}^d$ as follows:

- For every metric model E_l in OAHU, we find k nearest neighbors of x measured by Eq. 6, each of which is a candidate similar to x. Hence, $k(L + 1)$ candidates in total are found in OAHU. Let $\mathcal{N} = \{\{(x_j^{(l)}, y_j^{(l)})\}_{j=1}^k\}_{l=0}^L = \{\mathcal{N}_l\}_{l=0}^L$ denotes these candidates where $x_j^{(l)}$ is the jth neighbor of x found in E_l and $y_j^{(l)}$ is its associated label. The similarity score between x and $x_j^{(l)}$ is given by

$$S(x, x_j^{(l)}) = e^{-\frac{D^{(l)}(x_j^{(l)}, x) - d_{min}}{d_{max} - d_{min}}} \cdot \alpha^{(l)} \quad (23)$$

where $d_{min} = \min_{\mathcal{N}_l} D^{(l)}(x_j^{(l)}, x)$ and $d_{max} = \max_{\mathcal{N}_l} D^{(l)}(x_j^{(l)}, x)$.
- Then, the class association score of each class c_i is calculated by:

$$S(c_i) = \sum_{y_j^{(l)} = c_i} S(x, x_j^{(l)}) \quad (24)$$

- The final predicted label y of x in OAHU is determined as:

$$y = \arg\max_{c_i} S(c_i) \quad (25)$$

3.6.2 Similarity Comparison. The task of similarity comparison is to determine whether a given pair (x_1, x_2) is similar or not. To

do this, a user-specified threshold $\mathcal{T} \in (0, 1)$ is needed. The steps below are followed to perform the similarity comparison in OAHU.

- First, for every metric model E_l, we compute the distance $D^{(l)}(x_1, x_2)$ between x_1 and x_2. Since $D^{(l)}(x_1, x_2) \in [0, 2]$, we divide $D^{(l)}(x_1, x_2)$ by 2 for normalization.
- The similarity probability of (x_1, x_2) is then determined by

$$P(x_1, x_2) = \sum_{l=0,\ldots,L} \alpha^{(l)} \cdot p_l \qquad (26)$$

  where $p_l = 1$ if $D^{(l)}(x_1, x_2)/2 < \mathcal{T}$; Otherwise, $p_l = 0$.
- (x_1, x_2) is similar if $P(x_1, x_2) \geq 0.5$ and is dissimilar if $P(x_1, x_2) < 0.5$.

3.6.3 Analogue Retrieval. Given a query item x, the task of analogue retrieval is to find k items that are most similar to x in a database $\mathcal{D}$. Note that analogue retrieval is a simplified "classification" task where we only find similar items and ignore the labeling issue.

- We first retrieve the $k(L + 1)$ candidates that are similar to x following the same process in classification task. Let $\mathcal{N} = \{\{x_j^{(l)}\}_{j=1}^k\}_{l=0}^L = \{\mathcal{N}_l\}_{l=0}^L$ denote these candidates. The similarity score between $x_j^{(l)}$ and x is:

$$Sim(x_j^{(l)}, x) = e^{-\frac{D^{(l)}(x_j^{(l)}, x) - d_{min}}{d_{max} - d_{min}}} \cdot \alpha^{(l)} \qquad (27)$$

  where $d_{min} = \min_{\mathcal{N}_l} D^{(l)}(x_j^{(l)}, x)$ and $d_{max} = \max_{\mathcal{N}_l} D^{(l)}(x_j^{(l)}, x)$.
- Then we sort these items in descending order based on their similarity scores and keep only the top k distinct items as the final retrieval result.

3.7 Time and Space Complexity

Overall, the execution overhead of OAHU mainly arises from the training of the ANN-based metric model, especially from the gradient computation. Assume that the maximum time complexity for calculating the gradient of one layer is a constant C. The overall time complexity of OAHU is simply $O(nL^2C)$, where n is the number of input constraints and L is the number of hidden layers. Therefore, with pre-specified L and ANN architecture, OAHU is an efficient online metric learning algorithm that runs in linear time. Let d, S_{hidden} and S_{emb} denote the dimensionality of input data, number of units in each hidden layer and metric embedding size respectively. The space complexity of OAHU is given by $O\left(d \cdot S_{emb} + L \cdot S_{hidden}(d + S_{emb}) + \frac{L(L-1)}{2} S_{hidden}^2\right)$.

