Zero-Shot Learning with a Partial Set of Observed Attributes

Yaqing Wang, James T. Kwok and Quanming Yao
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
ywangcy, jamesk, qyaoaa@cse.ust.hk

Lionel M. Ni
Department of Computer and Information Science
University of Macau
Taipa, Macau
ni@umac.mo

***Abstract*—Attributes are human-annotated semantic descriptions of label classes. In zero-shot learning (ZSL), they are often used to construct a semantic embedding for knowledge transfer from known classes to new classes. While collecting all attributes for the new classes is criticized as expensive, a subset of these attributes are often easy to acquire. In this paper, we extend ZSL methods to handle this partial set of observed attributes. We first recover the missing attributes through structured matrix completion. We use the low-rank assumption, and leverage properties of the attributes by extracting their rich semantic information from external sources. The resultant optimization problem can be efficiently solved with alternating minimization, in which each of its subproblems has a simple closed-form solution. The predicted attributes can then be used as semantic embeddings in ZSL. Experimental results show that the proposed method outperform existing methods in recovering the structured missing matrix. Moreover, methods using our predicted attributes in ZSL outperforms methods using either the partial set of observed attributes or other semantic embeddings.**

I. Introduction

Zero-shot learning (ZSL) recognizes objects from an unseen class by linking them to the existing classes that one has already learned [1, 2]. It is particularly useful in situations where labeled samples for new classes are expensive or difficult to obtain. Examples include neural activity encoding [3], speech recognition [3], and image classification [4].

Additional semantic class descriptions (which will be called *semantic embeddings*) are introduced in ZSL to cut down the demand of labeled objects. The first and most widely used semantic embedding is class-level attributes, they are human-interpretable properties of the objects' classes, such as shape, color, and size [2, 4, 5]. Empirically, attributes are desirable in terms of transferability [6], human interpretability [2] and classification accuracy [7]. However, the collection of attributes may require expensive expert knowledge.

Recently, the use of unsupervised semantic embeddings is becoming popular due to the lower human effort. Typically, they are learned from independent natural language processing tasks. For example, Akata *et al.* [8] used the ornithology hierarchy in WordNet for class embedding. Frome *et al.* [9] used online text corpus to generate Word2Vec embeddings [10] for class names. Rohrbach *et al.* [11] represented each class with its co-occurrence statistics in searches with all other classes. Qiao *et al.* [12] extracted BOW word embeddings from online articles, and then embedded each class by a predefined vocabulary. While these unsupervised semantic embeddings do not need expert annotations, they still require human effort in correlating the class names with the corresponding semantic embeddings. Moreover, these embeddings are not designed for ZSL, and thus are expected to have inferior performance. An example is shown in Figure 1, which compares the class similarity matrices obtained by attributes and other unsupervised semantic embeddings. As can be seen, attributes preserve most information, while significant information can be lost by unsupervised semantic embeddings.

Attribute based ZSL can be cost-effective if it can deal with a partial set of (inexpensive) attributes collected for the unseen classes. Note that although some attributes can be expensive to collect, many attributes are related to simple properties (e.g., colors and shapes of objects in an image) that can be collected inexpensively, such as by crowdsourcing [14]. Yu *et al.* [15] also attempted to reduce the annotation effort of the attributes. However, they tried to predict attributes at an instance-level with the help of class-level attributes from all classes. In contrast, we only assume a partial set of class-level attributes, thus the annotation effort is much lower.

In this paper, we provide a method which enables ZSL to deal with a partial set of observed attributes. We start by formulating the recovery of missing attributes for the unseen classes as a structured matrix completion problem [16]. Unlike standard matrix completion, entries in the class-attribute matrix are not missing at random locations. By reordering its columns and rows, the missing entries can be arranged as a sub-matrix. On the other hand, a direct application of structured matrix completion ignores the rich semantic information in ZSL. To alleviate the problem, besides using the low-rank structure as in standard/structured matrix completion, we also take into account the semantic similarity among columns and rows. The resultant optimization problem can be solved by alternating minimization, in which each optimization subproblem has a simple closed-form solution. Its convergence is also very fast. Finally, the attributes recovered from the partial observed attributes are used in ZSL as semantic embeddings. Experiments on real-world data sets validate the effectiveness of the proposed method in predicting missing attributes and the efficacy of using the recovered attributes as

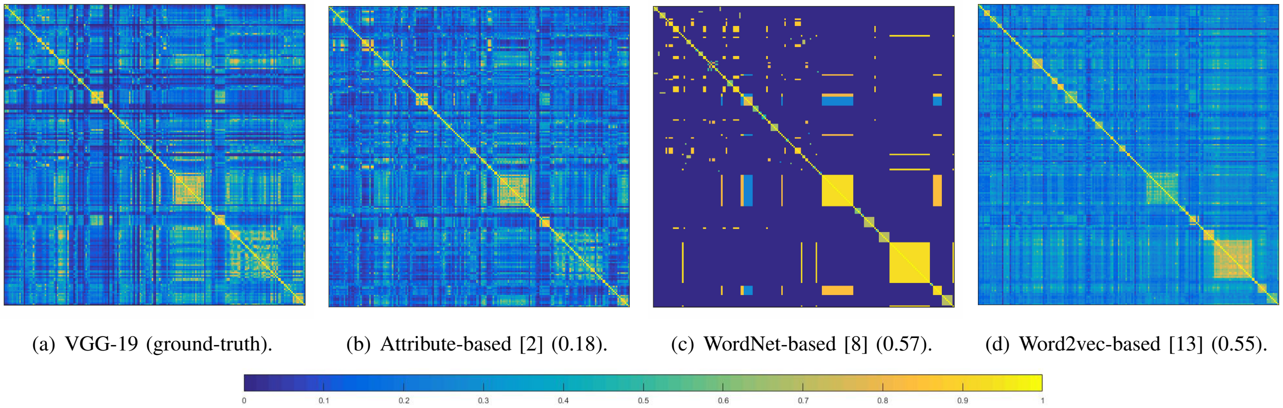

(a) VGG-19 (ground-truth).
(b) Attribute-based [2] (0.18).
(c) WordNet-based [8] (0.57).
(d) Word2vec-based [13] (0.55).

