

Learning Hypergraph-regularized Attribute Predictors

Sheng Huang[†] Mohamed Elhoseiny[‡] Ahmed Elgammal[‡] Dan Yang^{†,*}
[†]Chongqing University, P.R.China [‡]Rutgers University, USA
{huangsheng, dyang}@cqu.edu.cn {mhe19, elgammal}@cs.rutgers.edu

Abstract

We present a novel attribute learning framework named *Hypergraph-based Attribute Predictor (HAP)*. In HAP, a hypergraph is leveraged to depict the attribute relations in the data. Then the attribute prediction problem is casted as a regularized hypergraph cut problem in which HAP jointly learns a collection of attribute projections from the feature space to a hypergraph embedding space aligned with the attribute space. The learned projections directly act as attribute classifiers (linear and kernelized). This formulation leads to a very efficient approach. By considering our model as a multi-graph cut task, our framework can flexibly incorporate other available information, in particular class label. We apply our approach to attribute prediction, Zero-shot and N -shot learning tasks. The results on AWA, USAA and CUB databases demonstrate the value of our methods in comparison with the state-of-the-art approaches.

1. Introduction

Attribute learning aims at achieving an intermediate representation on top of the low-level visual feature space, which encodes semantic properties shared across different categories of objects or scenes. Such an intermediate representation plays a role as the vehicles of semantics in human-machine communication. Farhadi *et al.* [8] and Lampert *et al.* [18] showed that supervised attributes can be transferred across object categories, allowing description and naming of objects from categories not seen during training. Therefore, attributes provide a way to encode and share knowledge to achieve challenging tasks such as the Zero-Shot Learning (ZSL) problem, where the goal is to categorize classes that are unseen during training [18, 17].

A lot of approaches have been proposed for attribute learning, *e.g.* [18, 8, 27, 21, 1, 14, 10, 20]. Two fundamental issues remain unsolved, although recent researches have started to pay attention to them *e.g.* [1, 14, 10]. First, traditional approaches learn attributes independently from each other (one-vs-all classifiers) [18, 8], without explicitly exploiting the correlation between attributes. Second,

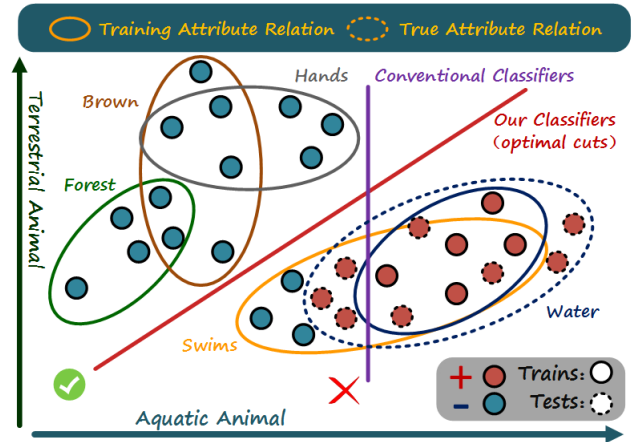


Figure 1. The visualization of the hypergraph cut for predicting the attribute "water". Each ellipse with solid lines denotes a hyperedge that encodes an attribute relation and each circle represents a sample.

learning attribute classifiers are typically done independent of the subsequent tasks, such as categorization or zero shot learning, and typically category labels are ignored in the learning process. Optimizing the attribute prediction independent of the succeeding task does not guarantee to yield the best attribute predictor for that task.

Correlations naturally exist among attributes. Is it better to exploit the correlation or to discourage the correlation during attribute learning, *i.e.* decorrelation? Several papers have argued that exploiting correlation between attributes improves their discriminative powers, *e.g.* [31, 28]. Recent works attempted to address the issue of joint attribute learning with a focus on decorrelating attributes [1, 14]. Jayaraman *et al.* [14] argued that attribute learning approaches are prone to learn visual features that correlates with attributes, not attributes themselves. Therefore they argued for decorrelation of attributes, by exploiting feature competition during learning through a multitask learning framework. Decorrelation of attributes might be suitable in tasks such as describing images with text, key-word based retrieval, or generating image annotations. However, preserving and exploiting correlation between attributes should

preserve the natural clustering in the data and should be better for classification and zero-shot learning tasks. Correlation is a nature of attributes and compulsively decorrelating the attributes may break the original relations of attributes in the visual space as well as in the semantic space.

To illustrate our point we use the example in Figure 1 where we used attributes from the AWA dataset [18]. Consider learning the attribute “water”. In the figure, there are two clusters denoted as two superclasses (terrestrial and aquatic animals). It is expected that the attributes in each cluster will be highly correlated and coexist in images. The conventional classifiers directly learn an optimal separation for the attribute “water” which clearly ignores the correlations as well as the natural clusters in the data. Although it achieves the optimal attribute prediction, it clearly reduces the utility of attribute predictors in subsequent tasks such as categorization, and is much easier to get mixed in overfitting. Instead we aim at a cut that, besides minimizing the attribute prediction loss, tries to preserve the clustering in the data. Preserving the correlation can be even more beneficial for attributes that are not visual. For example the attribute “swim” describes an action and it is very hard to predict it from visual features if it is forced to be decorrelated from other attributes such as “water”.

Our goal is to design a new attribute learning framework that addresses the two aforementioned issues, *i.e.* jointly learning attributes while exploiting the correlation between attributes, and exploiting class information as well as any available side information. We propose to model the attribute learning as a supervised hypergraph cut problem. As a generalization of graphs, hypergraphs are typically used to depict the high-order and multiple relations of data [35, 5, 15, 29]. One merit of hypergraph is that it can capture the correlations of multiple relations, since the partition of a vertex set who has many common hyperedges will lead to a heavy penalty [29]. In our formulation, we define a hypergraph where each vertex corresponds to a sample and a hyperedge is a vertex set sharing the same attribute label. Then, we can consider the attribute prediction problem as a hypergraph cut problem. More specifically, a collection of hypergraph cuts (one cut per attribute) that minimizes the loss of attribute relations (defined by the hyperedges) is jointly learned. Moreover, such cuts also minimize the attribute prediction errors of training data. Since hypergraph cuts can be deemed as the hypergraph embedding from the perspective of graph embedding [26, 3, 32], this step actually tries to align the embedding space, which encodes the attribute relations, with the semantic attribute space. We also propose attribute predictors (or classifiers) that can be obtained by introducing a mapping from the feature space to this aligned hypergraph embedding space. We name this approach Hypergraph-based Attribute Predictor (HAP), which can be combined with different hypergraph

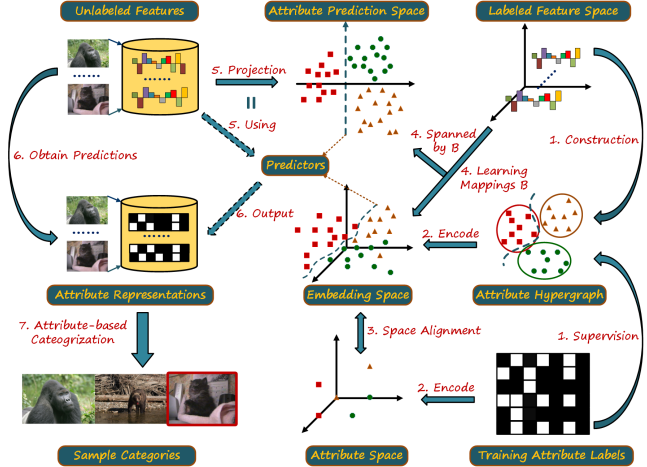


Figure 2. The overview of our approach, we learn a collection of the mappings (projections) from the feature space to the hypergraph embedding space which is aligned by the attribute space and encode the attribute relations. The learned projections span the Attribute Prediction Space (APS) in which each basis is an attribute predictor. The attribute prediction of a sample can be achieved by projection it to the APS and then the attribute-based categorization can be performed.

models to obtain classifiers from the cuts. We illustrate our model in Figure 2.

