

Toward Scene Recognition by Discovering Semantic Structures and Parts

by

Guang-Tong Zhou

M.Sc., Shandong University, 2010

B.Sc., Shandong University, 2007

Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Science

© **Guang-Tong Zhou 2015**
SIMON FRASER UNIVERSITY
Fall 2015

All rights reserved.

However, in accordance with the *Copyright Act of Canada*, this work may be reproduced without authorization under the conditions for “Fair Dealing.” Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.

Approval

Name: Guang-Tong Zhou
Degree: Doctor of Philosophy
Title: *Toward Scene Recognition by Discovering Semantic Structures and Parts*
Examining Committee: **Dr. Ping Tan** (chair)
Assistant Professor

Dr. Greg Mori
Senior Supervisor
Associate Professor

Dr. Oliver Schulte
Supervisor
Associate Professor

Dr. Mark Drew
Examiner
Professor

Dr. Derek Hoiem
External Examiner
Assistant Professor
Department of Computer Science
University of Illinois at Urbana-
Champaign

Date Defended: 17 November 2015

Abstract

Scene recognition is a fundamental and open problem in computer vision. It is an essential component of a variety of real-world applications, including image search, robotics, social media analysis, and many others.

The key to success in scene recognition is to well understand the rich semantics embedded in scenes. For example, it is intuitive to label *airport* for a scene of *sky*, *airplane*, *road*, and *building*. In this thesis, we identify two directions for exploiting scene semantics. On one hand, we advocate for the discovery of scene parts that correspond to various semantic components in scenes, like objects and surfaces. On the other hand, we promote the discovery of scene structures that capture the spatial relations among scene parts, like *sky-above-airplane*. By leveraging scene parts and structures in scene recognition, we are able to build strong recognition systems.

Our contributions are two-fold. First, we propose two clustering algorithms for the data-driven discovery of semantics in visual data. In detail, we develop latent maximum-margin clustering to model semantics as latent variables, and hierarchical maximum-margin clustering to discover tree-structured semantic hierarchies. Our second contribution is the development of two scene recognition methods that leverage scene structure discovery and part discovery. The first method recognizes scene by considering a scene image as a structured collage of objects. The second method discovers scene parts that are both discriminative and representative for scene recognition.

Keywords: Scene recognition; semantics discovery; maximum-margin clustering; hierarchical clustering; latent variable models

Dedication

To my mother, my father and my younger brother.

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Greg Mori, for his supervision during my Ph.D. study at Simon Fraser University. Greg has been an excellent role model, both professional and personally. Whenever I got a research or life problem, Greg is always there ready to help, with his immense knowledge, invaluable inspiration, continuing encouragement and great patience. I simply could not imagine a better supervisor.

I owe sincere thanks to Dr. Leonid Sigal for his mentorship during my internship at Disney Research. It is a wonderful experience to work closely with a world-class researcher like Leon, and learn from his advisement and expertise. I am also grateful to Ivailo Ivanov and Alexei Potiagalov for supervising my internship at SAP. Thank you for teaching me how to transform research ideas into practical software systems.

Thanks to Dr. Tian Lan and Dr. Weilong Yang for their detailed mentoring and generous help when I first started my exploration in the computer vision world. Tian and Weilong are exceptionally smart and knowledgeable. It is my great pleasure to share the time with them.

I would also like to thank my committee members. I am grateful to Dr. Oliver Schulte for personal encouragement and constructive suggestions throughout my entire Ph.D. study. I appreciate Dr. Mark Drew for his insightful comments and feedback. It is a great honor to have Dr. Derek Hoiem as my external examiner. Derek's work on scene understanding has always inspired my own research. I am also thankful to Dr. Ping Tan for serving as the chair of my committee.

I consider myself very lucky to have the chance to collaborate with so many wonderful people. Special thanks to Dr. Mark Schimdt, Dr. Sung Ju Hwang and Dr. Arash Vahdat who helped me to shape research ideas clearly and concretely. I am also grateful to all my collaborators including: Hexiang Hu, Zhiwei Deng, Lei Chen, Mehran Khodabandeh, Dr. Hossein Hajimirsadeghi, Dr. Zicheng Liao, Dr. Mehrsan Roshtkhari and Dr. Stephen Se.

I own much to my friends and labmates in Simon Fraser University, Disney Research, SAP and elsewhere. My life would not be so enjoyable without your sharing, blessing and encouragement. My sincerest thanks to Shuyang Sun and Da Huang, for being wonderful friends and roommates.

Finally, I owe my deepest thanks to my family – my mother, my father and my younger brother. No words can express my gratitude for all the life-time sacrifices they have made. This thesis is impossible without their love and support.