3.8 Extendability

As discussed in Section 3.6, OAHU is a fundamental module that can perform classification, similarity comparison, or analogue retrieval tasks independently. Therefore, it can be easily plugged into any existing deep models (e.g., a CNN or RNN) by replacing the classifier, i.e., fully-connected layers, with OAHU. The parameters of those feature-extraction layers (e.g., a convolutional layer) can be fine-tuned by optimizing $\mathcal{L}_{overall}$ directly. However, in this paper, to prove the validity of the proposed approach, we only focus on OAHU itself and leave the deep model extension for future work.

Task	Dataset	# features	# classes	# instances
	FASHION-MNIST	784	10	70,000
Image Classification	EMNIST	784	47	131,600
	CIFAR-10	1024	10	60,000
Face Verification	LFW	73	5749	13,233
Image Retrieval	CARS-196	4096	196	16,185
	CIFAR-100	4096	100	60,000

Table 2: Description of Datasets

4 EMPIRICAL EVALUATION

To verify the effectiveness of our method, we evaluate OAHU on three typical tasks, including (1) image classification (*classification*), (2) face verification (*similarity comparison*), and (3) image retrieval (*analogue retrieval*).

4.1 Baselines

To examine the quality of metrics learned in OAHU, we compare it with several state-of-the-art online metric learning algorithms. (1) LEGO [12] (pairwise constraint): an online metric learning algorithm that learns a Mahalanobis metric based on LogDet regularization and gradient descent. (2) RDML [14] (pairwise constraint): an online algorithm for regularized distance metric learning that learns a Mahalanobis metric with a regret bound. (3) OASIS [6] (triplet constraint): an online algorithm for scalable image similarity learning that learns a bilinear similarity measure over sparse representations. Finally, (4) OPML [19] (triplet constraint): a one-pass closed-form solution for online metric learning that learns a Mahalanobis metric with a low computational cost.

4.2 Implementation

We have implemented OAHU using Python 3.6.2 and Pytorch 0.4.0 library. All baseline methods were based on code released by corresponding authors, except LEGO and RDML. Due to the unavailability of a fully functional code of LEGO and RDML, we use our own implementation based on the authors' description [12, 14]. Hyperparameter setting of these approaches, including OAHU, varies with different tasks, and will be discussed later for each specific task.

4.3 Image Classification

4.3.1 Dataset and Classifier. To demonstrate the validacy of OAHU on complex practical applications, we use three publicly available real-world benchmark image datasets, including **FASHION-MNIST**[3] [26], **EMNIST**[4] [8] and **CIFAR-10**[5] [16] for evaluation. To be consistent with other image datasets, we convert images of CIFAR-10 into grayscale through the OpenCV API [3], resulting in 1024 features. The *k-NN* classifier is employed here, since it is widely used for classification tasks with only one parameter, and is also consistent with the deployment of OAHU discussed in Section 3.6.1. Details of these datasets are listed in "Image Classification" part of Table 2.

4.3.2 Experiment Setup and Constraint Generation. To perform the experiment, we first randomly shuffle each dataset and then divide

---

[3]https://github.com/zalandoresearch/fashion-mnist
[4]https://www.nist.gov/itl/iad/image-group/emnist-dataset
[5]https://www.cs.toronto.edu/ kriz/cifar.html

Methods	FASHION-MNIST			EMNIST			CIFAR-10			win/tie/loss
	Error Rate	F_1	$\mathcal{U}$	Error Rate	F_1	$\mathcal{U}$	Error Rate	F_1	$\mathcal{U}$	
LEGO	0.35±0.01 ●	0.64±0.00	0.58±0.00	0.84±0.00 ●	0.14±0.02	0.66±0.01	0.88±0.00 ●	0.05±0.02	0.59±0.00	3/0/0
RDML	0.22±0.01 ●	0.78±0.00	0.47±0.01	0.64±0.03 ●	0.32±0.05	0.50±0.00	0.76±0.01 ●	0.24±0.01	0.50±0.01	3/0/0
OASIS	0.21±0.02 ●	0.79±0.01	0.14±0.00	0.41±0.01 ●	0.59±0.00	0.24±0.00	0.79±0.02 ●	0.20±0.03	0.47±0.00	3/0/0
OPML	0.23±0.00 ●	0.78±0.00	0.30±0.01	0.43±0.01 ●	0.51±0.02	0.45±0.00	0.83±0.01 ●	0.16±0.00	0.48±0.04	3/0/0
OAHU	0.18±0.00	0.81±0.00	1.00±0.00	0.35±0.01	0.64±0.01	1.00±0.00	0.68±0.00	0.31±0.00	1.00±0.00	-