Fig. 1. Class similarity matrices [4] for various semantic embeddings extracted for the *CUB* data set. Figure 1(a) shows the ground-truth similarity matrix S^* based on VGG-19 features. Number in brackets shows the $\|\hat{S} - S^*\|_F / \|S^*\|_F$, where $\hat{S}$ is the similarity matrix estimated using the semantic embeddings.

semantic embeddings in ZSL.

The rest of the paper is organized as follows. Section II briefly introduces some related work. The proposed method is presented in Section III, which is followed by experimental results on both structured matrix completion and ZSL in Section IV. The last section presents some concluding remarks.

II. Related Work

A. Zero-Shot Learning (ZSL)

In a classification problem, ZSL aims at learning classifiers for the new classes without new training samples. We are given a set of training samples $D_s = \{(x_i, y_i)\}_{i=1}^n$, where x_i is the input and $y_i \in \mathbb{R}^{c_1}$ is the corresponding one-hot label vector for c_1 seen classes (i.e., $y_{i,k} = 1$ if sample i belongs to class $k \in \{1, 2, \dots, c_1\}$, and zero otherwise). On testing, we are given c_2 unseen classes, and the goal is to predict class labels of the test set $D_u = \{x_j\}_{j=1}^{N'}$.

There are four main approaches to ZSL. The first one [2] is based on independent classifiers built for the attributes. The attributes are assumed to be independent, which is barely tenable. The second approach [8] projects the attributes to a semantic space, and it then learns a function to match the projected samples with their semantic embeddings. The third approach [6] constructs classifiers for the unseen classes by using weighted averages of the existing classifiers. Finally, the last approach is based on transductive learning [17].

All these methods need semantic embeddings. However, as stated earlier, semantic embeddings may not be complete due to high expenses, and none of the above-mentioned methods can handle a partial set of attributes. We aim at producing a complete semantic embedding, which can then be used in existing ZSL methods. In particular, we will use the ESZSL algorithm proposed in [5] to test our predicted attributes' quality in ZSL. ESZSL is a simple but efficient embedding-based method. It learns a classifier $W \in \mathbb{R}^{d \times a}$ by solving the following optimization problem:

$$\begin{aligned} \min_W \quad & \|Y_s - X_s^\top W S_1\|_F^2 + \beta\lambda\|W\|_F^2 + \beta\|WS_1\|_F^2 \\ & + \lambda\|X_s^\top W\|_F^2, \end{aligned} \qquad (1)$$

where $X_s = [x_1, \dots, x_n] \in \mathbb{R}^{d \times n}$, $Y_s = [y_1, \dots, y_n]^\top \in \mathbb{R}^{n \times c_1}$, and $S_1 \in \mathbb{R}^{a \times c_1}$ contains semantic embeddings for the c_1 seen classes. As the semantic embeddings are based on a attributes, class i's semantic embedding is represented by $s_i \in \mathbb{R}^a$, with $s_{i,j}$ being the association between class i and attribute j. When other semantic embeddings are used, e.g., word embeddings, a can be set to be the length of the embedding. $\|W\|_F^2$ restricts the norm of classifier from arbitrarily large. $\|WS_1\|_F^2$ avoids unbalanced classes by constraining the norm of all projected semantic embeddings in the feature space, and $\|X_s^\top W\|_F^2$ constrains the norm of all projected features on semantic space to avoid over fitting. It can be shown that W has a simple closed-form solution:

$$W = (X_s X_s^\top + \beta I_n)^{-1} X_s Y_s S_1^\top (S_1 S_1^\top + \lambda I_n)^{-1}. \qquad (2)$$

On testing, given the semantic embeddings for c_2 unseen classes, class j's semantic embedding is represented as $S_2^j \in \mathbb{R}^a$. For a test sample x, ESZSL predicts its class as $s^\star = \arg\max_j x^\top W S_2^j$.

B. Structured Matrix Completion

In matrix completion, the underlying matrix[1] is often assumed to be low-rank. Intuitively, this is because there are only a small number of latent factors accounting for the classes' attributes, and, similarly, a few latent factors for the attributes' association strengths in classes. As rank minimization is NP-hard, the rank constraint is often replaced by matrix factorization with a fixed rank or by the nuclear norm regularizer. They perform well both theoretically [18] and empirically [19].

Standard matrix completion assumes that the locations of the missing entries are uniformly random. However, this may not be the case in practice. Specially, Cai *et al.* [16] defines a special setting called structured matrix completion, in which subsets of columns/rows are observed together [16]. This can happen in many real-world scenarios, such as multi-label learning [20], cross-domain collaborative filtering [21], and genomic data integration [16]. An illustration is shown in

[1] Here, it is the attribute-class matrix in ZSL.