In order to incorporate class information and any additional side information within the HAP formulation, we consider it as a multi-graph cut problem. One or several additional graphs (or hypergraphs), which encode side information, will be introduced as the penalties to the HAP. In that case, the new cuts should not only minimize the loss of attribute relations, but also the losses of the side information. In case of class labels, we formulate a new HAP framework, denoted as Class-Specific HAP (CSHAP), which enhances the discriminating ability of the attribute predictors. A hypergraph and a graph, which encode the class information in two different ways, are respectively leveraged to produce two different versions of the CSHAP approaches. We denote them Hypergraph-based CSHAP (CSHAP_H) and Graph-based CSHAP (CSHAP_G) respectively. Finally, all the proposed approaches are kernelized to incorporate the nonlinearity.

We summarize the contributions of this paper as follows:

1. As far as we know, our approach is the first to formulate attribute learning as a supervised hypergraph cut problem.
2. We propose an approach to construct predictors (linear and nonlinear classifier) jointly while solving for the cuts. This idea is applicable in general (not limited to the context of attribute learning) to any hypergraph cut algorithm, as a general path to derive the classifiers from graph model.
3. We provide a flexible solution to incorporate the side

information in attribute learning.

4. The proposed approach provides efficient attribute predictions, since the computational complexity of the attribute prediction is linear with respect to the dimension of feature.

We experimented with three datasets: Animal With Attributes (AWA) [18], Caltech-UCSD Birds (CUB) [30] and Unstructured Social Activity Attribute (USAA) [10]. The results on attribute prediction, Zero-shot, N-shot Learning, and categorization consistently validate the effectiveness of the proposed framework. The rest of paper is organized as follows: Section 2 presents the related works; Section 3 describes the proposed approach. Section 4 shows the experimental evaluation of our works; the conclusion is summarized in Section 5.

2. Related Works

2.1. Attributes

Traditional attribute learning approaches follow a supervised discriminative learning pipeline (one-vs-all classifiers) where attribute classifiers are learned independently given attribute labels for each image or each class [8, 18]. Recently several papers suggested approaches for joint learning of attributes [19, 31, 1, 14]. Wang *et al.* [31] and Song *et al.* [28] construct a graph of attributes in attribute domain from the training data and consider it as the latent variables in the latent SVM for categorization, and they showed that exploiting attribute relations helps improve class prediction. In contrast our hypergraph is constructed on the samples, which facilitates aligning the feature space with the attribute semantic space.

Mahajan *et al.* [19] proposed a joint learning framework that removes the correlations as the redundancies during learning the mapping between attributes and classes which actually ignores the contributions of the co-occurrence attributes. Akata *et al.* [1] proposed an approach that simultaneously target three problems: optimizing attribute prediction using class labels, using side information, and incremental learning. They achieved decorrelation by using dimensionality reduction on the class-attribute matrix, *i.e.* in the label space, and showed that the attribute dimensions can be reduced significantly without affecting the accuracy. In contrast our hypergraph construction achieve correlation/decorrelation by employing the sample-attribute relations and embedding the data samples in a space that is aligned with the attribute labels. Recently, Jayaraman *et al.* [14] attempted to decorrelate the attributes via solving a structure sparsity model in which semantic attribute groups are manually provided as auxiliary data.

Attribute learning is just the preliminary step for some other visual tasks. Few approaches have been proposed to incorporate the additional information, in particular class labels into attribute learning for benefiting the subsequent

tasks [8, 1, 31, 4, 33]. However, the attribute learning and the exploitation of the additional information are highly coupled in these approaches. It is very hard to add novel additional information into the model, or unplug the additional information exploitation part from the original models when the additional information of the given data is not available. In contrast, our proposed method can flexibly to address this issue by considering the attribute prediction task as multi-graph (hypergraph) cut problem, enabling adding any side information as an extra graph or hypergraph.

Zero-Shot Learning (ZSL) is the task of object recognition for categories with no training examples [18]. Several intermediate representations has been used for ZSL, including attributes [18, 8, 17, 1], linguistic knowledge [9, 23], textual description [7] and visual abstraction [2]. The core attribute-based ZSL approaches are the attribute learning. Therefore our attribute prediction can be integrated to several ZSL frameworks. Although our work is an attribute-based ZSL, other intermediate representation can also be readily plugged into the HAP framework to replace the attributes, since it generally provides a mapping from the low-level representation to intermediate representation. Several ZSL approaches can be extended to N-Shot Learning [34, 1, 9, 25].

2.2. Hypergraph Learning

Hypergraph is a generalization of the regular graph which has been widely applied to depict the high-order relations between data points [35, 5, 15, 29]. Since the attribute prediction task can be regarded as a multi-label classification problem, we will introduce not only the relevant hypergraph model, but also some hypergraph-based multi-label classification algorithms, which motivated our work. More specifically, Zhou *et al.* [35] proposed a normalized hypergraph model for embedding and transduction. Our formulation is based on Zhou’s model, however it is inductive. Moreover, we show how the hypergraph cut can be reformulated to provide direct linear or nonlinear class predictors. Chen *et al.* [5] leveraged the hypergraph to capture the correlation of categories and introduced it as a regularization to SVM model for multi-label classification. Similar to [5], Sun *et al.* [29] used the hypergraph to capture the correlation of classes and performed a hypergraph embedding as the new representation for multi-class classification. In [15] the hypergraph is utilized to measure the loss of multi-labels during the multi-kernel learning. In contrast to all the existing hypergraph learning algorithms, as far as we know, our model is the first approach that directly derives the multi-label classifier from the hypergraph embedding.

3. Approach

3.1. Preliminaries

We start by reviewing some basic definitions of hypergraphs and introducing the notations. Hypergraphs are a

generalization of graphs in which a hyperedge (the analogy of an edge) is an arbitrary non-empty subsets of the vertex set (Fig 1 shows a hypergraph with five hyperedges). Given a hypergraph $G = (V, E)$ in an arbitrary feature space, V and E are the vertex set and hyperedge set, where each vertex and hyperedge are respectively defined as $v \in V$ and $e \in E$. Moreover, a hyperedge e is a subset of V . The vertex-edge incidence matrix $H \in \mathcal{R}^{|V| \times |E|}$ is defined as follows

$$h(v, e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The degree of a hyperedge e , which is denoted as $\delta(e)$, is the number of vertices in e

$$\delta(e) = \sum_{v \in e} h(v, e), \quad (2)$$

and the degree of a vertex $v \in V$ is defined as follows

$$d(v) = \sum_{e \in E} w(e) = \sum_{e \in E} w(e)h(v, e), \quad (3)$$

where $w(e)$ is the weight of the hyperedge e . We denote the diagonal matrix forms of $\delta(e)$, $d(v)$ and $w(e)$ as D_e , D_v and W respectively. Note, here we defined H as a binary matrix for simplicity. For the continuous value case, a probabilistic hypergraph model [12] can be adopted in which each element of H denotes the probability of a vertex in a hyperedge.

3.2. Attribute Hypergraph

In our model, we define a hypergraph to depict the attribute relations of samples (corresponding to images in the training set). In this hypergraph, the vertex $v_i \in V$ is corresponding to the sample $x_i \in X$, which is the i -th column of the $d \times n$ -dimensional sample matrix X . Here, n is the number of samples and d is the dimension of the feature space. Each hyperedge is defined as a vertex set that shares the same attribute label. In such case, the number of hyperedges is equal to the number of attributes m and the $n \times m$ -dimensional matrix incident matrix H is exactly the attribute label matrix. The more common attributes among a set of images, the more hyperedges will exist between their corresponding vertices, and the stronger the link will be between these vertices. Therefore, break such a link will lead to a heavy penalty during the learning process. In this way, the hypergraph actually provides a natural way to capture the correlation/decorrelation of attributes. We regard the hyperedge e as a clique and consider the mean of the heat kernel weights of the pairwise edges in this clique as the hyperedge weight

$$w(e) = \frac{1}{\delta(e)(\delta(e) - 1)} \sum_{\{v_i, v_j\} \in e} \exp\left(-\frac{\|x_i - x_j\|^2}{\mu}\right). \quad (4)$$

Certainly, some other hyperedge weighing schemes can be also applied.