Table 3: Comparison of classification performance on competing methods over benchmark image datasets. ●/○ indicates OAHU performs statistically better/worse (0.05 significance level) than the respective method according to the p-values. The statistics of win/tie/loss is also included. Both mean and standard deviation of error rates, constraint utilization $\mathcal{U}$ and macro F_1 scores are reported. 0.00 denotes a value less than 0.005.
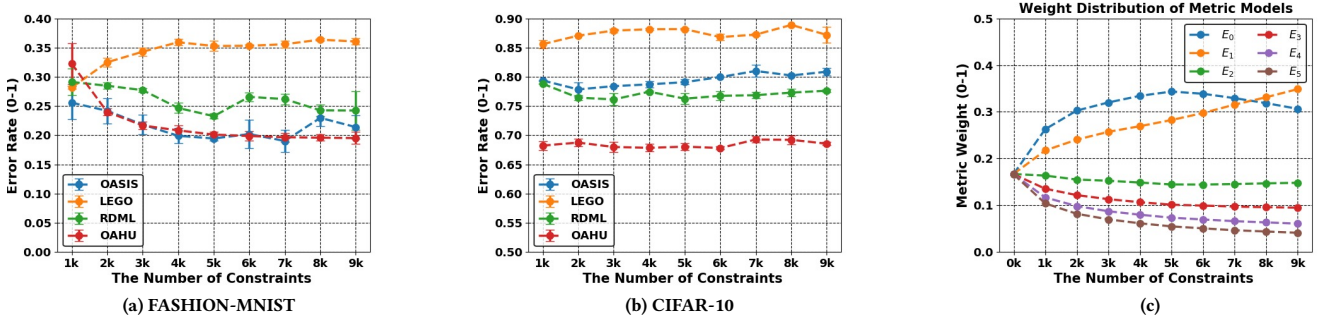

Figure 4: (a) / (b) Error rates of competing methods with increasing number of constraints on FASHION-MNIST / CIFAR-10 dataset. (c) The evolvement of metric weight distribution in OAHU with increasing number of constraints on FASHION-MNIST dataset.

it into development and test sets with the split ratio of 1 : 1. Hyperparameters of these baseline approaches were set based on values reported by the authors and fine-tuned via 10-fold cross-validation on the development set. Because OPML applies a one-pass triplet construction process to generate the constraints, we simply provide the training data in the development set as the input for OPML. For other approaches including OAHU, we first randomly sample 5,000 seeds (i.e., pairwise or triplet constraints) from the training data and then construct 5,000 more constraints by taking transitive closure over these seeds. Note that the same number of constraints were adopted by the authors of LEGO [12] and OPML [19] when comparing their approaches with baselines. Thus our comparison is fair. In OAHU, we set $\beta = 0.99$, $L = 5$, $S_{hidden} = 100$, $s = 0.1$, $\eta = 0.3$ and $S_{emb} = 50$ as default. The value of τ was found by 10-fold cross-validation on the development set. To reduce any bias that may result from random partition, all the classification results are averaged over 10 individual runs with independent random shuffle processes.

4.3.3 Evaluation Metric.
Similar to most of the baselines, e.g., LEGO [12] and OPML [19], we adopt the error rate and macro F_1 score[6] as the evaluation criterion. Moreover, the p-values of student's t-test were calculated to check statistical significance. In addition, the statistics of win/tie/loss are reported according to the obtained p-values. The constraint utilization $\mathcal{U}$, i.e., the fraction of input constraints that actually contribute to model update, is also reported for each method.

