A Discriminative Latent Model of Object Classes and Attributes

Yang Wang and Greg Mori

School of Computing Science, Simon Fraser University, Canada
{ywang12,mori}@cs.sfu.ca

Abstract. We present a discriminatively trained model for joint modelling of object class labels (e.g. "person", "dog", "chair", etc.) and their visual attributes (e.g. "has head", "furry", "metal", etc.). We treat attributes of an object as latent variables in our model and capture the correlations among attributes using an undirected graphical model built from training data. The advantage of our model is that it allows us to infer object class labels using the information of both the test image itself and its (latent) attributes. Our model unifies object class prediction and attribute prediction in a principled framework. It is also flexible enough to deal with different performance measurements. Our experimental results provide quantitative evidence that attributes can improve object naming.

1 Introduction

What can we say about an object when presented with an image containing it, such as images shown in Fig. 1? First of all, we can represent the objects by their categories, or names ("bird" "apple" "chair", etc). We can also describe those objects in terms of certain properties or attributes, e.g. "has feather" for (a), "red" for (b), "made of wood" (c) in Fig. 1.

In the computer vision literature, most work in object recognition focuses on the categorization task, also known as *object naming*, e.g. "Does this image window contain a person?" or "Is this an image of a dog (versus cat, chair, table, ...)?". Some recent work [7, 19] proposes to shift the goal of recognition from *naming* to *describing*, i.e. instead of naming the object, try to infer the properties or attributes of objects. Attributes can be parts (e.g. "has ear"), shape (e.g. "is round"), materials (e.g. "made of metal"), color (e.g. "is red"), etc. This attribute-centric approach to object recognition provides many new abilities compared with the traditional naming task, e.g. when faced with an object of a new category, we can still make certain statements (e.g. "red" " furry" "has ear") about it even though we cannot name it.

The concept of attributes can be traced back (at least) to the early work on intrinsic images [1], in which an image is considered as the product of characteristics (in particular, shading and reflectance) of a scene. Conceptually, we can consider shading and reflectance as examples of semantically meaningful

properties (or attributes) of an image. Recently there has been a surge of interest in the computer vision community on learning visual attributes. Ferrari and Zisserman [9] propose a generative model for learning simple color and texture attributes from loose annotations. Farhadi et al. [7] learn a a richer set of attributes including parts, shape, materials, etc. Vaquero et al. [18] introduce a video-based visual surveillance system which allows one to search based on people's fine-grained parts and attributes, e.g. an example could be "show me people with bald head wearing red shirt in the video".

The attribute-centric approach certainly has great scientific value and practical applications. Some attributes (e.g. "red") can indeed be recognized without considering object names, and it is possible for people to infer attributes of objects they have never seen before. But object naming is clearly still important and useful. Consider the image in Fig. 1(a), we as humans can easily recognize this object has the attribute "eye", even though the "eye" corresponds to a very tiny region in the image. Although it is not entirely clear how humans achieve this amazing ability, it is reasonable to believe that we are not running an "eye" detector in our brain in order to infer this attribute. More likely, we infer the object "has eye" in conjunction with recognizing it as a bird (or at least an animal). The issue becomes more obvious when we want to deal with attributes that are less visually apparent. For example, we as humans can recognize the images in Fig. 1(b,c) have the attributes "being edible" and "being able to sit on", respectively. But those attributes are very difficult to describe in terms of visual appearances of the objects – we infer those attributes most likely because we recognize the objects. In addition, the functions of objects cannot always easily be inferred directly from their visual attributes. Consider the two images in Fig. 1(d,e). They are similar in terms of most of their visual attributes – both are "blue", "made of metal", "3D boxy", etc. But they have completely different functions. Those functions can be easily inferred if we recognize Fig. 1(d) as a mailbox and Fig. 1(e) as a trash can.

(a) (b) (c) (d) (e)

Fig. 1. Why cannot we forget about object naming and only work on inferring attributes? Look at the image in (a), it is very hard to infer the attribute "has eye" since "eye" is a very tiny region. But we as humans can recognize it "has eyes" most likely because we recognize it is a bird. Other attributes are difficult to infer from visual information alone, e.g. "edible" for (b) and "sit on" for (c). Meanwhile, objects with similar visual attributes, e.g. (d) and (e), can have different functions, which can be easily inferred if we can name the objects.

Our ultimate goal is to build recognition systems that jointly learn object classes and attributes in a single framework. In this paper, we take the first steps toward this goal by trying to answer the following question: can attributes help object naming? Although conceptually the answer seems to be positive, there have only been limited cases supporting it in special cases. Kumar et al. [11] show that face verification can benefit from inferring attributes corresponding to visual appearances (gender, race, hair color, etc.) and so-called simile attributes (e.g. a mouth that *looks like* Barack Obama). Attributes have also been shown to be useful in solving certain non-traditional recognition tasks, e.g. when training and test classes are disjoint [7, 6, 12]. However, when it comes to the traditional object naming task, there is little evidence showing the benefit of inferring attributes. The work in [7] specifically mentions that attribute based representation does not help significantly in the traditional naming task. This is surprising since object classes and attributes are two closely related concepts. Attributes of an object convey a lot of information about the object category, e.g. an object that "has leg" "has head" "furry" should be more likely to be a dog than a car. Similarly, the name of an object also conveys a lot of information about its possible attributes, e.g. a dog tends to "have leg", and is not likely to "have wing". The work on joint learning of visual attributes and object classes by Wang and Forsyth [19] is the closest to ours. Their work demonstrates that attribute classifiers and object classifiers can improve the performance of each other. However, we would like to point out that the improvement in their work mainly comes from the fact that the training data are *weakly labeled*, i.e. training data are only labeled with object class labels, but not with exact locations of objects in the image. In this case, an object classifier (say "hat") and an attribute classifier (say "red") can help each other by trying to agree on the same location in an image labeled as "red" and "hat". That work does not answer the question of whether attributes can help object naming without this weakly labeled data assumption, e.g. when an image is represented by a feature vector computed from the whole image, rather than a local patch defined by the location of the object.