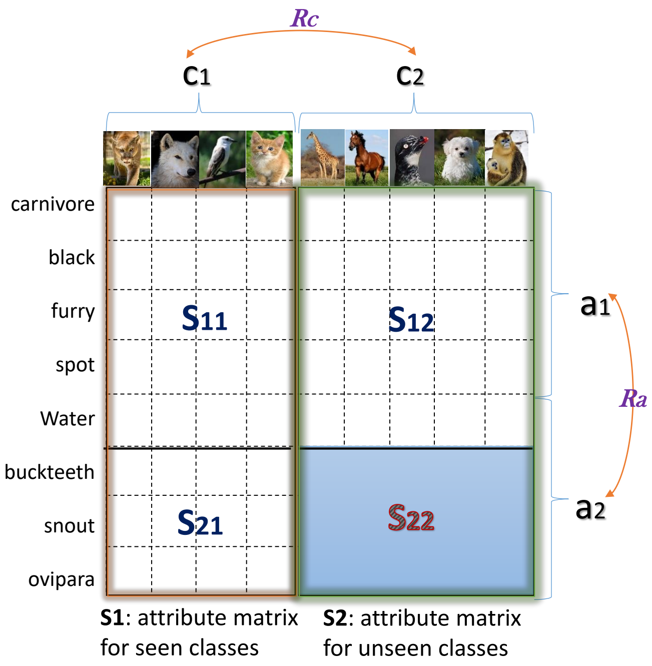

Rc
C1
C2
carnivore
black
furry
spot
Water
buckteeth
snout
ovipara
S11
S12
S21
S22
a1
Ra
a2
S1: attribute matrix for seen classes
S2: attribute matrix for unseen classes

Fig. 2. Illustration of Structured Matrix Completion. We put attributes of both seen and unseen classes into a matrix, then reorder the rows to leave all missing entries in block S_{22} which is our recover target. R_a and R_c measure semantic similarity between attributes and classes extracted from external information sources.

Figure 2. Here, we have reordered the columns and rows so that the missing entries are arranged in a sub-matrix. Let the attribute-class matrix be $S \in \mathbb{R}^{a \times c}$, where $c = c_1 + c_2$ and $a = a_1 + a_2$. $S_1 = [S_{11}; S_{21}]$ contains attribute annotations for the seen classes, while $S_2 = [S_{12}; S_{22}]$ contains those for the unseen classes. The goal is to recover S_{22}, with all its entries missing. In this setting, Cai *et al.* [16] showed that nuclear-norm based methods can fail with high probability.

To alleviate this problem, Cai *et al.* [16] proposed the SMC method. Assume that $\text{rank}([S_{11}, S_{12}]) = \text{rank}([S_{11}^\top, S_{21}^\top]^\top) = \text{rank}(S) = r$. Then, $\text{rank}(S_{11}) = r$. SMC recovers S_{22} as

$$S_{21} S_{11}^\dagger S_{12} = S_{21} V \Sigma^{-1} U^\top S_{12}, \tag{3}$$

where $U \Sigma V^\top$ is the SVD of S_{11}, and † denotes the pseudo-inverse.

Though SMC can have perfect recovery when the underlying matrix is exactly low-rank, it cannot incorporate prior information or other regularizers easily.

III. Proposed Method

In this section, we first recover the values of the missing attributes using structured matrix completion, and then use this as semantic embedding in existing ZSL methods. Besides the low-rank assumption, we improve the structure matrix completion performance by incorporating relationships among the columns (classes) and rows (attributes). Intuitively, similar classes have similar attributes annotations, and similar attributes are likely to be annotated by similar classes. The proposed method will be called Semantic Structured Matrix Factorization (SSMF). When new classes emerge, we only need to re-train SSMF but not the ZSL classifier (i.e., ESZSL).

A. Formulation

Assume that the matrix S to be reconstructed is low-rank (with rank r). Specifically, let S be approximated as $UV^\top$, where $U = [U_1; U_2] \in \mathbb{R}^{a \times r}$, U_1 contains the latent factors for the a_1 seen attributes and U_2 is for the a_2 unseen attributes. Similarly, $V = [V_1; V_2] \in \mathbb{R}^{c \times r}$, where V_1 contains the latent factors for the c_1 seen classes and V_2 is for the c_2 unseen classes. For $i, j \in \{1, 2\}$, sub-matrix S_{ij} in Figure 2 is then approximated as

$$S_{ij} \simeq U_i V_j^\top. \tag{4}$$

The similarity among the seen and unseen attributes is contained in the matrix $R_a \in \mathbb{R}^{a_1 \times a_2}$. Similarly, similarity among the seen and unseen classes is contained in the matrix $R_c \in \mathbb{R}^{c_1 \times c_2}$. Such information can be easily extracted from external sources such as an online corpus. Details will be discussed in Section IV-A2. To recover the sub-matrix $S_{22} \simeq U_2 V_2^\top$ (which contains the unseen attributes of the unseen classes), we assume that it can be obtained by the transform R_c on S_{21}. In other words,

$$U_2 V_2^\top \simeq S_{21} R_c. \tag{5}$$

Besides exploiting the similarity between seen and unseen classes, one can also utilize the similarity between seen and unseen attributes to recover S_{22}. We then approximate S_{22} as

$$U_2 V_2^\top \simeq R_a^\top S_{12}. \tag{6}$$

Note that both (5) and (6) are in the form of linear combination, which has been commonly used in ZSL when leveraging information from the seen classes [6, 7]. Combining all these, SSMF can then be formulated as the following optimization problem:

$$\begin{aligned} & \min_{U_1, U_2, V_1, V_2} \mathcal{L}(U_1, U_2, V_1, V_2) \qquad (7) \\ = \; & \alpha(\|U_1\|_F^2 + \|U_2\|_F^2 + \|V_1\|_F^2 + \|V_2\|_F^2) \\ & + \beta \|U_2 V_2^\top - R_a^\top S_{12}\|_F^2 + \lambda \|U_2 V_2^\top - S_{21} R_c\|_F^2 \\ & + \|U_1 V_1^\top - S_{11}\|_F^2 + \|U_1 V_2^\top - S_{12}\|_F^2 \\ & + \|U_2 V_1^\top - S_{21}\|_F^2. \end{aligned}$$

Here, $\|U_1\|_F^2, \|U_2\|_F^2, \|V_1\|_F^2, \|V_2\|_F^2$ are standard regularizers, the $\|U_2 V_2^\top - R_a^\top S_{12}\|_F^2$ and $\|U_2 V_2^\top - S_{21} R_c\|_F^2$ terms are for enforcing (5) and (6), respectively, while the last three terms are from the low-rank assumption in (4).