4.3.4 Analysis.
Table 3 compares the classification performance of OAHU with baseline methods on benchmark datasets. We observe that LEGO and RDML perform poorly on complex image datasets, e.g., EMNIST (47 classes), since they learn a Mahalanobis metric from pairwise constraints, and are biased to either the similarity or dissimilarity relation when updating model parameters. OASIS performs better than OPML because the bilinear metric eliminates the positive and symmetric requirement of the covariance matrix, i.e., A in $d(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T A \mathbf{x}_2$ which results in a larger hypothesis space [6]. However, OAHU outperforms all the baseline approaches by providing not only significantly lower error rate, but also higher F_1 score, indicating the best overall classification performance among all competing methods.

To illustrate the effect of different numbers of constraints on the learning of a metric, we vary the input constraint numbers[7] for all competing methods except OPML. Here we do not compare with OPML because it internally determines the number of triplets to use via a one-pass triplet construction process. Therefore, explicitly varying the input triplet number for OPML is impossible. Figure 4a and Figure 4b show the comparison of error rates on FASHION-MNIST and CIFAR-10 dataset respectively. The large variance of error rates observed in both OASIS and RDML (i.e., error bars in the figures) indicates the strong correlation of their classification performance with the quality of input constraints. All baseline approaches present an unstable, and sometimes, even worse performance with an increasing number of constraints. This is because these baselines are biased to those "hard" constraints

---

[6]http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[7]It is done by first generating 10,000 constraints as discussed in Section 4.3.2 and then uniformly sample the desired number of constraints from them.

leading to positive loss values and incorrectly ignore the emerging failure cases (see Figure 2). In contrast, OAHU provides the best and most stable classification performance with minimal variance, indicating its robustness to the quality of input constraints and better capability of capturing the semantic similarity and dissimilarity relations in data.

OAHU is better mainly because it not only learns from every input constraint by optimizing the adaptive-bound triplet loss (see constraint utilization $\mathcal{U}$ in Table 3), but also automatically adapts the model complexity to achieve better performance (see Figure 4c).

4.4 Face Verification

4.4.1 Dataset and Experiment Setup. For face verification, we evaluate our method on Labeled Faces in the Wild Database (LFW) [18]. This dataset has two views: View 1 is used for development purpose (contains a training and a test set), and View 2 is taken as a benchmark for comparison (i.e., a 10-fold cross validation set containing pairwise images). The goal of face verification in LFW is to determine whether a pair of face images belongs to the same person. Since OASIS, OPML, and OAHU are triplet-based methods, following the setting in OPML [19], we adopt the image unrestricted configuration to conduct experiments for a fair comparison. In the image unrestricted configuration, the label information (i.e. actual names) in training data can be used and as many pairs or triplets as one desires can be formulated. We use View 1 for hyper-parameter tuning and evaluate the performance of all competing algorithms on each fold (i.e., 300 intra-person and 300 inter-person pairs) in View 2. Note that the default parameter values in OAHU and baselines were set using the same values as discussed in Section 4.3.2. Attribute features of LFW are used in the experiment so that the result can be easily reproduced and the comparison is fair. Attribute features[8] (73 dimension) provided by Kumar et al. [17] are "high-level" features describing nameable attributes such as race, age etc., of a face image. Details of LFW are described in "Face Verification" part in Table 2.

4.4.2 Constraint Generation and Evaluation Metric. We follow the same process described in Section 4.3.2 to generate constraints for training purpose. However, we add an additional step here: if a generated constraint contains any pair of instances in the validation set (i.e., all pairs in View 1) or the test set (i.e., all folds in View 2), we simply discard it and re-sample a new constraint. This step is repeated until the new constraint does not contain any pair in both sets. In the experiment, 10, 000 pair or triplet constraints in total are generated for training.

Note, due to the one-pass triplet generation strategy of OPML, we let itself determine the training triplets. For each test image pair, we first compute the distance between them measured by the learned metric, and then normalize the distance into [0, 1]. The widely accepted Receiver Operating Characteristic (ROC) curve and Area under Curve (AUC) are adopted as the evaluation metric.

4.4.3 Analysis. The ROC curves of all competing methods are given in Figure 5a with corresponding AUC values calculated. It can be observed that the proposed approach OAHU outperforms all baseline approaches by providing significantly higher AUC value.