Our training data consist of images with ground-truth object class labels (e.g. "person", "dog", "chair", etc.) and attribute labels (e.g. "has torso", "metal", "red", etc.). During testing, we are given a new image without the ground-truth attribute labels, and our goal is to predict the object class label of the test image. We introduce a discriminative model for jointly modelling object classes and attributes. Our model is trained in the latent SVM framework [8]. During testing, we treat the attributes as the latent variables and try to infer the class label of a test image.

The contributions of this paper are three-fold. Firstly and most importantly, we propose a model clearly showing that attributes can help object naming. Our model is also very flexible – it can be easily modified to improve upon many different performance measurements. Secondly, most previous work (e.g. [7, 19]) assumes attributes are independent of each other. This is clearly not true. An object that "has ear" is more likely to "has head", and less likely to be "made

of metal". An important question is how to model the correlations among attributes. We introduce the *attribute relation graph*, an undirected graphical model built from training data, to capture these correlations. Thirdly, our model can be generally applied to address a whole class of problems which we call *recognition with auxiliary labels*. Those problems are characterized as classification tasks with certain additional information provided on training data. Many problems in computer vision can be addressed in this framework. For example, in pedestrian detection, auxiliary labels can be the body part locations. In web image classification, auxiliary labels can be the textual information surrounding an image. There has been work that tries to build recognition systems that make use of those auxiliary labels, e.g. [17] for pedestrian detection and [20] for object image classification. However, those work typically use a simple two-stage classification process by first building a system to predict the auxiliary labels, then learning a second system taking into account those auxiliary labels. Conceptually, it is much more appealing to integrate these two stages in a unified framework and learn them jointly, which is exactly what we do in this paper.

2 Model Formulation

A training example is represented as a tuple $(\mathbf{x}, \mathbf{h}, y)$. Here $\mathbf{x}$ is the image itself. The object class label of the image is represented by $y \in \mathcal{Y}$, where $\mathcal{Y}$ is a finite label alphabet. The attributes of the image $\mathbf{x}$ are denoted by a K-dimensional vector $\mathbf{h} = (h_1, h_2, ..., h_K)$, where $h_k \in \mathcal{H}_k$ $(k = 1, 2, ..., K)$ indicates the k-th attribute of the image. We use $\mathcal{H}_k$ to indicate the set of possible configurations of the k-th attribute. For example, if the k-th attribute is "2D boxy", we will have $\mathcal{H}_k = \{0, 1\}$, where $h_k = 1$ means this object is "2D boxy", while $h_k = 0$ means it is not. If the k-the attribute is "leg", $h_k = 1$ means this object "has leg", while $h_k = 0$ means it does not. The datasets used in this paper only contain binary-valued attributes, i.e. $\mathcal{H}_k = \{0, 1\}$ $(k = 1, 2, ..., K)$. For ease of presentation, we will simply write $\mathcal{H}$ instead of $\mathcal{H}_k$ from now on when there are no confusions. But we emphasize that our proposed method is not limited to binary-valued attributes and can be generalized to multi-valued or continuous-valued attributes.

We assume there are certain dependencies between some attribute pairs (h_j, h_k). For example, h_j and h_k might correspond to "head" and "ear", respectively. Then their values are highly correlated, since an object that "have head" tends to "have ear" as well. We use an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which we call the *attribute relation graph*, to represent these dependency relations between attribute pairs. A vertex $j \in \mathcal{V}$ corresponds to the j-th attribute, and an edge $(j, k) \in \mathcal{E}$ indicates that attributes h_j and h_k have a dependency. We only consider dependencies of pairs of attributes in this paper, but it is also possible to define higher-order dependencies involving more than two attributes. We will describe how to obtain the graph $\mathcal{G}$ from training data in Sec. 5.

Given a set of N training examples $\{(\mathbf{x}^{(n)}, \mathbf{h}^{(n)}, y^{(n)})\}_{n=1}^{N}$, our goal is to learn a model that can be used to assign the class label y to an unseen test image

$\mathbf{x}$. Note that during testing, we do not know the ground-truth attributes $\mathbf{h}$ of the test image $\mathbf{x}$. Otherwise the problem will become a standard classification problem and can be solved using any off-the-shelf classification method.