B. Optimization by Alternating Minimization

Problem (7) is non-convex. However, by using alternating minimization, all the subproblems become convex and have simple closed-form solutions. Specifically, let $A = U_1^\top U_1$,

$B = U_2^\top U_2$, $C = V_1^\top V_1$, $D = V_2^\top V_2$, and I be the identity matrix, it is easy to show that

$$\begin{aligned}
U_1 &= (\alpha I + (C + D))^{-1}(S_{11}V_1 + S_{12}V_2), && (8)\\
U_2 &= (\alpha I + C + (\beta + \lambda)D)^{-1} \\
&\quad \cdot(S_{21}V_1 + \beta(R_a^\top S_{12})V_2 + \lambda S_{21}R_cV_2), && (9)\\
V_1 &= (\alpha I + (A + B))^{-1}(S_{11}^\top U_1 + S_{21}^\top U_2), && (10)\\
V_2 &= (\alpha I + A + (\beta + \lambda)B)^{-1} \\
&\quad \cdot(S_{12}^\top U_1 + \beta(R_a^\top S_{12})^\top U_2 + \lambda(S_{21}R_c)^\top U_2)(11)
\end{aligned}$$

The whole procedure is shown in Algorithm 1, which is guaranteed to converge to a critical point of (7) [22]. Empirically, it often converges rapidly in fewer than 10 iterations.

Algorithm 1 Semantic Structured Matrix Factorization (SSMF).

```
Input: Observed blocks $S_{11} \in \mathbb{R}^{a_1 \times c_1}$,$S_{21} \in \mathbb{R}^{a_2 \times c_1}$, $S_{12} \in \mathbb{R}^{a_1 \times c_2}$, similarity matrices $R_a \in \mathbb{R}^{a_2 \times a_1}$, $R_c \in \mathbb{R}^{c_1 \times c_2}$, hyper-parameters $\alpha, \beta, \lambda$, rank $r$.
Output: Recovered block $S_{22}^* \in \mathbb{R}^{a_2 \times c_2}$.
Initialize: Initialize $U_1 \in \mathbb{R}^{a_1 \times r}$, $U_2 \in \mathbb{R}^{a_2 \times r}$, $V_1 \in \mathbb{R}^{c_1 \times r}$,$V_2 \in \mathbb{R}^{c_2 \times r}$ with the normal distribution.
while not converged do
    $U_1^* = \arg\min_{U_1} \mathcal{L}(U_1, U_2^*, V_1^*, V_2^*)$ by (8);
    $U_2^* = \arg\min_{U_2} \mathcal{L}(U_1^*, U_2, V_1^*, V_2^*)$ by (9);
    $V_1^* = \arg\min_{V_1} \mathcal{L}(U_1^*, U_2^*, V_1, V_2^*)$ by (10);
    $V_2^* = \arg\min_{V_2} \mathcal{L}(U_1^*, U_2^*, V_1^*, V_2)$ by (11);
end while
return $S_{22}^* = U_2^*(V_2^*)^\top$.
```

C. Computational Complexity

Even though the closed-form solution requires matrix inverse, each sub-problem has low computational cost. Let $\bar{a} = \max(a_1, a_2)$ and $\bar{c} = \max(c_1, c_2)$. As the number of classes is usually larger than the number of attributes, we assume that $\bar{a} \leq \bar{c}$. In each iteration of Algorithm 1, solving each sub-problem consists of two steps: matrix multiplication takes $O(\bar{a}\bar{c}r)$ time and matrix inverse takes $O(r^3)$ time. As the matrix is supposed to be low-rank, r is small compared with $\bar{c}$ and $\bar{a}$. Thus, the total per-iteration time complexity is $O(\bar{a}\bar{c}r)$.

D. Usage with ESZSL

With the recovered sub-matrix $\hat{S}_{22}$, we can use it in any state-of-the-art ZSL method. The main aim here is only to show that the predicted attributes serve as better semantic embeddings than other methods (e.g., word embeddings and WordNet). Thus, we use the aforementioned ESZSL for simplicity and efficiency. Let $\hat{S}_2 = [S_{12}; \hat{S}_{22}] \in \mathbb{R}^{a \times c_2}$. $\hat{S}_2^j \in \mathbb{R}^a$ is the recovered attribute for class j. With classifier W pretrained by ESZSL, we predict the class of a test sample x as $s^\star = \arg\max_j x^\top W \hat{S}_2^j$.

IV. Experiments

In this section, we evaluate the performance of the proposed method in the contexts of structured matrix completion (Section IV-B) and zero-shot learning (Section IV-C).

A. Setup

1) Data Sets: Experiments are performed on three image data sets commonly used in ZSL (Table I): (i) *Animals with Attributes (AwA)* [2], (ii) *CUB-200-2011 Birds (CUB)* [23] and (iii) *SUN attribute (SUN)* [24]. While *AwA* and *CUB* have attributes annotated per class, *SUN* only has annotations per instances. Thus, as in [4], we use the average attribute annotation within each class to construct the class-attribute matrix for *SUN*.

TABLE I
Summary of the ZSL data sets used.

	#classes				
	seen	unseen	total	#attributes	#images
AwA	40	10	50	85	30,475
CUB	150	50	200	312	11,788
SUN	-	-	717	102	14,340

2) Attribute and Class Similarity Matrices: The class similarity matrix $R_c \in \mathbb{R}^{c_1 \times c_2}$ and attribute similarity matrix $R_a \in \mathbb{R}^{a_1 \times a_2}$ are constructed through three steps. We first extract word embeddings for the names of classes and attributes, then compute the semantic similarity between the word embeddings, and finally put the computed values into the similarity matrices.