[8]http://www.cs.columbia.edu/CAVE/databases/pubfig/download/lfw_attributes.txt
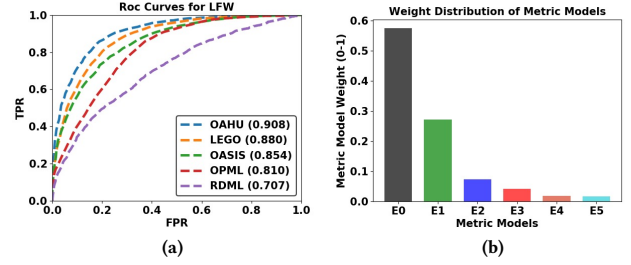
Figure 5: (a) ROC Curves of OAHU and contrastive methods on LFW dataset. AUC value of each method is given in bracket. (b) Metric weight distribution of OAHU.
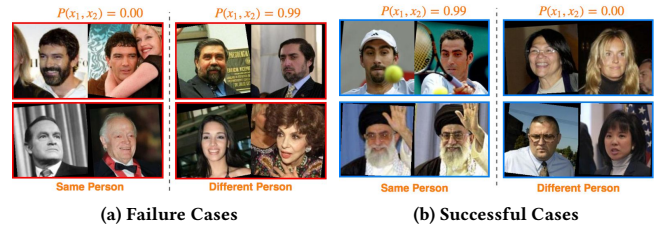

(a) Failure Cases   (b) Successful Cases

Figure 6: Face verification examples in the test set (View 2) ($\mathcal{T}$ = 0.55). For images belonging to same person, $P(x_1, x_2)$ should be close to 1 and for images belonging to different people, $P(x_1, x_2)$ should drop to 0.

As demonstrated in Figure 5b, OAHU is better mainly because it automatically adapts the model complexity to incorporate non-linear models, which help to separate instances that are difficult to distinguish in simple linear models. Figure 6 presents some example comparisons in the test set for both successful and failure cases. Here, the comparison probability $P(x_1, x_2)$ computed based on Eq. 26 is expected to approach 1 if both images are from the same person and should be close to 0 if they belong to different people. Apparently, although OAHU performs much better than baselines, it may still fail in some cases where severe concept drift, e.g. aging and shave, are present.

4.5 Image Retrieval

4.5.1 Dataset and Experiment Setup. To demonstrate the performance of OAHU on image retrieval task, we show experiments on two benchmark datasets, which are CARS-196[9] [15] and CIFAR-100[10] [16]. The CARS-196 dataset is divided into 8, 144 training images and 8, 401 test images, where each class has been split roughly in a 50-50 ratio. Similarly, we randomly split CIFAR-100 into 30, 000 training images and 30, 000 test images by equally dividing images of each class. For each image in CARS-196 and CIFAR-100, its deep feature, i.e., the output of the last convolutional layer, is extracted from a VGG-19 model pretrained on the ImageNet [20] dataset. Details of these 2 datasets are listed in the "Image Retrieval" part of Table 2. We form the development set of each dataset by including

[9]https://ai.stanford.edu/ jkrause/cars/car_dataset.html
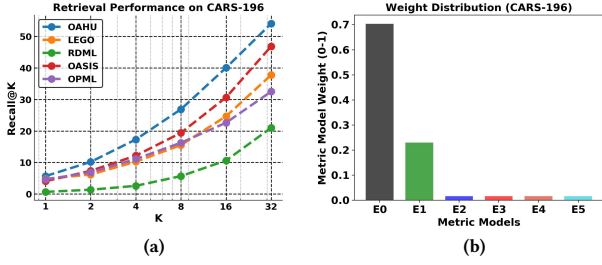[10]https://www.cs.toronto.edu/ kriz/cifar.html

Figure 7: (a) Recall@K score on the test split of CARS-196. (b) Metric weight distribution of OAHU on CARS-196.
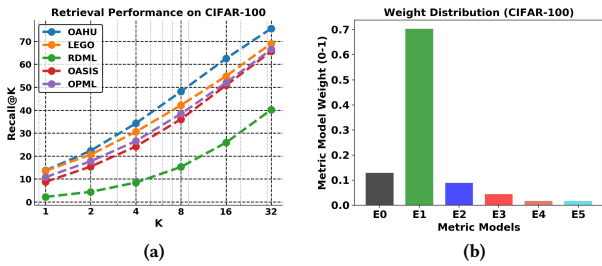

Figure 8: (a) Recall@K score on the test split of CIFAR-100. (b) Metric weight distribution of OAHU on CIFAR-100.

all its training images. For each dataset, hyper-parameters of baseline approaches were set based on values reported by the authors, and fine-tuned via 10-fold cross-validation on the development set. In OAHU, we set $\beta = 0.99$, $L = 5$, $S_{hidden} = 100$, $s = 0.1$, $\eta = 0.3$ and $S_{emb} = 50$ as default. The value of τ in OAHU was found via 10-fold cross-validation on the development set.