3.3. Hypergraph-based Attribute Predictor

Normalized hypergraph cut is often utilized to learn the high-order relation and correlation information. The main idea of our model stems from the hypergraph-based transduction which is regarded as a regularized normalized hypergraph cut model [35]. In contrast, our method is a supervised inductive model. Since our proposed attribute predictor (or classifier) is based on the hypergraph model, we call it Hypergraph-based Attribute Predictor (HAP).

In HAP, a collection of hypergraph cuts $F = [f_1, \dots, f_m]$ is defined as the predictors of attributes in the feature space where m is the number of attributes and the cut f_i is a column vector whose elements are the predictions of i -th attribute for each sample. An optimal cut should not disrupt the hyperedges during hypergraph partition as much as possible. In other words, the optimal cut should keep the attribute relations of samples as much as possible since each hyperedge is given by an attribute label. Similar to Zhou's normalized hypergraph [35], we can define an attribute relation loss function with respect to the given hypergraph G and a collection of hyperedge cuts F can be denoted as follows

$$\Omega(F, G) = \frac{1}{2} \sum_{e \in E} \sum_{(u, v) \in e} \frac{w(e)}{\delta(e)} \left\| \frac{F_u}{\sqrt{d(u)}} - \frac{F_v}{\sqrt{d(v)}} \right\|^2, \quad (5)$$

where F_u returns a row vector corresponding to the predictions of attributes for the vertex u . Clearly, the loss will be reduced when the signs of F_u and F_v are identical. Following some deductions, Equation 21 can be reformulated as follows

$$\begin{aligned} \Omega(F, G) &= \frac{1}{2} \sum_{e \in E} \sum_{(u, v) \in e} \frac{w(e)}{\delta(e)} \left\| \frac{F_u}{\sqrt{d(u)}} - \frac{F_v}{\sqrt{d(v)}} \right\|^2 \\ &= \text{Tr}(F^T (I - D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2}) F) \\ &= \text{Tr}(F^T L_H F), \end{aligned} \quad (6)$$

where L_H is the normalized hypergraph Laplacian matrix which is derived from the hypergraph of attributes, and I is an identity matrix. $\text{Tr}(\cdot)$ is the trace of the matrix. The detail deductions of Equation 21 can be found in the supplementary material.

Besides measuring the loss of attribute relation information, we also need to consider the attribute prediction errors of the train data, which can be obtained via calculating the Euclidean distance between attribute predictions F and the attribute label matrix. In order to make zero as the classification boundary, we define a shifted attribute label matrix Y via shifting the attribute labels as $Y = 2H - \mathbf{1}$ where $\mathbf{1}$ is a matrix of the same size as H whose elements are all equal to 1. In Y , if an attribute exists in a sample, its corresponding attribute label is 1, otherwise it is -1. Given this definition, the attribute prediction loss is defined as

$$\Delta(F, Y) = \|F - Y\|^2 \quad (7)$$

Simultaneously minimizing the the previous two losses leads to our model

$$\begin{aligned}\hat{F} &= \arg \min_F \{\Omega(F, G) + \lambda \Delta(F, Y)\} \\ &= \text{Tr}(F^T L_H F) + \lambda \|F - Y\|^2,\end{aligned}\quad (8)$$

where λ is a positive parameter to reconcile these two losses. Now, the optimal hypergraph cut f_i introduces a binary partition to hypergraph that can preserve the information of i -th attribute relation and reduce the prediction error of i -th attribute as much as possible.

From the perspective of graph embedding, the hypergraph cuts are the embedding of the given hypergraph, where the embedding coordinate of sample u is the u -th row of F . Equation 8 actually aligns the hypergraph embedding space (defined by F) with the shifted attribute space. Consequently, we now transform the problem of seeking attribute predictors/cuts to the problem of finding a mapping from the feature space to this aligned embedding space, *i.e.*

$$F = X^T B, \quad (9)$$

where the projection matrix $B = [\beta_1, \dots, \beta_i, \dots, \beta_m]$ is such collection of mappings whose i -th column is a predictor of the i -th attribute, corresponding to the i -th hypergraph cut $f_i = \beta_i^T X$. We then substitute Equation 9 into Equation 8, and introduce L_2 -norm constraint to B to avoid the overfitting. Thus, the Equation 8 is reformulated as the following optimization problem with respect to B

$$\hat{B} = \arg \min_B (\text{Tr}(B^T X L_H X^T B) + \lambda \|X^T B - Y\|^2 + \eta \|B\|^2). \quad (10)$$

where η is a positive regularization parameter. Since L_H is a positive semi-definite matrix, this problem is a typical Regularized Least Square (RLS) problem that can be efficiently solved. We obtain the partial derivative of Equation 10 with respect to B , and equate it to zero, which leads to a closed-form solution for B as follows

$$\begin{aligned}X L_H X^T B + \lambda (X X^T - X Y) + \eta B &= 0 \\ \Rightarrow B &= (X L_H X^T + \lambda X X^T + \eta I)^{-1} (\lambda X Y) \\ \Rightarrow B &= (X L_H X^T + \lambda X X^T + \eta I)^{-1} (\lambda X (2H - \mathbf{1})).\end{aligned}\quad (11)$$

At test time, given a unlabeled sample z_i , its attribute predictions can be achieved by projecting the sample into the subspace spanned by B ,

$$\mathbf{p}_i = \text{sign}(z_i^T B), \quad (12)$$

where $\text{sign}(\cdot)$ returns the sign of each element of a vector and $\mathbf{p}_i = [p_{i1}, \dots, p_{ij}, \dots, p_{im}]$ is a row vector encoded the predicted attributes. Its j -th element $p_{ij} = z_i^T \beta_j$ is the confidence of the existence of the j -th attribute with respect to the sample z_i . We call the subspace spanned by B Attribute Prediction Space (APS), since each basis of this space actually is a predictor of a specific attribute.

3.4. Incorporating Class and Side Information

As a regularized graph learning approach, it is flexible to introduce other meaningful constraints to further enhance attribute learning. In this section, we take the class label as an example to show how to leverage any additional information to enhance our model. The exploitation of class labels can enhance the classification abilities of HAP algorithms, since homogenous samples always share more similarities in attributes.

We adopt two approaches to incorporate the class information. The first approach uses a hypergraph $G_C = (V, E_C)$ to depict the class relation of samples, similar to the way we used a hypergraph to depict the attribute relations in the previous subsection. It is not hard to derive the hypergraph Laplacian L_C from this hypergraph via following the same way as Equation 21. The second approach, following [3, 11], constructs a pairwise graph $G_L = (V, E_L)$ in a supervised way for encoding the class information. We can encode class information using a graph, since unlike the attributes, the classes are disjoint. Two samples are connected with an edge if they belong to the same class (homogenous samples). Similar to the hypergraph model, the heat kernel weighting is adopted as the edge weighting scheme. Finally, the well known Laplacian Eigenmapping model can easily derive the graph Laplacian L_L .

Introducing such class-label graph G_L or hypergraph G_C to the loss function in Equation 21 leads to the new loss function

$$\begin{aligned}\Omega(F, G, G_*) &= \text{Tr}(F^T L_H F + \gamma F^T L_* F) \\ &= \text{Tr}(B^T X (L_H + \gamma L_*) X^T B) \\ &= \text{Tr}(B^T X L_W X^T B),\end{aligned}\quad (13)$$

where G_* denotes either G_L or G_C and L_* is the corresponding Laplacian matrix. $L_W = L_H + \gamma L_*$ is the combination of the original and new Laplacian matrices. According to Equation 10, the new objective function can be reformulated as follows

$$\hat{B} = \arg \min_B (\text{Tr}(B^T X L_W X^T B) + \lambda \|X^T B - Y\|^2 + \eta \|B\|^2) \quad (14)$$

and the solution of B achieved by just replacing L_H with L_W in Equation 11

$$B = (X L_W X^T + \lambda X X^T + \eta I)^{-1} (\lambda X (2H - \mathbf{1})). \quad (15)$$

We call these models Class Specific Hypergraph-based Attribute Predictor (CSHAP). To distinguish the Hypergraph-based CSHAP and Graph-based (Laplacian eigenmapping-based) CSHAP, we respectively denote them by CSHAP_H and CSHAP_G for short. We hypothesize that CSHAP_G is expected to capture the intra-manifold structure between the samples better than CSHAP_H , since CSHAP_H just group the homogenous samples using hyperedges, while CSHAP_G preserves the pair-wise structure.