Specifically, the construction of R_c is as follows (R_a is constructed in a similar manner). First, we extract word embeddings corresponding to the class names from the English Wikipedia[2]. The SkipGram version of Word2Vec [10] (with negative sampling and a window size of 20) is used to obtain 400-dimensional word embedding vectors. Next, we define the similarity measure between word embeddings. Let $k = \log(n) + 1$, and n be the number of words. The semantic similarity between class i's word embedding x_i and class j's word embedding x_j is defined as

$$\text{sim}(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right), \qquad (12)$$

where σ is the mean distance of the vector to its k-th nearest neighbor. Finally, the semantic similarity of an unseen class i with respect to all the seen classes are stored in $R_{c_i} \in \mathbb{R}^{c_1}$ (i.e., $R_{c_{i,j}} = \text{sim}(x_i, x_j)$, where j is a seen class). These vectors are ℓ_1-normalized and become the columns of R_c.

3) Splitting into Seen and Unseen Classes/Attributes: *AwA* and *CUB* have standard splits into seen and unseen classes [2, 8], while *SUN* does not. Although using the standard split allows easier comparison with existing work, the performance of ZSL methods can vary significantly with different class/attribute splits [6]. Hence, we also experiment

[2] We download the entire dump from https://dumps.wikimedia.org/enwiki/20160407/enwiki-20160407-pages-articles.xml.bz2

with random class splitting with varying ratios for the seen classes. Similarly, the attributes are also randomly split.

TABLE II

RRE RESULTS ON *AwA* AND *CUB*, USING STANDARD CLASS SPLIT AND RANDOM ATTRIBUTE SPLIT. THE BEST AND COMPARABLE RESULTS (ACCORDING TO THE PAIRWISE T-TEST WITH 95% CONFIDENCE) ARE IN BOLD. RECALL THAT *SUN* DO NOT HAVE A STANDARD CLASS SPLIT, AND SO ITS RESULTS ARE NOT REPORTED HERE.

		ratio of seen attributes		
		0.25	0.50	0.75
AwA	SMC	0.81±0.09	0.69±0.15	0.65±0.13
	GRALS	0.73±0.02	0.62±0.01	0.63±0.03
	1NN-A	0.97±0.04	0.95±0.04	0.96±0.05
	KNN-A	0.76±0.01	0.75±0.02	0.75±0.03
	1NN-C	0.63±0.01	0.63±0.02	0.65±0.04
	KNN-C	0.64±0.02	0.64±0.02	0.65±0.04
	SSMF	**0.53±0.04**	**0.45±0.02**	**0.45±0.04**
CUB	SMC	0.62±0.11	0.50±0.03	0.44±0.06
	GRALS	0.64±0.01	0.64±0.01	0.64±0.03
	1NN-A	1.22±0.08	1.20±0.14	1.22±0.17
	KNN-A	0.87±0.01	0.87±0.00	0.86±0.01
	1NN-C	0.68±0.01	0.69±0.02	0.69±0.04
	KNN-C	0.62±0.01	0.62±0.01	0.62±0.03
	SSMF	**0.44±0.02**	**0.37±0.02**	**0.33±0.02**

TABLE III

RRE RESULTS ON *AwA*, USING RANDOM CLASS AND ATTRIBUTE SPLITS.

ratio of seen classes		ratio of seen attributes		
		0.25	0.50	0.75
0.25	SMC	0.76±0.12	0.66±0.08	0.67±0.09
	GRALS	0.80±0.06	0.70±0.02	0.69±0.04
	1NN-A	1.08±0.05	1.09±0.05	1.07±0.04
	KNN-A	0.79±0.02	0.79±0.03	0.78±0.02
	1NN-C	0.78±0.04	0.77±0.05	0.76±0.08
	KNN-C	0.68±0.03	0.69±0.03	0.68±0.05
	SSMF	**0.62±0.03**	**0.60±0.05**	**0.57±0.07**
0.50	SMC	0.77±0.13	0.64±0.06	0.63±0.09
	GRALS	0.75±0.03	0.66±0.04	0.64±0.03
	1NN-A	1.08±0.04	1.08±0.04	1.05±0.03
	KNN-A	0.79±0.01	0.79±0.02	0.77±0.02
	1NN-C	0.68±0.02	0.68±0.03	0.65±0.05
	KNN-C	0.65±0.02	0.66±0.03	0.65±0.03
	SSMF	**0.56±0.04**	**0.53±0.03**	**0.50±0.05**
0.75	SMC	0.69±0.16	**0.58±0.06**	0.55±0.08
	GRALS	0.72±0.04	0.63±0.05	0.63±0.06
	1NN-A	1.05±0.03	1.05±0.04	1.04±0.03
	KNN-A	0.79±0.01	0.78±0.02	0.77±0.02
	1NN-C	0.65±0.03	0.65±0.03	0.62±0.07
	KNN-C	0.64±0.04	0.65±0.04	0.64±0.04
	SSMF	**0.51±0.03**	**0.51±0.05**	**0.47±0.03**

TABLE IV

RRE RESULTS ON *CUB*, USING RANDOM CLASS AND ATTRIBUTE SPLITS.