4.5.2 Constraint Generation and Evaluation Metric. With the training images in the development set, we generate the training constraints in the same way as Section 4.3.2. However, in this experiment, we sample $50,000$ constraints ($25,000$ seeds) for each dataset due to the vast amount of classes included in them. The Recall@K metric [13] is applied for performance evaluation. Each test image (query) first retrieves K most similar images from the test set and receives a score of 1 if an image of the same class is retrieved among these K images and 0 otherwise. Recall@K averages this score over all the images.

4.5.3 Analysis. The Recall@K scores on the test set of CARS-196 and CIFAR-100 are shown in Figure 7a and Figure 8a, respectively. We observe that OAHU outperforms all baseline approaches by providing significantly higher Recall@K scores. In addition, the Recall@K score of OAHU grows more rapidly than that provided by most baselines. Figure 7b and Figure 8b illustrate the metric weight distribution of OAHU on CARS-196 and CIFAR-100 respectively. Compared to the baselines that are only linear models, OAHU takes full advantage of non-linear metrics to improve the model expressiveness. Figure 9 shows some examples of successful and failure queries on both datasets. Despite the huge change in the viewpoint,
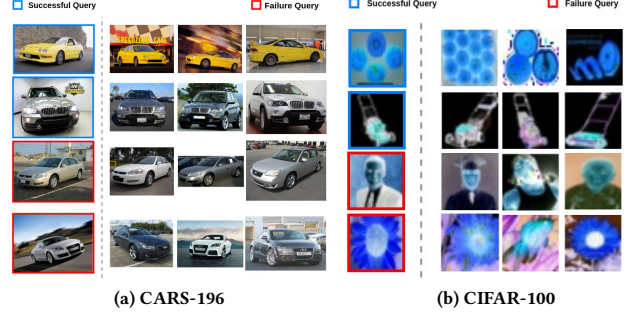

Figure 9: Examples of successful and failure queries on the CARS-196 and CIFAR-100 datasets using our embedding. Images in the first column are query images and the rest are three most similar images. Best viewed on a monitor zoomed in.

configuration, and illumination, our method can successfully retrieve examples from the same class and most retrieval failures come from subtle visual difference among images of different classes.

4.6 Sensitivity of Parameters

The three main parameters in OAHU is the control parameter τ. We vary τ from 0.1 to 0.9 and S_{emb} from 10 to 100 to study their effect on the classification performance. We observe that both classification accuracy (1-error rate) and F_1 score significantly drop if τ is greater than 0.6 but remain unchanged with various τ values if $\tau \leq 0.6$. This observation verifies Theorem 3.1 which states that classes are separated if $\tau \in (0, \frac{2}{3})$. On the other hand, the classification accuracy of OAHU peaks at embedding size $S_{emb} = 50$ where all semantic information are maintained. If we further increase the embedding size, noisy information is going to be introduced which degrades the classification performance. Therefore, we suggest to choose $\tau = 0.1$ and $S_{emb} = 50$ as default.

5 LIMITATIONS AND CONCLUSIONS

In this paper, we propose a novel online metric learning framework, called OAHU, that learns a ANN-based metric with adaptive complexity and a full constraint utilization rate. Compared to the state-of-the-art baselines, OAHU not only obtains the best performance by automatically adapting its complexity to improve the model expressiveness, but also is more robust in terms of the quality of input constraints. Like existing state-of-the-art solutions, OAHU fails when facing severe concept drift data that invalidate the learned similarity metric. We leave the extension of OAHU to severe concept drift scenarios for future work.

6 ACKNOWLEDGMENTS

REFERENCES

[1] Charu C. Aggarwal, Zhi-Hua Zhou, Alexander Tuzhilin, Hui Xiong, and Xindong Wu (Eds.). 2015. *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*. IEEE Computer Society. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7373198

[2] Aurélien Bellet, Amaury Habrard, and Marc Sebban. 2013. A Survey on Metric Learning for Feature Vectors and Structured Data. *CoRR* abs/1306.6709 (2013). arXiv:1306.6709 http://arxiv.org/abs/1306.6709

[3] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[4] Casey Breen, Latifur Khan, and Arunkumar Ponnusamy. 2002. Image classification using neural networks and ontologies. In *Proceedings. 13th International Workshop on Database and Expert Systems Applications*. IEEE, 98–102.