If more additional information are available, the graph Laplacians that encode these information, can be also added, $L_W = L_H + \gamma_1 L_1 + \dots + \gamma_a L_a$. Such positive regularization parameters $\gamma_i, i \in \{1, \dots, a\}$ can be deduced by multiple kernel learning, since each Laplacian matrix is associated with an affinity matrix (similarity matrix) which can be considered as a kernel matrix.

3.5. Kernelization of HAP

The mapping from the feature space to the shifted attribute space (aligned embedding space) may be not linear. This motivated us to present the kernelization for our method. According to the generalized representer theorem [24], a minimizer of a regularized empirical risk function over a RKHS can be represented as a linear combination of kernels, evaluated on the training set. Inspired by the representer theorem on the attribute classification risk function, we embed kernel representation of the samples (*i.e.* $K_X B$). This transformation could be interpreted that each dimension is the embedding is linear combination of the kernel-evaluations on the training set, which matches the representer theorem.

$$F = K_X B \quad (16)$$

where K_X is an $n \times n$ kernel matrix associated with a kernel function $k(\cdot, \cdot)$, and n is the number of points in the training set. Therefore, the objective functions of the Kernelized HAP (KHAP) and Kernelized CSHAP (KCSHAP) can be denoted as follows

$$\hat{B} = \arg \min_B \{ \text{Tr}(B^T K_X L_A K_X B) + \lambda \|K_X B - Y\|^2 + \eta \|B\|^2 \}$$

where L_A is equal to L_H in the KHAP case and L_W in the KCSHAP case. The solution of Equation 17 is

$$B = (K_X L_A K_X)^T + \lambda (K_X^2 + \eta I)^{-1} (\lambda K_X Y) \quad (17)$$

Then, the attribute predictions can be obtained as follows

$$\mathbf{p}(z_*) = \text{sign}(\mathbf{k}(z_*)^T B) = \text{sign} \left(\sum_{i=1}^N \mathbf{k}(z_*, x_i)^T B \right) \quad (18)$$

where $\mathbf{k}(z_*) = [k(z_*, x_1), \dots, k(z_*, x_n)]$. z_* is the test sample.

3.6. Zero-Shot and N-Shot Learning

Typically there are two ways to annotate the samples using attributes, either to assign the attributes for each sample, or to assign the attributes for each class. Our proposed approach supports both of these two scenarios.

At zero-shot or N-shot time, before we classify samples based on the predicted attributes, we use the sigmoid function to normalize the obtained attribute confidences $\mathbf{s}_i = z_i^T B$ into the range $[0, 1]$.

$$\mathbf{r}_i = \frac{1}{1 + \exp(-\frac{\mathbf{s}_i}{\rho})}, \quad (19)$$

where ρ is a positive scaling parameter and $\mathbf{r}_i = [r_{i1}, \dots, r_{im}]$ is the normalized attribute confidence vector which can be deemed as the probabilities of the existences of attributes.

In the case where only the classes are labeled with attributes, we follow the approach of Direct Attribute Prediction (DAP) [18, 17] where the Bayes' rule is adopted to calculate the posterior of a test class of a given sample based on its attribute probabilities \mathbf{r} . The sample is labeled with the class with the maximum posterior.

With regard to the case where each sample is annotated with attributes, we define the mean of the attribute prototypes in the same class as the attribute template for this class. We denote the template of class j as \mathbf{t}_j . The elements of this template indicate the prior probabilities of the attributes with respect to this class. We classify the samples by directly measuring the Euclidean distance between the attribute existence probabilities of the sample and the attribute template of a class

$$\mathcal{L}(z_i) = \arg \min_j \|\mathbf{r}_i - \mathbf{t}_j\|^2 \quad (20)$$

where $\mathcal{L}(\cdot)$ returns the class label of a sample.

4. Experiments

4.1. Experimental Setups

Datasets: We use three datasets to validate the proposed approach: Animal With Attributes (AWA) [18], Caltech-UCSD Birds (CUB) [30] and Unstructured Social Activity Attribute (USAA) [10]. AWA contains 30,475 images of 50 animal classes. Each class is annotated with 85 attributes. Following [18, 17], we divide the dataset into 40 classes (24,295 images) to be used for training and 10 classes (6180 images) for testing. CUB (2011 version) [30] contains roughly 11,800 images of 200 bird classes. Each class is annotated with 312 binary attributes. We split the dataset following [1] to facilitate direct comparison (150 classes for training and the rest 50 classes for testing). USAA is a video dataset [10] with 69 instance-level attributes for 8 classes of complex social group activity videos from YouTube. Each class has around 100 training and testing videos respectively. We follow [10] for splitting the dataset by randomly dividing the 8 classes into two disjoint sets of four classes each for training and testing (the mean accuracies will be reported).

Features: We adopt the 4096-dimensional deep learning features named DeCAF [6] as the baseline feature for the AWA dataset since these features are already been available online for comparison¹. We extract 4096-dimensional deep learning features called Caffe [16] for representing the images in CUB database. The USAA databases already provided the 14,000-dimensional baseline features² which are constructed from six histogram features, namely RGB

Table 1. Average Attribute Prediction Accuracies (in AUC).

Approaches	Prediction Accuracies (%)		
	AWA	USAA	CUB
HAP	74.0	61.7±1.3	68.5
CSHAP_H	74.0	62.2±0.8	68.7
CSHAP_G	74.3	61.8±1.8	68.5
DAP [17]	72.8/63.0*	—	61.8
IAP [17]	72.1/73.8*	—	—
ALE [1]	65.7	—	60.3

color histograms, SIFT, rgSIFT, PHOG, SURF and local self-similarity histograms [10].

Metrics: We report the classification accuracy (in %) averaged over the classes as the N-shot learning and ZSL accuracy in the AWA and CUB databases. In the USAA database, we follow [10, 9] and report the absolute classification accuracy of data. For attribute prediction accuracies, we report the average Area Under Curve (AUC) for the ROC.

4.2. Attribute Prediction

We report the attribute prediction performance of different approaches in Table 1. Three well known attribute learning approaches, namely Direct Attribute Prediction (DAP) [18, 17], Indirect Attribute Prediction (IAP) [18, 17] and Attribute Label Embedding (ALE) [1] are reported for comparison. The sign ‘*’ indicates the performance of running the code provided by the authors on the DeCafe features we are using, which are also provided by the authors³. From the results, we can find that HAP, CSHAP_H and CSHAP_G outperform all the compared approaches. For example, the accuracy gains of HAP, CSHAP_H and CSHAP_G over DAP are 11%, 11% and 11.3% respectively under the same features and experimental settings. The CSHAP_H performed slightly better than the other two HAP algorithms. This is not surprising, since the contribution of the class label in CSHAP_H and CSHAP_G is expected to be limited for attribute prediction.

4.3. Zero-Shot Learning

The results of seven recent ZSL approaches are complemented for comparison in Table 2. These are Attribute Hierarchical Label Embedding (AHLE) [1], Hierarchies Label Embedding (HLE) [1], Multi-modal Latent Attribute Topic Model (M2LATM) [10], Propagated Semantic Transfer (PST) [22], Zero-Shot Random Forests (ZSRF) [13], Category-Level Attribute approach (CLA) [33] and Decorrelated Attributes (DA) [14]. For [13, 33] we only compared with their results on the attributes provided by the dataset, and not their results using discovered attributes. As before, the sign ‘*’ indicates the performance of running the code

¹<http://www.ist.ac.at/chl/AwA/AwA-features-decaf.tar.bz2>.

²<http://www.eecs.qmul.ac.uk/yf300/USAA/download/>

³We use the code and features available in the AWA webpage. The parameters of the model are well tuned using cross validation to get the best performance.

Table 2. Zero-shot Learning Accuracies.