ratio of seen classes		ratio of seen attributes		
		0.25	0.50	0.75
0.25	SMC	0.57±0.04	0.56±0.05	0.52±0.04
	GRALS	0.67±0.01	0.67±0.03	0.67±0.02
	1NN-A	1.19±0.07	1.20±0.07	1.14±0.06
	KNN-A	0.88±0.00	0.88±0.01	0.88±0.01
	1NN-C	0.71±0.02	0.72±0.04	0.75±0.03
	KNN-C	0.64±0.01	0.64±0.04	0.66±0.01
	SSMF	**0.51±0.01**	**0.48±0.03**	**0.45±0.01**
0.50	SMC	0.57±0.06	0.52±0.06	0.46±0.03
	GRALS	0.64±0.01	0.65±0.03	0.66±0.02
	1NN-A	1.19±0.06	1.20±0.07	1.14±0.05
	KNN-A	0.88±0.00	0.88±0.02	0.88±0.01
	1NN-C	0.68±0.02	0.69±0.03	0.71±0.02
	KNN-C	0.63±0.01	0.64±0.03	0.65±0.01
	SSMF	**0.47±0.02**	**0.41±0.02**	**0.38±0.01**
0.75	SMC	0.63±0.06	0.57±0.09	0.46±0.03
	GRALS	0.63±0.01	0.64±0.03	0.65±0.02
	1NN-A	1.19±0.05	1.20±0.07	1.14±0.06
	KNN-A	0.88±0.01	0.88±0.02	0.88±0.01
	1NN-C	0.68±0.01	0.69±0.03	0.71±0.02
	KNN-C	0.62±0.01	0.63±0.03	0.64±0.02
	SSMF	**0.44±0.04**	**0.39±0.02**	**0.34±0.00**

TABLE V

RRE RESULTS ON *SUN*, USING RANDOM CLASS AND ATTRIBUTE SPLITS.

ratio of seen classes		ratio of seen attributes		
		0.25	0.50	0.75
0.25	SMC	0.68±0.14	0.57±0.05	0.49±0.08
	GRALS	0.64±0.07	0.55±0.05	0.49±0.03
	1NN-A	1.14±0.11	1.17±0.12	1.07±0.12
	KNN-A	0.89±0.01	0.89±0.03	0.88±0.02
	1NN-C	0.76±0.03	0.76±0.04	0.75±0.11
	KNN-C	1.23±0.38	1.21±0.36	1.17±0.33
	SSMF	**0.57±0.05**	**0.46±0.02**	**0.41±0.05**
0.50	SMC	0.63±0.08	0.55±0.06	0.48±0.05
	GRALS	0.59±0.06	0.54±0.02	0.48±0.03
	1NN-A	1.13±0.11	1.17±0.12	1.07±0.11
	KNN-A	0.89±0.01	0.89±0.03	0.88±0.02
	1NN-C	0.74±0.03	0.74±0.03	0.73±0.09
	KNN-C	1.91±1.46	1.80±1.30	1.68±1.16
	SSMF	**0.56±0.02**	**0.45±0.03**	**0.40±0.05**
0.75	SMC	0.71±0.06	0.64±0.12	0.54±0.13
	GRALS	0.58±0.02	0.50±0.02	0.47±0.07
	1NN-A	1.14±0.12	1.18±0.13	1.08±0.13
	KNN-A	0.89±0.01	0.89±0.03	0.88±0.02
	1NN-C	0.74±0.02	0.74±0.03	0.73±0.08
	KNN-C	0.84±0.20	0.83±0.20	0.80±0.20
	SSMF	**0.54±0.03**	**0.45±0.02**	**0.39±0.05**

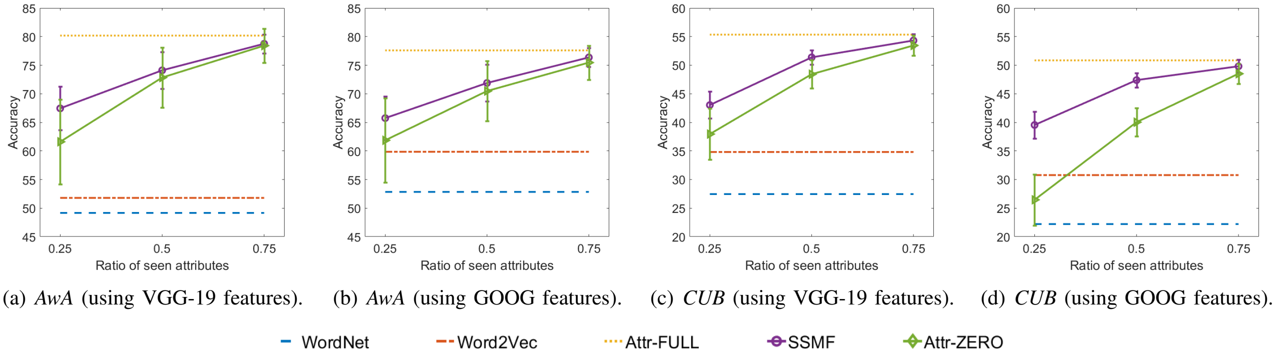
B. Structured Matrix Completion

The proposed SSMF is compared with (i) SMC [16]; and (ii) GRALS [19], which performs matrix completion with graph Laplacian regularization. As further baselines, the following nearest-neighbor-based methods are compared:

1) 1NN-A: Consider class i with an unseen attribute j. 1NN-A first finds the seen attribute a that is closest[3] to j. The value of attribute a on class i is then used to fill in.
2) KNN-A: This is similar to 1NN-A, except that we use a linear combination of all the seen attribute values for class i. The combination weight for each seen attribute is proportional to its similarity with attribute j in (12).
3) 1NN-C: This is similar to 1NN-A, but is based on classes instead of attributes. Specifically, 1NN-C first finds the seen class C that is closest[4] to i. The value of attribute j on class C is then used to fill in.
4) KNN-C: Analogous to 1NN-C, this is similar to KNN-A, but is based on the class embedding vectors instead of the attribute embedding vectors.

[3] Each seen attribute is viewed as a vector in $\mathbb{R}^{c_1}$, and the distance defined in (12) is used for finding the closest attribute.

[4] Each seen class is viewed as a vector in $\mathbb{R}^{a_1}$.