[5] Carla E. Brodley and Peter Stone (Eds.). 2014. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. AAAI Press. http://www.aaai.org/Library/AAAI/aaai14contents.php

[6] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. 2010. Large Scale Online Learning of Image Similarity Through Ranking. *Journal of Machine Learning Research* 11 (2010), 1109–1135. https://doi.org/10.1145/1756006.1756042

[7] Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. 2015. Net2Net: Accelerating Learning via Knowledge Transfer. *CoRR* abs/1511.05641 (2015). arXiv:1511.05641 http://arxiv.org/abs/1511.05641

[8] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. EMNIST: an extension of MNIST to handwritten letters. *CoRR* abs/1702.05373 (2017). arXiv:1702.05373 http://arxiv.org/abs/1702.05373

[9] Yoav Freund and Robert E. Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* 55, 1 (1997), 119–139. https://doi.org/10.1006/jcss.1997.1504

[10] Çaglar Gülçehre, Marcin Moczulski, Francesco Visin, and Yoshua Bengio. 2016. Mollifying Networks. *CoRR* abs/1608.04980 (2016). arXiv:1608.04980 http://arxiv.org/abs/1608.04980

[11] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. 2016. Deep Networks with Stochastic Depth. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*. 646–661. https://doi.org/10.1007/978-3-319-46493-0_39

[12] Prateek Jain, Brian Kulis, Inderjit S. Dhillon, and Kristen Grauman. 2008. Online Metric Learning and Fast Similarity Search. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*. 761–768. http://papers.nips.cc/paper/3446-online-metric-learning-and-fast-similarity-search

[13] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 1 (2011), 117–128. https://doi.org/10.1109/TPAMI.2010.57

[14] Rong Jin, Shijun Wang, and Yang Zhou. 2009. Regularized Distance Metric Learning: Theory and Algorithm. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*. 862–870. http://papers.nips.cc/paper/3703-regularized-distance-metric-learningtheory-and-algorithm

[15] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3D Object Representations for Fine-Grained Categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia.

[16] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[17] Neeraj Kumar, Alexander C. Berg, Peter N. Belhumeur, and Shree K. Nayar. 2009. Attribute and simile classifiers for face verification. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*. 365–372. https://doi.org/10.1109/ICCV.2009.5459250

[18] Gary B. Huang Erik Learned-Miller. 2014. *Labeled Faces in the Wild: Updates and New Reporting Procedures*. Technical Report UM-CS-2014-003. University of Massachusetts, Amherst.

[19] Wenbin Li, Yang Gao, Lei Wang, Luping Zhou, Jing Huo, and Yinghuan Shi. 2018. OPML: A one-pass closed-form solution for online metric learning. *Pattern Recognition* 75 (2018), 302–314. https://doi.org/10.1016/j.patcog.2017.03.016

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y

[21] Doyen Sahoo, Quang Pham, Jing Lu, and Steven C. H. Hoi. 2018. Online Deep Learning: Learning Deep Neural Networks on the Fly. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. 2660–2666. https://doi.org/10.24963/ijcai.2018/369

[22] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y. Ng. 2004. Online and batch learning of pseudo-metrics. In *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*. https://doi.org/10.1145/1015330.1015376

[23] Blake Shaw, Bert C. Huang, and Tony Jebara. 2011. Learning a Distance Metric from a Network. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*. 1899–1907. http://papers.nips.cc/paper/4392-learning-a-distance-metric-from-a-network

[24] Suraj Srinivas and Radhakrishnan Venkatesh Babu. 2016. Learning Neural Network Architectures using Backpropagation. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. http://www.bmva.org/bmvc/2016/papers/paper104/index.html

[25] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. 2005. Distance Metric Learning for Large Margin Nearest Neighbor Classification. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*. 1473–1480. http://papers.nips.cc/paper/2795-distance-metric-learning-for-large-margin-nearest-neighbor-classification

[26] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:cs.LG/cs.LG/1708.07747