We are interested in learning a discriminative function $f_{\mathbf{w}} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ over an $\mathbf{x}$ image and its class label y, where $\mathbf{w}$ are the parameters of this function. During testing, we can use $f_{\mathbf{w}}$ to predict the class label y^* of the input $\mathbf{x}$ as $y^* = \arg\max_{y \in \mathcal{Y}} f_{\mathbf{w}}(\mathbf{x}, y)$. Inspired by the latent SVM [8] (also called the max-margin hidden conditional random field [21]), we assume $f_{\mathbf{w}}(\mathbf{x}, y)$ takes the following form: $f_{\mathbf{w}}(\mathbf{x}, y) = \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$, where $\Phi(\mathbf{x}, \mathbf{h}, y)$ is a feature vector depending on the image $\mathbf{x}$, its attributes $\mathbf{h}$ and its class label y. We define $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$ as follows:

$$\begin{aligned} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y) = \mathbf{w}_y^\top \phi(\mathbf{x}) + \sum_{j \in \mathcal{V}} \mathbf{w}_{h_j}^\top \varphi(\mathbf{x}) + \sum_{j \in \mathcal{V}} \mathbf{w}_{y,h_j}^\top \omega(\mathbf{x}) \\ + \sum_{(j,k) \in \mathcal{E}} \mathbf{w}_{j,k}^\top \psi(h_j, h_k) + \sum_{j \in \mathcal{V}} v_{y,h_j} \end{aligned} \tag{1}$$

The model parameters $\mathbf{w}$ are simply the concatenation of the parameters in all the factors, i.e. $\mathbf{w} = \{\mathbf{w}_{h_j}; \mathbf{w}_{y,h_j}; \mathbf{w}_{j,k}; \mathbf{w}_y; v_{y,h_j}\}_{y \in \mathcal{Y}, h_j \in \mathcal{H}, j \in \mathcal{V}, (j,k) \in \mathcal{E}}$. The details of the potential functions in Eq. (1) are described in the following.

Object class model $\mathbf{w}_y^\top \phi(\mathbf{x})$: This potential function represents a standard linear model for object recognition without considering attributes. Here $\phi(\mathbf{x}) \in \mathbb{R}^d$ represents the feature vector extracted from the image $\mathbf{x}$, the parameter $\mathbf{w}_y$ represents a template for object class y. If we ignore other potential functions in Eq. (1) and only consider the object class model, the parameters $\{\mathbf{w}_y\}_{y \in \mathcal{Y}}$ can be obtained by training a standard multi-class linear SVM.

In our current implementation, rather than keeping $\phi(\mathbf{x})$ as a high dimensional vector of image features, we simply represent $\phi(\mathbf{x})$ as the score of a pre-trained multi-class linear SVM. In other words, we first ignore the attributes in the training data and train a multi-class SVM from $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$. Then we use $\phi(\mathbf{x}; y)$ to denote the SVM score of assigning $\mathbf{x}$ to class y. Note that we explicitly put y in the notation of $\phi(\cdot)$ to emphasize that the value depends on y. We use $\phi(\mathbf{x}; y)$ as the feature vector. In this case, $\mathbf{w}_y$ is a scalar used to re-weight the SVM score corresponding to class y. This significantly speeds up the learning algorithm with our model. Similar tricks have been used in [3, 22].

Global attribute model $\mathbf{w}_{h_j}^\top \varphi(\mathbf{x})$: This potential function is a standard linear model trained to predict the label (1 or 0) of the j-th attribute for the image $\mathbf{x}$, without considering its object class or other attributes. The parameter $\mathbf{w}_{h_j}$ is a template for predicting the j-th attribute to have label h_j. If we only consider this potential function, the parameters $\{\mathbf{w}_{h_j}\}_{h_j \in \mathcal{H}}$ can be obtained via a standard binary linear SVM trained from $\{(\mathbf{x}^{(n)}, h_j^{(n)})\}_{n=1}^N$. Similarly, instead of keeping $\varphi(\mathbf{x})$ as a high dimensional vector of image features, we simply represent it using a scalar $\varphi(\mathbf{x}; j, h_j)$, which is the score of predicting the j-th attribute of $\mathbf{x}$ to be h_j by the pre-trained binary SVM.

Class-specific attribute model $\mathbf{w}_{y,h_j}^\top \omega(\mathbf{x})$: In addition to the global attribute model, we also define a class-specific attribute model for each object class

$y \in \mathcal{Y}$. Here $\mathbf{w}_{y,h_j}$ is a template for the j-th attribute to take the label h_j *if the object class is* y. If we only consider this potential function, $\mathbf{w}_{y,h_j}$ ($h_j \in \{0,1\}$) for a fixed y can be obtained by learning a binary linear SVM from training examples of object class y. Similarly, we represent $\omega(\mathbf{x})$ as a scalar $\omega(\mathbf{x}; y, j, h_j)$, which is the score of predicting the j-th attribute to be h_j by an SVM pre-trained from examples of class y.

The motivations for this potential function are two-fold. First, as pointed out by Farhadi et al. [7], learning an attribute classifier across object categories is difficult. For example, it is difficult to learn a classifier to predict the attribute "wheel" on a dataset containing cars, buses, trains. The learning algorithm might end up learning "metallic" since most of the examples of "wheels" are surrounded by "metallic" surfaces. Farhadi et al. [7] propose to address this issue by learning a "wheel" classifier *within a category* and do feature selection. More specifically, they learn a "wheel" classifier from a single object category (e.g. cars). The "wheel" classifier learned in this fashion is less likely to be confused by "metallic", since both positive and negative examples (i.e. cars with or without "wheel") in this case have "metallic" attributes. Then they can select features that are useful for differentiating "wheel" from "non-wheel" based on the classifier trained within the car category. The disadvantage of the feature selection approach in [7] is that it is disconnected from the model learning and requires careful manual tuning. Our class-specific attribute model achieves a similar goal to the feature selection strategy in [7], but in a more principled manner since the feature selection is implicitly achieved via the model parameters returned by the learning algorithm.