Approach	Classification Accuracies (%)		
	AWA	USAA	CUB
HAP	45.0	44.1±3.6	17.5
CSHAP_H	45.6	45.3±4.2	17.5
CSHAP_G	45.0	44.6±3.7	17.5
DAP [17]	41.4/42.8*	35.2	10.5
IAP [17]	42.2/35.7*	—	—
ALE [1]	37.4	—	18.0
HLE [1]	39.0	—	12.1
AHLE [1]	43.5	—	17.0
M2LATM [10]	41.3	41.9	—
PST [22]	42.7	36.2	—
ZSRF [13]	43.0	—	—
CLA [33]	42.3	—	—
DA [14]	30.6	—	—

provided by the AWA authors on the same features we are using, which are also provided by the authors.

From the results in Table 2, we can notice that the proposed approaches outperform the compared methods on the AWA and USAA datasets. For example, the accuracy gains of CSHAP_H over DAP and AHLE are 2.8% and 2.1% on AWA dataset. On the USAA dataset, the performance improvements of HAP algorithms over other approaches are more significant. CSHAP_H obtains 10.1% and 3.0% more accuracies in comparison with DAP and M2LATM. Although HAP algorithms have not obtained the best performance in comparison with AHLE in CUB dataset, it still outperforms other approaches. Moreover, the performance gaps between HAP algorithms and AHLE are only 0.5% in this dataset.

In the experiments, CSHAP_H often achieves better results than HAP, since the CSHAP_H attempts to leverage the class labels for clustering the homogenous samples together in the attribution prediction step. Similar phenomenon can be also observed for CSHAP_G while its improvement is less significant. We attribute it to the mechanism of CSHAP_G where it tries to preserve the manifold structure of each class using the given class labels. In ZSL case, the test classes are unseen in the train dataset. Therefore, CSHAP_G may not capture the manifold structures of unseen classes. Another interesting phenomenon is that CSHAP can enhance HAP in the AWA and USAA databases, while cannot improve on the CUB database. This is because the CUB dataset has more attributes which are already enough for distinguish classes while the other two datasets have less attributes so that they benefitted more from the complementary information (The number of attributes of USAA, AWA and CUB dataset are 69, 85 and 312 respectively).

4.4. N-Shot Learning

We extend ZSL into N-Shot Learning (NSL) where a few (N) samples of the test classes are added to the training dataset. The experiments are conducted on the USAA and AWA datasets. Figure 3 shows the trend of the three approaches as N increases. In these experiments, we find that

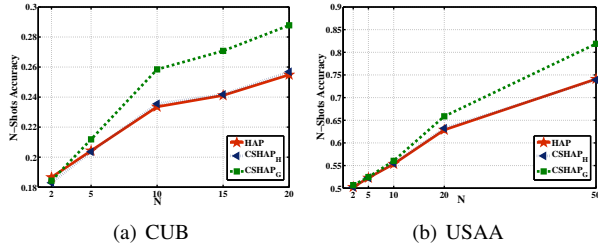


Figure 3. N-Shot Learning accuracies on two datasets.

all three methods can get similar performances when N is small. Interestingly, along increasing N , CSHAP_G significantly outperforms others. This confirms with our hypothesis about the way CSHAP_G captures the intra-class manifold structure while incorporating the class information, in contrast to just grouping the homogenous samples together as CSHAP_H does. In such case, the addition of the samples of the test classes improves the quality of the captured intra-class manifold structures, and therefore improves the performance of CSHAP_G .

Note the performances of HAP methods on USAA dataset are much better than the ones reported in section 4.3, since only half of the testing samples and training samples are used, similar strategy is used in CUB dataset. Compared to the previous experiment N samples per each test class have to be taken out for the NSL training.

4.5. Full Data Categorization

We consider the attributes as the feature representation to tackle the common categorization task. The default splits of data defined in the datasets are employed. Table 3 reports the results. Two classification methods are employed for classification. The first one is the one defined in Equation 20. The second one is the simple Euclidean distance-based Nearest Neighbor Classifier (NNC). The sign ‘†’ indicates the results using NNC. Similar to the in NSL, CSHAP_G achieves the best performances since it better integrates the class information.

Table 3. Full Data Categorization Accuracy (%).

Database	Classification Accuracies (%)			
	HAP	CSHAP_H	CSHAP_G	AHLE
CUB	23.1/21.4†	23.1/21.2†	26.1/24.5†	23.5
USAA	50.1/48.1†	50.1/48.1†	50.9/49.2†	—

4.6. Evaluations of Kernel HAP Algorithms

We also conduct several experiments to test the potentials of the kernel HAP algorithms in Zero-Shot Learning. The Gaussian kernel and Cauchy kernel are applied. Figure 4 shows the ZSL performances of these kernel HAP algorithms in USAA and CUB datasets. In USAA dataset, we can find that Gaussian kernel improves the ZSL accuracies of HAP, CSHAP_H and CSHAP_G from 44.1%, 45.3% and 44.6% to 46.3%, 48.2% and 46.7%. The ZSL accuracies of three Cauchy kernel-based HAP algorithms are 46.1%,

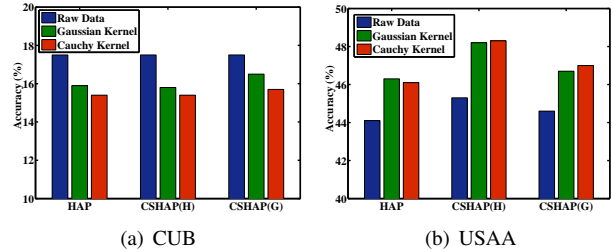


Figure 4. The performances of Kernel HAP algorithms in USAA and CUB datasets.

48.3% and 47.0%. In CUB database, these two kernels actually reduced the ZSL accuracies of HAP algorithms. The accuracies of the kernel HAP algorithms is around 15.5% to 16.5%. We attribute this to the fact that the deep features used in CUB dataset are originally designed to be linear.

4.7. The Parameters and Computational Cost

The experimental results of the choices of the parameters are reported in the supplementary material. Since HAP is graph-based algorithm and involves the matrix inversion, its computational complexity for training is the minimum of $\mathcal{O}(nmd)$ and $\mathcal{O}(d^3)$ and its computational complexity of testing is $\mathcal{O}(d)$. So, it is more time consuming for training but quite efficient for testing. Taking the CUB dataset as an example (5994 samples for training and 5974 samples for testing), the time for training 312 attribute predictors is 23.33 seconds. The time for predicting the attributes of all test samples is 0.14 seconds. The code is written in matlab and the experimental hardware configuration is Quad-Core CPU: 2.5 GHz, RAM: 8G.

5. Conclusion

We presented a novel attribute prediction approach called Hypergraph-based Attribute Predictor (HAP) via deriving a collection of attribute classifiers from the hypergraph embedding, in which the attribute relations are considered as hyperedges and the hypergraph cuts are the attribute predictions. The hypergraph formulation facilitates exploiting the correlations of the attributes as well as jointly learning the attribute predictors. Moreover, the additional information can be flexibly incorporated into HAP via encoding the information in a penalty graph or hypergraph. To generalize the mappings between the feature space and attribute space which are known as the attribute predictors, we also kernelized the model. Extensive experiments on three well known attribute datasets demonstrated the effectiveness of our model for attribute prediction, Zero-Shot Learning, N-Shot Learning and categorization. From the results we can conclude that the CSHAP_G variant is the best to integrate class labels for N-shot learning, however the three proposed variants performs similarly in zero-shot learning task.

Acknowledgement

The work described in this paper was partially supported by the National Natural Science Foundation of China (Grant no. 61173131, 91118005, 11202249), Program for Changjiang Scholars and Innovative Research Team in University (Grant No. IRT1196) and the Fundamental Research Funds for the Central Universities (Grant Nos. CDJZR12098801 and CDJZR11095501). Dan Yang is the corresponding author of this paper.