(a) *AwA* (using VGG-19 features). (b) *AwA* (using GOOG features). (c) *CUB* (using VGG-19 features). (d) *CUB* (using GOOG features).

Fig. 3. Classification accuracies on unseen classes for the *AwA* and *CUB* data sets. The ±1 standard deviation are indicated by the vertical bars. Note `Attr-FULL`, `WordNet` and `Word2Vec` are not affected by attributes splits.

`SMC`, `1NN-A`, `KNN-A`, `1NN-C`, and `KNN-C` do not have hyperparameters to tune. For `GRALS` and `SSMF`, hyperparameter tuning is based on the 5-fold cross-validation procedure discussed in [16]. For performance evaluation, we use the relative reconstruction error $RRE = \|\hat{S}_{22} - S_{22}\|_F / \|S_{22}\|_F$ as in [16], where $\hat{S}_{22}$ and S_{22} are the recovered and true submatrices, respectively. To reduce statistical variability, results are averaged over 20 repetitions. Results based on the standard class splits (for *AwA* and *CUB*) are shown in Table II, while those based on random class splits are shown in Tables III (for *AwA*), IV (for *CUB*), and V (for *SUN*). With more seen classes and attributes, performance becomes better for `SMC`, `SSMF` and `GARLS`, but not for the nearest-neighbor-based methods. As can be seen, `SSMF` outperforms the others, showing the benefits of exploiting both the low-rank structure and semantic similarity. Neither considering semantic similarity only (i.e., nearest-neighbor-based methods) nor leveraging low-rank structure only (i.e., `SMC`) is satisfactory. Though `GRALS` also considers both the low-rank structure and pairwise similarity (by the graph Laplacian on all attributes and classes), it assumes that the entries in the class-attribute matrix are missing at random uniformly. Thus, it does not perform as well in our structured matrix completion setting.

C. Zero-Shot Learning

In this section, we use the recovered attribute-class annotation matrix as input semantic embedding to the `ESZSL` algorithm in [5]. We use the popular VGG-19 [25] and GOOG [26] features,[5] which have been commonly used for zero-shot learning on image data sets [6, 7, 12, 13]. The VGG-19 features are 4096-dimensional vectors extracted from the last fully connected hidden layer of the 19-layer VGG net. The GOOG features are 1024-dimensional vectors extracted from the last pooling layer of the GoogLeNet. Both feature sets are pre-trained on the ILSVRC2014 data set.

Besides using the learned attribute-class matrix for semantic embedding, two other semantic embeddings that do not require the presence of class attributes are compared:

(i) `Word2Vec`: As in [7, 13], the word embeddings are learned on Wikipedia.

(ii) `WordNet`: As in [7, 11, 13], the embeddings are extracted from the WordNet hierarchy. Each class i is represented by a vector h_i, such that $h_{i,k} = 1$ if k is an ancestor of i or $i = k$, and 0 otherwise. We only extract WordNet embeddings for *AwA* and *CUB*, as most of *SUN*'s class names are compound words that do not exist in WordNet's vocabulary.

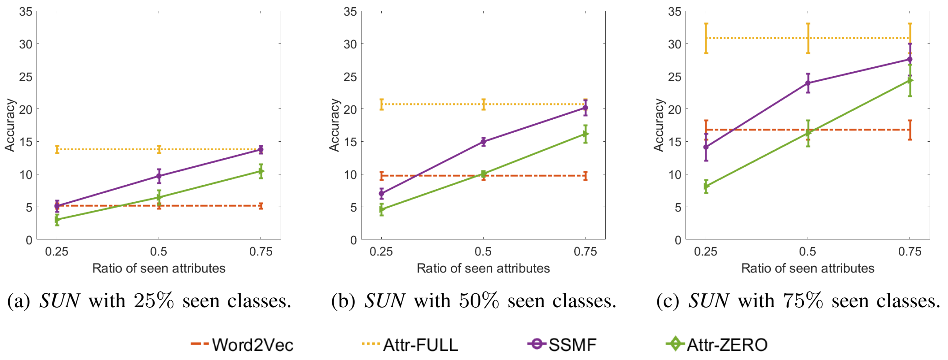
To further illustrate the quality of the learned class-attribute matrix, we also compare with (i) `Attr-FULL`, which contains ground-truth values of the whole attribute-class matrix. This serves as an upper bound on the ZSL performance; (ii) `Attr-ZERO`, which simply sets the unseen attribute values to zero.

As `SSMF` clearly outperforms the other methods in Section IV-B, we only experiment with `SSMF` in this ZSL setting. Hyperparameters are tuned using 5-fold cross-validation as in Section IV-B. To reduce statistical variability, results are averaged over 20 repetitions.

Figure 3 shows the classification accuracies on the unseen classes for the *AwA* and *CUB* data sets (with the standard class split), while Figure 4 shows the classification accuracies for the *SUN* data set (with random class splits). As shown in Figure 3, the VGG-19 features perform slightly better than GOOG. We speculate that this is because VGG-19 is higher-dimensional, and thus can capture more information from the images. The use of class attribute information leads to significantly better ZSL performance over those that do not (`Word2Vec` and `WordNet`). This also agrees with the observations in [6, 7]. Moreover, as expected, accuracy improves with the increasing number of seen classes and seen attributes.

As the similarity values in the attribute-class matrix are in $[0, 1]$, `Attr-ZERO` cannot discriminate missing attributes from dissimilar ones, and thus its performance is inferior to `SSMF`. However, when a high proportion of the attributes become seen (e.g., 75%), its performance on *AwA* can be quite competitive. This is because the ratio of seen classes on this data set is high (80%). Hence, when the number of seen attributes is also large, most of the information has already

[5] The VGG-19 features are from [4], while the GOOG features (only available on the *AwA* and *CUB* data sets) are from [13].