Second, the same attribute might appear differently across multiple classes. For example, consider the attribute "leg". Many object classes (e.g. people, cats) can "have leg" . But the "legs" of people and "legs" of cats can be very different in terms of their visual appearances. If we learn a "leg" attribute classifier by considering examples from both people and cats categories, the learning algorithm might have a hard time figuring out what "legs" look like due to the appearance variations. By separately learning a "leg" classifier for each object category, the learning becomes easier since the positive examples of "legs" within each category are similar to each other. This allows the learning algorithm to use certain visual properties (e.g. furry-like) to learn the "leg" attribute for cats, while use other visual properties (e.g. clothing-like) to learn the "leg" attribute for people.

One might think that the class-specific attribute model eliminates the need for the global attribute model. If this is the case, the learning algorithm will set $\mathbf{w}_{h_j}$ to be zero. However, in our experiment, both $\mathbf{w}_{h_j}$ and $\mathbf{w}_{y,h_j}$ have non-zero entries, indicating these two models are complementary rather than redundant.

Attribute-attribute interaction $\mathbf{w}_{j,k}^{\top}\psi(h_j, h_k)$: This potential function represents the dependencies between the j-th and the k-th attributes. Here $\psi(h_j, h_k)$ is a sparse binary vector of length $|\mathcal{H}| \times |\mathcal{H}|$ (i.e. 4 in our case, since $|\mathcal{H}| = 2$) with a 1 in one of its entries, indicating which of the four possible configurations $\{(1,1),(1,0),(0,1),(0,0)\}$ is taken by (h_j, h_k), e.g. $\psi(1,0) = [0,1,0,0]^{\top}$.

The parameter $\mathbf{w}_{j,k}$ is a 4-dimensional vector representing the weights of all those configurations. For example, if the j-th and the k-th attributes correspond to "ear" and "eye". The entries of $\mathbf{w}_{j,k}$ that correspond to (1,1) and (0,0) will probably tend to have large values, since "ear" and "eye" tend to appear together in any object.

Object-attribute interaction v_{y,h_j}: This is a scalar indicating how likely the object class being y and the j-th attribute being h_j. For example, let y correspond to the object class "people" and the j-th attribute is "torso", then $v_{y,1}$ will probably have a large value since most "people" have "torso" (i.e. $h_j = 1$).

3 Learning Objective

If the ground-truth attribute labels are available during both training and testing, we can simply consider them as part of the input data and solve a standard classification problem. But things become tricky when we want to take into account the attribute information on the training data, but do not want to "overly trust" this information since we will not have it during testing. In this section, we introduce two possible choices of learning approaches and discuss why we choose a particular one of them.

Recall that an image-label pair $(\mathbf{x}, y)$ is scored by the function of the form $f_{\mathbf{w}}(\mathbf{x}, y) = \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$. Given the model parameter $\mathbf{w}$, we need to solve the following inference problem during testing:

$$\mathbf{h}^* = \arg\max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y) \quad \forall y \in \mathcal{Y} \tag{2}$$

In our current implementation, we assume $\mathbf{h}$ forms a tree-structured model. In this case, the inference problem in Eq. (2) can be efficiently solved via dynamic programming or linear program relaxation [16, 21].

Learning with latent attributes: Given a set of N training examples $S = \{(\mathbf{x}^{(n)}, \mathbf{h}^{(n)}, y^{(n)})\}_{n=1}^N$, we would like to train the model parameter $\mathbf{w}$ that tends to produce the correct label for an image $\mathbf{x}$. If the attributes $\mathbf{h}$ are unobserved during training and are treated as latent variables, a natural way to learning the model is to use the latent SVM [8, 21] formulation as follows:

$$\begin{aligned} &\min_{\mathbf{w},\xi} \beta||\mathbf{w}||^2 + \sum_{n=1}^{N} \xi^{(n)} \\ &\text{s.t.} \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}^{(n)}, \mathbf{h}, y^{(n)}) - \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}^{(n)}, \mathbf{h}, y) \geq \Delta(y, y^{(n)}) - \xi^{(n)}, \forall n, \forall y \end{aligned} \tag{3}$$

where β is the trade-off parameter controlling the amount of regularization, and $\xi^{(n)}$ is the slack variable for the n-th training example to handle the case of soft margin, $\Delta(y, y^{(n)})$ is a loss function indicating the cost of misclassifying $y^{(n)}$ as y. In standard multi-class classification problems, we typically use the 0-1 loss $\Delta_{0/1}$ defined as:

$$\Delta_{0/1}(y, y^{(n)}) = \begin{cases} 1 \text{ if } y \neq y^{(n)} \\ 0 \text{ otherwise} \end{cases} \tag{4}$$