References

- [1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for attribute-based classification. In *CVPR*, pages 819–826, 2013. 1, 3, 6, 7
- [2] S. Antol, C. L. Zitnick, and D. Parikh. Zero-shot learning via visual abstraction. In *ECCV*, pages 401–416, 2014. 3
- [3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003. 2, 5
- [4] C.-Y. Chen and K. Grauman. Inferring analogous attributes. In *CVPR*, 2014. 3
- [5] G. Chen, J. Zhang, F. Wang, C. Zhang, and Y. Gao. Efficient multi-label classification with hypergraph regularization. In *CVPR*, pages 1658–1665, 2009. 2, 3
- [6] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014. 6
- [7] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *ICCV*, pages 2584–2591, 2013. 3
- [8] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, pages 1778–1785, 2009. 1, 3
- [9] Y. Fu, T. M. Hospedales, T. Xiang, Z. Fu, and S. Gong. Transductive multi-view embedding for zero-shot recognition and annotation. In *ECCV*, pages 584–599, 2014. 3, 7
- [10] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong. Learning multi-modal latent attributes. *TPAMI*, 36(2):303–316, 2014. 1, 3, 6, 7
- [11] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang. Face recognition using laplacianfaces. *TPAMI*, 27(3):328–340, 2005. 5
- [12] Y. Huang, Q. Liu, S. Zhang, and D. N. Metaxas. Image retrieval via probabilistic hypergraph ranking. In *CVPR*, pages 3376–3383, 2010. 4
- [13] D. Jayaraman and K. Grauman. Zero shot recognition with unreliable attributes. In *NIPS*, 2014. 7
- [14] D. Jayaraman, F. Sha, and K. Grauman. Decorrelating semantic visual attributes by resisting the urge to share. In *CVPR*, 2014. 1, 3, 7
- [15] S. Ji, L. Sun, R. Jin, and J. Ye. Multi-label multiple kernel learning. In *NIPS*, pages 777–784, 2009. 2, 3
- [16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 6
- [17] C. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *TPAMI*, 36(3):453, 2014. 1, 3, 6, 7
- [18] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, pages 951–958, 2009. 1, 2, 3, 6, 7
- [19] D. Mahajan, S. Sellamanickam, and V. Nair. A joint learning framework for attribute models and object descriptions. In *ICCV*, pages 1227–1234, 2011. 3
- [20] D. Parikh and K. Grauman. Relative attributes. In *ICCV*, pages 503–510, 2011. 1
- [21] A. Parkash and D. Parikh. Attributes for classifier feedback. In *ECCV*, pages 354–368, 2012. 1
- [22] M. Rohrbach, S. Ebert, and B. Schiele. Transfer learning in a transductive setting. In *NIPS*, pages 46–54, 2013. 7
- [23] M. Rohrbach, M. Stark, and B. Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *CVPR*, pages 1641–1648, 2011. 3
- [24] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *COLT*, 2001. 6
- [25] V. Sharmanska, N. Quadrianto, and C. H. Lampert. Augmented attribute representations. In *ECCV*, pages 242–255, 2012. 3
- [26] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 22(8):888–905, 2000. 2
- [27] B. Siddiquie, R. S. Feris, and L. S. Davis. Image ranking and retrieval based on multi-attribute queries. In *CVPR*, pages 801–808, 2011. 1
- [28] F. Song, X. Tan, and S. Chen. Exploiting relationship between attributes for improved face verification. In *BMVC*, 2011. 1, 3
- [29] L. Sun, S. Ji, and J. Ye. Hypergraph spectral learning for multi-label classification. In *KDD*, pages 668–676, 2008. 2, 3
- [30] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 3, 6
- [31] Y. Wang and G. Mori. A discriminative latent model of object classes and attributes. In *ECCV*, pages 155–168, 2010. 1, 3
- [32] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: a general framework for dimensionality reduction. *TPAMI*, 29(1):40–51, 2007. 2
- [33] F. X. Yu, L. Cao, R. S. Feris, J. R. Smith, and S.-F. Chang. Designing category-level attributes for discriminative visual recognition. In *CVPR*, pages 771–778, 2013. 3, 7
- [34] X. Yu and Y. Aloimonos. Attribute-based transfer learning for object categorization with zero/one training example. In *ECCV*, pages 127–140, 2010. 3
- [35] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*, pages 1601–1608, 2006. 2, 3, 4

The Supplementary Materials

The Derivation Details of Attribute Relation Loss Function

In this section, we will introduce the detailed derivations of Equation (6). Before it, let us review some related notations. The attribute relations are encoded in a given hypergraph $G = (V, E)$ where V is the vertex set and E denotes the hyperedge sets. In this hypergraph, each vertex is corresponding to an instance and each hyperedge is associated with an attribute relation. The degree of a hyperedge $e \in E$, which is denoted as $\delta(e)$, is the number of vertices in e . The (v, e) -th element of vertex-edge incidence matrix $H \in \mathcal{R}^{|V| \times |E|}$ is considered as $h(v, e) = 1$ if $v \in e$ otherwise $h(v, e) = 0$ where $v \in V$. $d(v) = \sum_{v \in e, e \in E} w(e) = \sum_{e \in E} w(e)h(v, e)$ denotes the degree of the vertex v . $w(e)$ is the weight of the hyperedge e . We denote the diagonal matrix forms of $\delta(e)$, $d(v)$ and $w(e)$ as D_e , D_v and W respectively. We obtain the attribute predictions by defining a collection of hypergraph cuts which is denoted as F . F_u returns a row vector of F which is corresponding to the predictions of attributes for the vertex u . L_H is the normalized hypergraph Laplacian matrix which is derived from the hypergraph of attributes, and I is an identity matrix. $\text{Tr}(\cdot)$ is the trace of the matrix.

Now we can present the detailed derivations of Equation (6) as follows:

$$\begin{aligned}
\Omega(F, G) &= \frac{1}{2} \sum_{e \in E} \sum_{(u,v) \in e} \frac{w(e)}{\delta(e)} \left\| \frac{F_u}{\sqrt{d(u)}} - \frac{F_v}{\sqrt{d(v)}} \right\|^2 \\
&= \sum_{e \in E} \sum_{u,v \in V} \frac{w(e)h(u,e)h(v,e)}{\delta(e)} \left(\frac{(F_u)^2}{d(u)} - \frac{F_u F_v^T}{\sqrt{d(u)d(v)}} \right) \\
&= \sum_{e \in E} \sum_{u \in V} \frac{w(e)h(u,e)(F_u)^2}{d(u)} \sum_{v \in V} \frac{h(v,e)}{\delta(e)} - \sum_{e \in E} \sum_{u,v \in V} \frac{F_u w(e)h(u,e)h(v,e)F_v^T}{\delta(e)\sqrt{d(u)d(v)}} \\
&= \sum_{e \in E} (F_u)^2 \sum_{u \in V} \frac{w(e)h(u,e)}{d(u)} - \sum_{e \in E} \sum_{u,v \in V} \frac{F_u w(e)h(u,e)h(v,e)F_v^T}{\delta(e)\sqrt{d(u)d(v)}} \tag{21} \\
&= \sum_{e \in E} (F_u)^2 - \sum_{e \in E} \sum_{u,v \in V} \frac{F_u w(e)h(u,e)h(v,e)F_v^T}{\delta(e)\sqrt{d(u)d(v)}} \\
&= \text{Tr}(F^T F) - \text{Tr}(F^T D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2} F) \\
&= \text{Tr}(F^T (I - D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2}) F) \\
&= \text{Tr}(F^T L_H F),
\end{aligned}$$

The Influences of Parameters

There are several parameters in HAP models. They are μ , λ , η and γ . μ is used for controlling the degree of hyperedge weighting. λ is used for controlling the trade off between the attribute relation loss and the attribute prediction error. η is employed for avoiding the overfitting. γ is adopted for controlling the degree of penalty of the side information loss. The ultimate goals of different attribute learning-based systems are different. Some systems may aim at annotation or retrieval. These systems pay more attention on the improvement of the attribute prediction accuracy. Some other systems may focus on the categorization, *i.e.*, Zero-shot Learning, N-shot Learning and Attribute-based categorization. These systems pay more attention on the exploitation of discriminating power of attributes. In our approach, it is available for us to tune the parameters to decide which evaluation metric we care more. Therefore, we will separately discuss the choices of parameters in these two cases.