(a) *SUN* with 25% seen classes. (b) *SUN* with 50% seen classes. (c) *SUN* with 75% seen classes.

Fig. 4. Classification accuracies on unseen classes for the *SUN* data set (using the VGG-19 features). With classes split randomly, all semantic embeddings are influenced. Thus all results are measured with standard deviation.

TABLE VI
COMPARISON WITH THE STATE-OF-THE-ART THAT DO NOT USE ATTRIBUTE (ON *AwA* AND *CUB* WITH THE STANDARD SPLIT).

	feature	semantic information	*AwA*	*CUB*
Rohrbach[11]	Low level	WordNet	17.8	-
		Wikipedia	19.7	-
		Yahoo Web	19.5	-
		Yahoo Image	23.6	-
		Flickr Image	22.9	-
ALE[8]	Fisher Vector	WordNet	39.0	12.1
SJE[7]	GOOG	Word2Vec	51.2	28.4
		GloVe	58.8	24.2
		Bag of Words	44.9	22.1
		WordNet	51.2	20.6
LatEm[13]	GOOG	Word2Vec	61.1	31.8
		GloVe	62.9	32.5
		WordNet	57.5	24.2
SynC[6]	GOOG	Word2Vec	57.5	-
Qiao[12]	VGG-19	Bag of Words	66.5	29.0
SSMF+ESZSL	VGG-19	25% attributes	67.5	43.0
		50% attributes	74.1	51.4
		75% attributes	**78.8**	**54.3**
		Word2Vec	51.8	34.8
		WordNet	49.2	27.5
	GOOG	25% attributes	65.7	39.5
		50% attributes	71.9	47.4
		75% attributes	76.4	49.8
		Word2Vec	59.9	30.8
		WordNet	52.8	22.2

been captured and so even discarding the unseen attributes in the unseen classes can still have good performance. However, when the number of seen classes is much smaller (as in Figures 4(a) and 4(b)), the performance difference between `Attr-ZERO` and `SSMF` can be substantial. In practical situations, in order to reduce human annotation effort, many attributes will be missing. Hence, the proposed method can be significantly better.

Finally, Table VI summarizes the state-of-the-art ZSL performance using unsupervised semantic embeddings on the *AwA* and *CUB* data sets (using the standard class split). As can be seen, the use of partial attribute information outperforms all of them.

V. CONCLUSION

In this paper, we extend zero-shot learning with a partial set of observed attributes. Instead of annotating the missing attributes, which can be expensive, we recover them by formulating as a structured matrix completion problem. Semantic similarity between rows and columns is also incorporated. The resultant optimization problem can be efficiently solved by alternating minimization. Experimental results show that the proposed method performs well in predicting the missing attributes. On plugging the predicted attributes into a standard zero-shot learning algorithm, the performance is better than using either the partially observed set of attributes or other semantic embeddings.

ACKNOWLEDGMENT

This research was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region (Grant 614513) and the University of Macau Grant SRG2015-00050-FST.

REFERENCES

[1] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks." in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008, pp. 646–651.

[2] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2009, pp. 951–958.

[3] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Advances in Neural Information Processing Systems*, 2009, pp. 1410–1418.

[4] Z. Zhang and V. Saligrama, "Zero-shot learning via semantic similarity embedding," in *Proceedings of the International Conference on Computer Vision*, 2015, pp. 4166–4174.

[5] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *Proceedings of*

the 32nd International Conference on Machine Learning, 2015, pp. 2152–2161.

[6] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha, "Synthesized classifiers for zero-shot learning," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5327–5336.

[7] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele, "Evaluation of output embeddings for fine-grained image classification," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2927–2936.

[8] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2013, pp. 819–826.

[9] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov *et al.*, "Devise: A deep visual-semantic embedding model," in *Advances in Neural Information Processing Systems*, 2013, pp. 2121–2129.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[11] M. Rohrbach, M. Stark, G. Szarvas, I. Gurevych, and B. Schiele, "What helps where–and why? semantic relatedness for knowledge transfer," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2010, pp. 910–917.

[12] R. Qiao, L. Liu, C. Shen, and A. v. d. Hengel, "Less is more: zero-shot learning from online textual documents with noise suppression," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2249–2257.

[13] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele, "Latent embeddings for zero-shot classification," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2016, pp. 69–77.

[14] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast-but is it good? evaluating non-expert annotations for natural language tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008, pp. 254–263.

[15] F. X. Yu, L. Cao, M. Merler, N. Codella, T. Chen, J. R. Smith, and S.-F. Chang, "Modeling attributes from category-attribute proportions," in *Proceedings of the 22nd International Conference on Multimedia*, 2014, pp. 977–980.

[16] T. Cai, T. T. Cai, and A. Zhang, "Structured matrix completion with applications to genomic data integration," *Journal of the American Statistical Association*, 2015.

[17] Y. Fu, T. M. Hospedales, T. Xiang, Z. Fu, and S. Gong, "Transductive multi-view embedding for zero-shot recognition and annotation," in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 584–599.

[18] E. J. Candès and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.

[19] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Advances in Neural Information Processing Systems*, 2015, pp. 2107–2115.

[20] A. Goldberg, B. Recht, J. Xu, R. Nowak, and X. Zhu, "Transduction with matrix completion: Three birds with one stone," in *Advances in Neural Information Processing Systems*, 2010, pp. 757–765.

[21] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu, "Personalized recommendation via cross-domain triadic factorization," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 595–606.

[22] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: A survey and extensions," *Mathematical Methods of Operations Research*, vol. 66, no. 3, pp. 373–407, 2007.

[23] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.

[24] G. Patterson and J. Hays, "Sun attribute database: Discovering, annotating, and recognizing scene attributes," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2751–2758.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Tech. Rep. arXiv:1409.1556, 2014.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.