Learning with observed attributes: Now since we do observe the ground-truth attributes $\mathbf{h}^{(n)}$ on the training data, one might think a better choice would be to fix those values for $y^{(n)}$ rather than maximizing over them, as follows:

$$\min_{\mathbf{w},\xi} \beta||\mathbf{w}||^2 + \sum_{n=1}^{N} \xi^{(n)}$$
$$\text{s.t.} \mathbf{w}^\top \Phi(\mathbf{x}^{(n)}, \mathbf{h}^{(n)}, y^{(n)}) - \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}^{(n)}, \mathbf{h}, y) \geq \Delta(y, y^{(n)}) - \xi^{(n)}, \forall n, \forall y \quad (5)$$

The two formulations Eq. (3) and Eq. (5) are related as follows. First, let us define $\widehat{\mathbf{h}}^{(n)}$ as $\widehat{\mathbf{h}}^{(n)} = \arg\max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}^{(n)}, \mathbf{h}, y^{(n)})$. Then it is easy to show that Eq. (3) is a non-convex optimization, while Eq. (5) is convex. In particular, Eq. (5) provides a convex upper-bound on Eq. (3). The bound is tight if $\widehat{\mathbf{h}}^{(n)}$ and $\mathbf{h}^{(n)}$ are identical for $\forall n$.

Discussion: Even though Eq. (5) provides a surrogate of optimizing Eq. (3) as its upper bound, our initial attempt of using the formulation in Eq. (5) suggests that it does not work as well as that in Eq. (3). We believe the reason is the optimization problem in Eq. (5) assumes that we will have access to the ground-truth attributes *during testing*. So the objective being optimized in Eq. (5) does not truthfully mimic the situation at run-time. This will not be an issue if the bound provided by Eq. (5) is tight. Unfortunately, having a tight bound means we need to set the parameters $\mathbf{w}$ to almost perfectly predict $\mathbf{h}$ given $(\mathbf{x}^{(n)}, y^{(n)})$, which is obviously difficult.

This might be surprising given the fact that the formulation in Eq. (3) seems to ignore some information (i.e. ground-truth attribute labels) during training. At first glance, this argument seems to be reasonable, since Eq. (3) does not require the ground-truth attributes $\mathbf{h}^{(n)}$ at all. But we would like to argue that this is not the case. The information provided by the ground-truth attributes on training data has been implicitly injected into the feature vectors $\varphi(\mathbf{x})$ and $\omega(\mathbf{x})$ defined in the global attribute model and class-specific attribute model (see the descriptions in Sec. 2), since $\varphi(\mathbf{x})$ and $\omega(\mathbf{x})$ are vectors of SVM scores. Those scores are obtained from SVM classifiers trained using the ground-truth attribute labels. So implicitly, Eq. (3) already makes use of the information of the ground-truth attributes from the training data. In addition, Eq. (3) effectively models the uncertainty caused by the fact that we do not know the attributes during testing and it is difficult to correctly predict them. So in summary, we choose the **learning with latent attributes** (i.e. non-convex version) formulated in Eq. (3) as our learning objective. But we would like to emphasize that the convex version in Eq. (5) is also a reasonable learning objective. In fact, it has been successfully applied in other applications [3]. We leave the further theoretical and empirical studies of these two different formulations as future work.

4 Non-Convex Cutting Plane Training

The optimization problem in Eq. (3) can be solved in many different ways. In our implementation, we adopt a non-convex cutting plane method proposed

in [4] due to its ease of use. First, it is easy to shown that Eq. (3) is equivalent to $\min_{\mathbf{w}} L(\mathbf{w}) = \beta||\mathbf{w}||^2 + \sum_{n=1}^{N} R^n(\mathbf{w})$ where $R^n(\mathbf{w})$ is a hinge loss function defined as:

$$R^n(\mathbf{w}) = \max_{y} \left(\Delta(y, y^{(n)}) + \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}^{(n)}, \mathbf{h}, y) \right) - \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}^{(n)}, \mathbf{h}, y^{(n)}) \quad (6)$$

The non-convex cutting plane method in [4] aims to iteratively building an increasingly accurate piecewise quadratic approximation of $L(\mathbf{w})$ based on its sub-gradient $\partial_{\mathbf{w}} L(\mathbf{w})$. The key issue here is how to compute the sub-gradient $\partial_{\mathbf{w}} L(\mathbf{w})$. Let us define:

$$\begin{aligned} \mathbf{h}_y^{(n)} &= \arg\max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}^{(n)}, \mathbf{h}, y) \quad \forall n,\ \forall y \in \mathcal{Y} \\ y^{*(n)} &= \arg\max_{y} \left(\Delta(y, y^{(n)}) + \mathbf{w}^\top \Phi(\mathbf{x}^{(n)}, \mathbf{h}_y^{(n)}, y) \right) \end{aligned} \quad (7)$$

As mentioned in Sec. 2, the inference problem in Eq. (7) can be efficiently solved if the attribute relation graph forms a tree. It is easy to show a sub-gradient $\partial_{\mathbf{w}} L(\mathbf{w})$ can be calculated as follows:

$$\partial_{\mathbf{w}} L(\mathbf{w}) = 2\beta \cdot \mathbf{w} + \sum_{n=1}^{N} \Phi(\mathbf{x}^{(n)}, \mathbf{h}_{y^{*(n)}}^{(n)}, y^{*(n)}) - \sum_{n=1}^{N} \Phi(\mathbf{x}^{(n)}, \mathbf{h}_{y^{(n)}}^{(n)}, y^{(n)}) \quad (8)$$

Given the sub-gradient $\partial_{\mathbf{w}} L(\mathbf{w})$ computed according to Eq. (8), we can minimize $L(\mathbf{w})$ using the method in [4]. In order to extend the algorithm to handle more general scenarios involving multi-valued or continuous-valued attributes, we can simply modify the maximization over $\mathbf{h}$ in Eq. (6,7) accordingly. For example, $\arg\max_{\mathbf{h}}$ will be replaced by some continuous optimization in the case of continuous attributes.