The Influences of Parameters to The Attribute Prediction

HAP has three parameters, μ , λ and η , need to tuned. CSHAP algorithms have one more parameter γ need to be tuned. In the parameter selection procedure, we choose one parameter to tune and fix the values of the other parameters. The initial values of μ , λ , η and γ are equal to 0.1. The parameter selection procedure is start from μ to γ . Once the optimal value of a parameter is learned, its corresponding initial value is replaced by that optimal value for more accurately estimating the optimal values of the rest parameters. Figures 5, 6 and 7, respectively reports the attribute prediction accuracies using

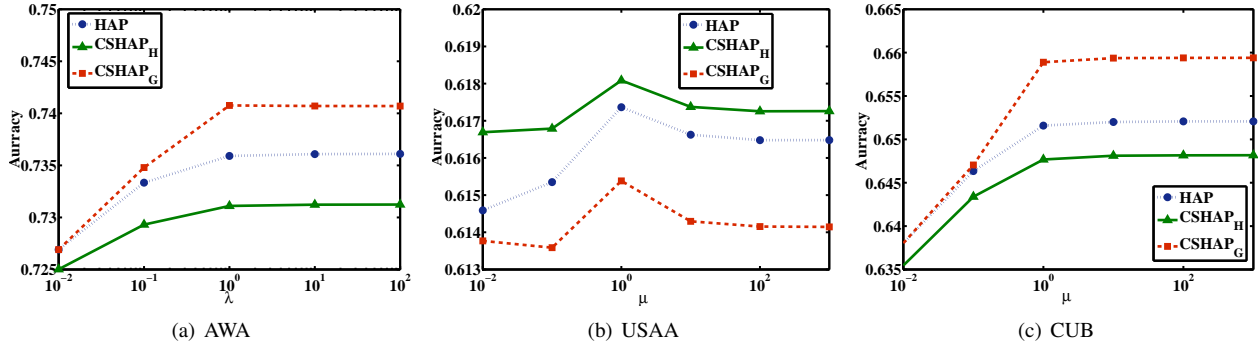


Figure 5. The influences of μ to the attribute prediction accuracies.

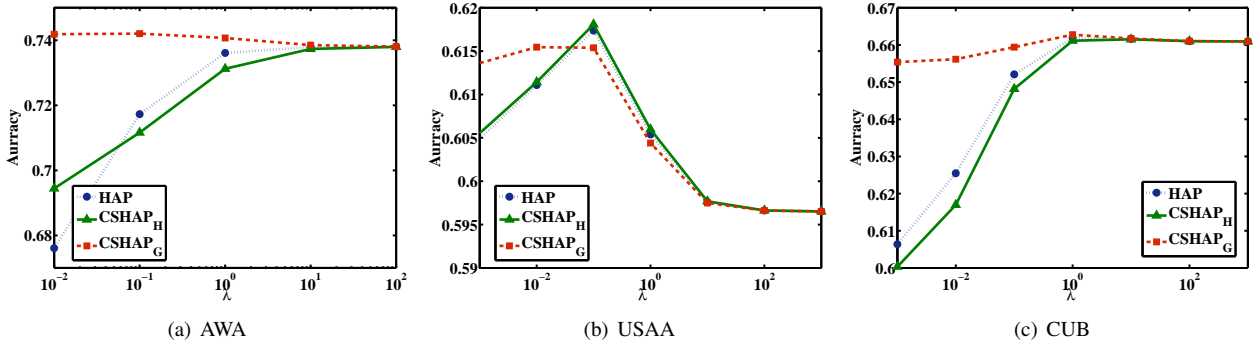


Figure 6. The influences of λ to the attribute prediction accuracies.

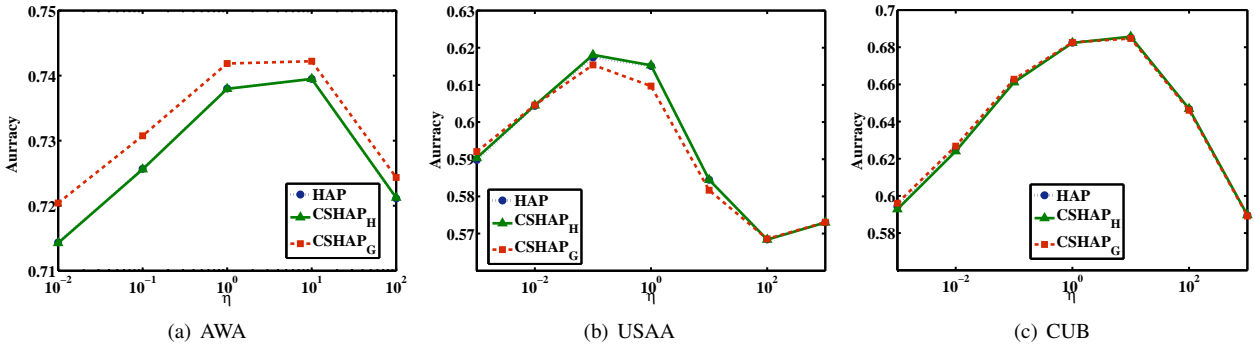


Figure 7. The influences of η to the attribute prediction accuracies.

different μ , λ , η on different databases. From the observations, all HAP algorithms can achieve the best performances on all three databases when $\mu = 1$; The best choices of λ for AWA, USAA and CUB databases are 10, 0.1 and 1 respectively and such numbers of η are 10, 0.1 and 10. Figure 8 plots the relationships between γ and the attribute prediction accuracy on three different databases. We can find that CSHAP_G is more sensitive to γ . It is not hard to conclude from the observations that the optimal values of γ are 1, 0.01 and 10 for AWA, USAA and CUB databases respectively.

The Influences of Parameters to Zero-Shot Learning

In Zero-Shot Learning (ZSL), we need to employ the sigmoid function to normalize the attribute confidences, which are obtained by our models, into range [0,1]. So, there is one additional parameter ρ should be studied in this section. We follow the aforementioned parameter selection manner to select the parameters. The selection procedure is start from μ to ρ where the initial value of ρ is 0.5. Figure 9 shows the ZSL accuracies under different μ . On AWA and CUB database, all three approach can get the best performances when $\mu = 0.1$ while the optimal value of μ on USAA database is 1. Compared with other parameters, HAP algorithms are relatively insensitive to μ when its value is bigger than 1. Figures 10 and 11

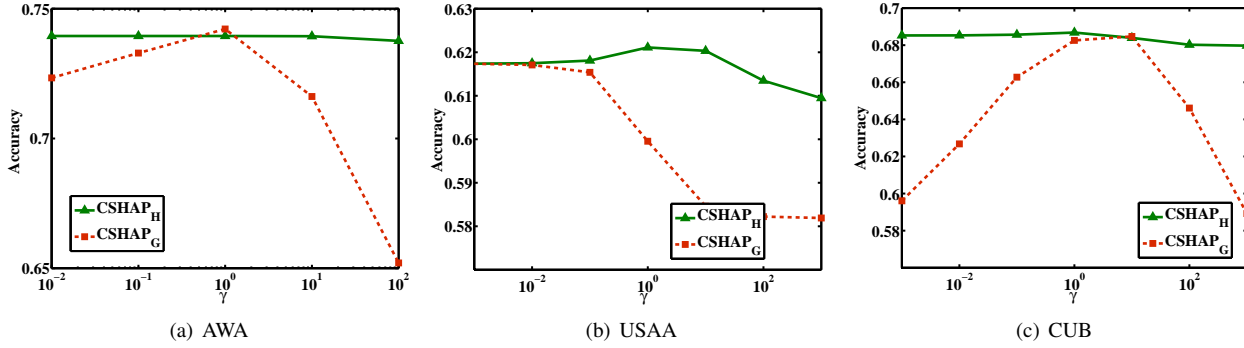


Figure 8. The influences of γ to the attribute prediction accuracies.

demonstrate the impacts of ZSL accuracies from λ and η respectively. The curves of these figures share similar behavior that their peaks are very explicit. From the observations, we can know that the optimal values of λ are 1, 0.01 and 0.1 on AWA, USAA and CUB databases respectively while such numbers of η are 1, 0.1 and 0.1. As same as the phenomenon observed in Figure 8, Figure 12 also shows that CSHAP_G is very sensitive to γ but CSHAP_H is robust to γ . Here, we suggest to set the γ of AWA, USAA and CUB databases to 1, 10^{-3} and 10^{-3} respectively. Figure 13 reports the ZSL performances under different ρ . However, it is really hard to conclude uniform setting for each database. So we choose different ρ for different approaches. More specifically, we suggest to choose the ρ in the range [0.007, 0.02] for CSHAP_H while choose the ρ in the range [0.06, 0.1] for HAP and CSHAP_G on AWA database. On USAA database, CSHAP_G can get good ZSL performances when ρ is in the range [0.005, 0.01] while the good ρ for CSHAP_H and HAP should be above 0.3. The impacts of ρ to the performances of all three algorithms are similar on CUB databases. The observations indicate that the ρ which is larger than 1 can get the good performances for all three HAP algorithms.