5 Attribute Relation Graph

We now describe how to build the attribute relation graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. In order to keep the inference problem in Eq. (2) tractable, we will assume $\mathcal{G}$ is a tree-structured graph. Our approach is inspired by the Chow-Liu algorithm [2] for learning Bayesian network structures.

A vertex $j \in \mathcal{V}$ corresponds to the j-th attribute. An edge $(j, k) \in \mathcal{E}$ means the j-th and the k-th attributes have dependencies. In practice, the dependencies between certain attribute pairs might be weaker than others, i.e. the value of one attribute does not provide much information about the value of the other one. We can build a graph that only contains edges corresponding to those strong dependencies. The graph $\mathcal{G}$ could be built manually by human experts. Instead, we adopt an automatic process to build $\mathcal{G}$ by examining the co-occurrence statistics of attributes in the training data. First, we measure the amount of dependency between the j-th and the k-th attributes using the normalized mutual information defined as $\text{NormMI}(j, k) = \frac{\text{MI}(j,k)}{\min\{H(j), H(k)\}}$, where $\text{MI}(j, k)$ is the mutual

information between the j-th and the k-th attributes, and $H(j)$ is the entropy of the j-th attribute. Both $\mathrm{MI}(j,k)$ and $H(j)$ can be easily calculated using the empirical distributions $\tilde{p}(h_j)$, $\tilde{p}(h_k)$ and $\tilde{p}(h_j, h_k)$ estimated from the training data.

A large $\mathrm{NormMI}(j,k)$ means a strong interaction between the j-th and the k-th attributes. We assign a weight $\mathrm{NormMI}(j,k)$ to the connection (j,k), then run a maximum spanning tree algorithm to find the edges $\mathcal{E}$ to be included in the attribute relation graph $\mathcal{G}$. Similar ideas have been used in [13] to find correlations between video annotations. The attribute relation graph with 64 attributes built from our training data is shown in Fig. 2.

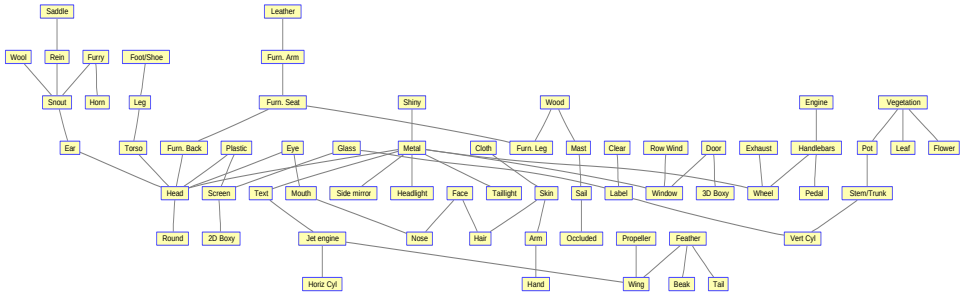

Fig. 2. Visualization of the attribute relation graph learned from the training images of the a-Pascal dataset.

6 Other Loss Functions

This paper mainly deals with multi-class classification problems, where the performance of an algorithm is typically measured by its overall accuracy. It turns out we can modify the learning approach in Sec. 3 to directly optimize other performance measurements. In this section, we show how to adapt the learning objective so it optimizes a more sensible measurement for problems involving highly skewed class distributions.

First we need a new interpretation of Eq. (3). From Eq. (3), it is easy to show $\xi^{(n)} \geq \Delta(y^*(n), y^{(n)})$, where $y^*(n) = \arg\max_y f_{\mathbf{w}}(\mathbf{x}^n, y)$ is the predicted class label of $\mathbf{x}$ by the model $f_{\mathbf{w}}$. So $\xi^{(n)}$ can be interpreted as an upper bound of the loss incurred on $\mathbf{x}^{(n)}$ by the model. The cumulative loss on the whole training data is then upper bounded by $\sum_{n=1}^{N} \xi^{(n)}$. In the case of 0-1 loss, the cumulative loss is exactly the number of training examples incorrectly classified by the model, which is directly related to the overall training error. So we can interpret Eq. (3) as minimizing (an upper bound of) the overall training error, with a regularization term $\beta||\mathbf{w}||^2$.

If the distribution of the classes is highly skewed, say 90% of the data are of a particular class, the overall accuracy is not an appropriate metric for measuring

the performance of an algorithm. A better performance measure is the mean per-class accuracy defined as follows. Let n_{pq} $(p, q \in \mathcal{Y})$ be the number of examples in class p being classified as class q. Define $m_p = \sum_q n_{pq}$, i.e. m_p is the number of examples with class p. Then the mean per-class accuracy is calculated as $1/|\mathcal{Y}| \times \left(\sum_{p=1}^{|\mathcal{Y}|} n_{pp}/m_p\right)$.