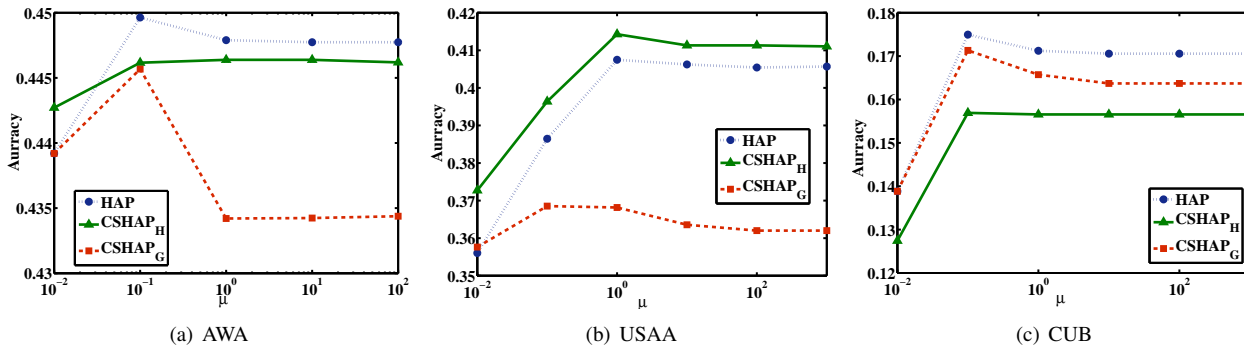


Figure 9. The influences of μ to the ZSL accuracies.

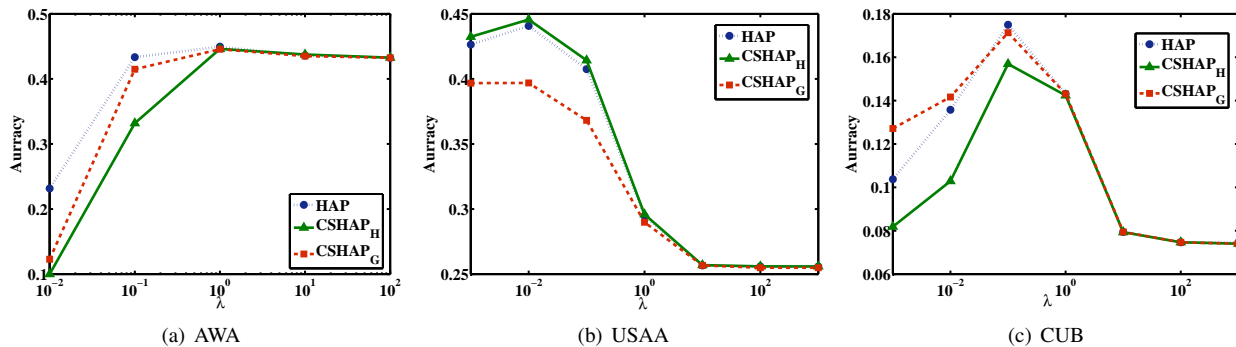


Figure 10. The influences of λ to the ZSL accuracies.

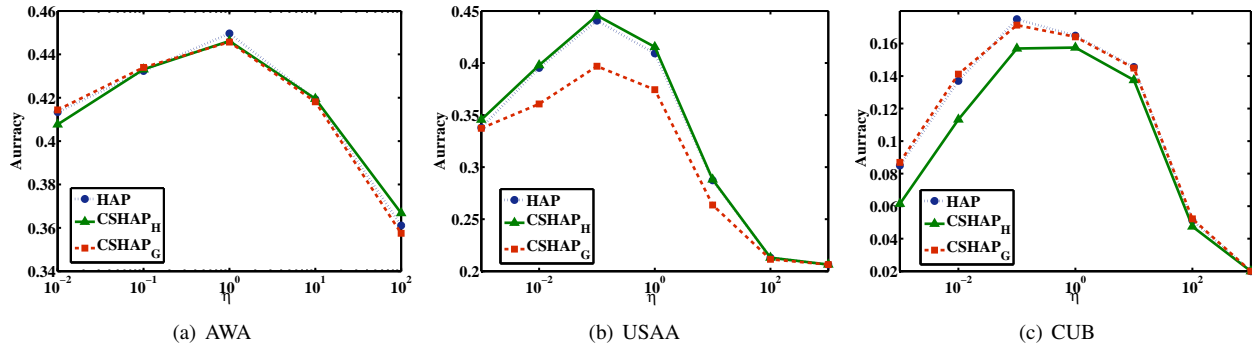


Figure 11. The influences of η to the ZSL accuracies.

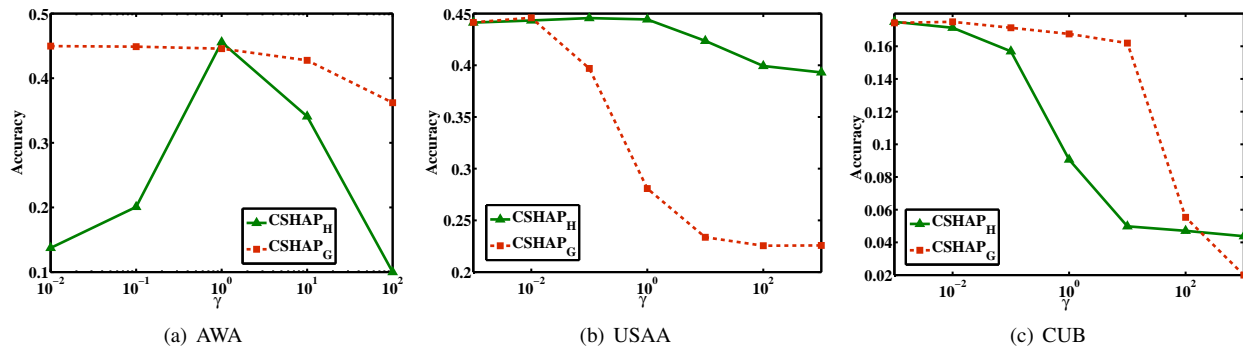


Figure 12. The influences of γ to the ZSL accuracies.

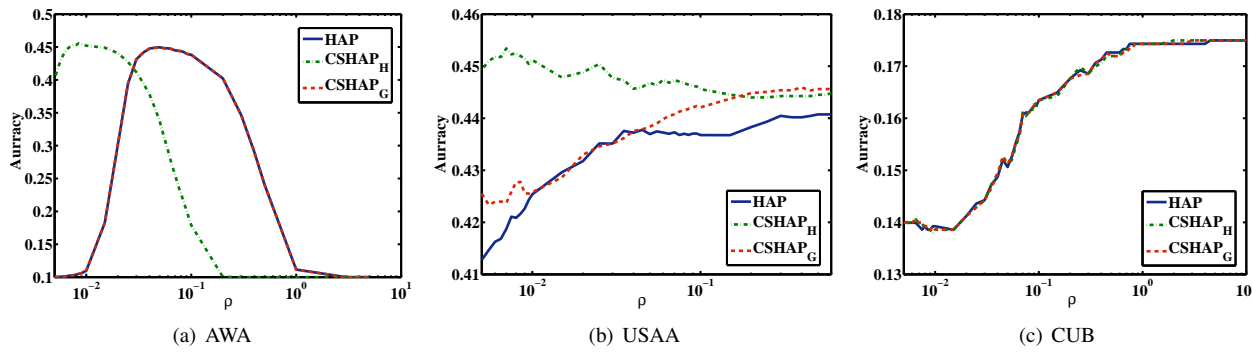


Figure 13. The influences of ρ to the ZSL accuracies.