We can define the following new loss function that properly adjust the loss according to the distribution of the classes on the training data:

$$\Delta_{\text{new}}(y, y^{(n)}) = \begin{cases} \frac{1}{m_p} \text{ if } y \neq y^{(n)} \text{ and } y^{(n)} = p \\ 0 \quad \text{otherwise} \end{cases} \tag{9}$$

It is easy to verify that $\sum_{n=1}^{N} \Delta_{\text{new}}(y^*(n), y^{(n)})$ directly corresponds to the mean per-class accuracy on the training data. The optimization in Eq. (3) with Δ_{new} will try to directly maximize the mean per-class accuracy, instead of the overall accuracy. This learning algorithm with Δ_{new} is very similar to that with $\Delta_{0/1}$. All we need to do is use Δ_{new} in Eq. (3).

Our learning approach can also be extended for detection tasks [8]. In that case, we can adapt our algorithm to directly optimize other metrics more appropriate for detections (e.g. F-measure, area under ROC curve, or the 50% overlapping criterion in Pascal VOC challenge [5]) using the technique in [10, 15]. We omit the details due to space constraints. The flexibility of optimizing different performance measurements is an important advantage of the max-margin learning method compared with other alternatives, e.g. the hidden conditional random fields [14].

7 Experiments

We test our algorithm on two datasets (called **a-Pascal** and **a-Yahoo**) introduced in [7]. The first dataset (a-Pascal) contains 6340 training images and 6355 test images collected from Pascal VOC 2008 challenge. Each image is assigned one of the 20 object class labels: people, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, and TV/monitor. Each image also has 64 binary attribute labels, e.g. "2D boxy", "has hair","shiny", etc. The second dataset (a-Yahoo) is collected for 12 object categories from Yahoo images. Each image in a-Yahoo is described by the same set of 64 attributes. But the object class labels in a-Yahoo are different from those in a-Pascal. Object categories in a-Yahoo are: wolf, zebra, goat, donkey, monkey, statue of people, centaur, bag, building, jet ski, carriage, and mug.

We follow the experiment setup in [7] as close as possible. However, there is one caveat. These two datasets are collected to study the problem of *attribute prediction*, not *object class prediction*. Farhadi et al. [7] use the training images in a-Pascal to learn their model, and test on both the test images in a-Pascal and images in a-Yahoo. We are interested in the problem of *object class prediction*, so we cannot use the model trained on a-Pascal to predict the class labels for images

in a-Yahoo, since they have different object categories. Instead, we randomly split a-Yahoo dataset into equal training/testing sets, so we can train a model on a-Yahoo training set and test on a-Yahoo test set.

We use the training images of a-Pascal to build the attribute relation graph using the method in Sec. 5. The graph is shown in Fig. 2. We use the exact same graph in the experiments on the a-Yahoo dataset. In order to do a fair comparison with [7], we use exactly the same image features (called *base feature* in [7]) in their work. Each image is represented as a 9751-dimensional feature vector extracted from information on color, texture, visual words, and edges. Note that since the image features are extracted from the whole image, we have essentially eliminated the weakly labeled data assumption in [19].

Figure 3 (left) shows the confusion matrix of our model trained with $\Delta_{0/1}$ on the a-Pascal dataset. Table 1 summarizes our results compared with other baseline methods. Since this dataset is heavily biased toward "people" category, we report both overall and mean per class accuracies. Here we show the results of our approach with $\Delta_{0/1}$ and Δ_{new}. The baseline algorithm is to train an SVM classifier based on the base features. To make a fair comparison, we also report results of SVM with $\Delta_{0/1}$ and Δ_{new}. We also list the result of the baseline algorithm taken from [7] and the best reported result in [7]. The best reported result in [7] is obtained by performing sophisticated feature selection and extracting more semantic attributes. We can see that both of our models outperform the baseline algorithms. In particular, the mean per class accuracies of our models are significantly better. It is also interesting to notice that models (both our approach and SVMs) trained with Δ_{new} achieve lower overall accuracies than $\Delta_{0/1}$, but higher mean per class accuracies. This is exactly what we expect, since the former optimizes an objective directly tied to the mean per class accuracy, while the latter optimizes one directly tied to the overall accuracy.

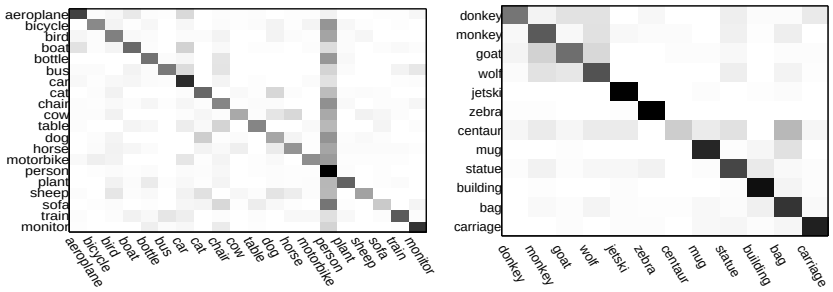

Fig. 3. Confusion matrices of the classification result of our approach with $\Delta_{0/1}$ on the a-Pascal (left) and a-Yahoo (right) datasets. Horizontal rows are ground truths, and vertical columns are predictions. Each row is normalized to sum to 1. The mean per class accuracy is calculated by averaging the main diagonal of this matrix. Dark cells correspond to high values.

The results on a-Yahoo are summarized in Table 2. Here we compare with baseline SVM classifiers using the base features. Farhadi et al. [7] did not perform object category prediction on this dataset, so we cannot compare with them. On this dataset, the performances of using $\Delta_{0/1}$ and Δ_{new} are relatively similar. We believe it is because this dataset is not heavily biased toward any particular class. So optimizing the overall accuracy is not very different from optimizing the mean per-class accuracy. But the results still show the benefits of attributes for object classification. Figure 3(right) shows the confusion matrix of our approach trained with $\Delta_{0/1}$ on this dataset.

Table 1. Results on the a-Pascal dataset. We report both overall and mean per class accuracies, due to the fact that this dataset is heavily biased toward "people" category.

method	overall	mean per-class
Our approach with $\Delta_{0/1}$	**62.16**	**46.25**
Our approach with Δ_{new}	**59.15**	**50.84**
SVM with $\Delta_{0/1}$	58.77	38.52
SVM with Δ_{new}	53.74	44.04
[7] (base features+SVM)	58.5	34.3
[7] (best result)	59.4	37.7

Table 2. Results on the a-Yahoo dataset. Similarly, we report both overall and mean per class accuracies.

method	overall	mean per-class
Our approach with $\Delta_{0/1}$	**78.67**	**71.45**
Our approach with Δ_{new}	**79.88**	**73.31**
SVM with $\Delta_{0/1}$	74.43	65.96
SVM with Δ_{new}	74.51	66.74

8 Conclusion

We have presented a discriminatively trained latent model for joint modelling of object classes and their visual attributes. Different from previous work [7, 19], our model encapsulates the correlations among different attributes via the attribute relation graph built from training data and directly optimize the classification accuracy. Our model is also flexible enough to be easily modified according to different performance measurements. Our experimental results clearly demonstrate that object naming can benefit from inferring attributes of objects. Our work also provides a rather general way of solving many other classification tasks involving auxiliary labels. We have successfully applied a similar technique to

recogize human actions from still images by considering the human poses as auxiliary labels [22].

References

1. Barrow, H.G., Tenenbaum, J.M.: Recovering intrinsic scene characteristics from images. In: Computer Vision Systems. Academic Press (1978)
2. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory 14(3), 462–467 (1968)
3. Desai, C., Ramanan, D., Fowlkes, C.: Discriminative models for static human-object interactions. In: Workshop on Structured Models in Computer Vision (2010)
4. Do, T.M.T., Artieres, T.: Large margin training for hidden markov models with partially observed states. In: ICML (2009)
5. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL visual object classes (VOC) challenge. IJCV 88(2), 303–338 (2010)
6. Farhadi, A., Endres, I., Hoiem, D.: Attribute-centric recognition for cross-category generalization. In: IEEE CVPR (2010)
7. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing objects by their attributes. In: IEEE CVPR (2009)
8. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: IEEE CVPR (2008)
9. Ferrari, V., Zisserman, A.: Learning visual attributes. In: NIPS. MIT Press (2007)
10. Joachims, T.: A support vector method for multivariate performance measures. In: ICML (2005)
11. Kumar, N., Berg, A.C., Belhumeur, P.N., Nayar, S.K.: Attribute and simile classifiers for face verification. In: IEEE ICCV (2009)
12. Lampert, C.H., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: IEEE CVPR (2009)
13. Qi, G.J., Hua, X.S., Rui, Y., Tang, J., Mei, T., Zhang, H.J.: Correlative multi-label video annotation. In: ACM Multimedia (2007)
14. Quattoni, A., Wang, S., Morency, L.P., Collins, M., Darrell, T.: Hidden conditional random fields. IEEE PAMI 29(10), 1848–1852 (June 2007)
15. Ranjbar, M., Mori, G., Wang, Y.: Optimizing complex loss functions in structured prediction. In: ECCV (2010)
16. Taskar, B., Lacoste-Julien, S., Jordan, M.I.: Structured prediction, dual extragradient and Bregman projections. JMLR 7, 1627–1653 (2006)
17. Tran, D., Forsyth, D.: Configuration estimates improve pedestrian finding. In: NIPS. MIT Press (2008)
18. Vaquero, D.A., Feris, R.S., Tran, D., Brown, L., Hampapur, A., Turk, M.: Attribute-based people search in surveillance environments. In: IEEE Workshop on Applications of Computer Vision (2009)
19. Wang, G., Forsyth, D.A.: Joint learning of visual attributes, object classes and visual saliency. In: IEEE ICCV (2009)
20. Wang, G., Hoiem, D., Forsyth, D.: Building text features for object image classification. In: IEEE CVPR (2009)
21. Wang, Y., Mori, G.: Max-margin hidden conditional random fields for human action recognition. In: IEEE CVPR (2009)
22. Yang, W., Wang, Y., Mori, G.: Recognizing human actions from still images with latent poses. In: IEEE CVPR (2010)