Table of Contents

Approval	ii
Abstract	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vi
List of Tables	ix
List of Figures	x
1 Introduction	1
1.1 Understanding Scene Recognition	2
1.1.1 Why Should We Care About Scene Recognition?	2
1.1.2 What Is the Difference From Object Recognition?	2
1.1.3 What Drives Success in Scene Recognition?	2
1.2 Exploiting Semantics in Scenes	4
1.2.1 Discovering Scene Structures	5
1.2.2 Discovering Scene Parts	5
1.3 Related Work	5
1.3.1 Discovering Scene Structures	6
1.3.2 Discovering Scene Parts	6
1.3.3 Clustering for Semantics Discovery	7
1.4 Contributions	8
1.4.1 Latent Maximum-Margin Clustering	8
1.4.2 Hierarchical Maximum-Margin Clustering	8
1.4.3 Scene Recognition by Semantic Structure Discovery	9
1.4.4 Scene Recognition by Semantic Part Discovery	9
1.5 Thesis Organization	9

2	Latent Maximum-Margin Clustering	10
2.1	Overview	10
2.2	Related Work	11
2.3	Latent Maximum-Margin Clustering	12
2.3.1	Maximum-Margin Clustering	13
2.3.2	Latent Maximum-Margin Clustering	14
2.3.3	Optimization	15
2.4	Tag-Based Video Clustering	16
2.5	Experiments	17
2.5.1	Results	19
2.6	Summary	20
3	Hierarchical Maximal-Margin Clustering	22
3.1	Overview	22
3.2	Related Work	24
3.3	Hierarchical Maximum-Margin Clustering	25
3.4	Optimization	27
3.4.1	Building the Hierarchy	27
3.4.2	Splitting A Node	28
3.5	Experiments	30
3.5.1	Results	33
3.6	Summary	35
4	Scene Recognition by Semantic Structure Discovery	39
4.1	Overview	39
4.2	Related Work	41
4.3	The Object Matching Based Distance Model	42
4.3.1	Model Formulation	42
4.4	Inference	44
4.5	Learning	46
4.6	Experiments	46
4.6.1	Results	48
4.7	Summary	51
5	Scene Recognition by Semantic Part Discovery	54
5.1	Overview	54
5.2	Related Work	56
5.3	Part Discovery for Scene Recognition	57
5.4	Optimization	58
5.5	Experiments	61

5.5.1	Evaluating the Joint Learning Framework	62
5.5.2	Evaluating Part Discovery	62
5.6	Summary	65
6	Conclusions, Limitations and Future Directions	66
6.1	Future Directions	67
	Bibliography	69

List of Tables

Table 2.1	Clustering results (in %) on the three datasets. The figures boldfaced are the best performance among all the compared methods.	19
Table 2.2	Runtime results (in seconds) on the three datasets. Specifically, we report the processing time for running each clustering algorithm on the pre-prepared features. The results are collected on a machine with Intel Xeon 2.8GHz CPU and 16GB memory.	20
Table 3.1	Clustering performance on the four datasets. SP, PS and RI are reported in percentage, and the boldfaced numbers achieve the best performance among flat and hierarchical methods (excluding HMMC variants). The runtime (in seconds) is measured on a machine with Intel Xeon 2.8GHz CPU and 16GB memory.	34
Table 4.1	Object recognition performance (AP and mAP in %) on PASCAL 07. The figures boldfaced are the best performance among <i>Full</i> and state-of-the-art methods. Paired <i>t</i> -tests are also conducted on the AP values to examine <i>Full</i> against all the other methods. We list the returned <i>p</i> -values in the last column, where the boldfaced figures indicate no significance between <i>Full</i> and the compared methods under 5% significance level.	49
Table 4.2	Recognition results (AP and mAP in %) on SUN 09. We only report AP on the 15 largest scene classes due to space limitations. The mAP results are averaged over all 58 classes. See the caption of Table 4.1 for more details.	50
Table 4.3	We list the five most discriminative object categories (<i>i.e.</i> , highly weighed by α in Eq. (4.3)) with respect to each local object feature on sample scene classes. We also provide the five most discriminative spatial relations (<i>i.e.</i> , highly weighed by β in Eq. (4.4)) among these object categories.	51
Table 5.1	Per-class recognition accuracy (%) of the joint learning framework.	62
Table 5.2	Per-class recognition accuracy (%) of part discovery methods.	63
Table 5.3	Per-class recognition accuracy (%) using various sets of parts.	63

List of Figures

Figure 1.1	Visualization of three sample scenes. (a) and (b) are 2D images from [128], and (c) is a 3D stereo from [130].	2
Figure 1.2	Visualization of the scene and object categories in the SUN dataset [128, 127]. There are a total of 131,072 images spanning in 908 categories, and 326,582 manually segmented objects of 5,650 object categories. The area of each word is proportional to the frequency of that scene/object category. The figures are from [127].	3
Figure 1.3	A scene recognition pipeline based on semantic scene structure discovery. The left image is a sample <i>airport</i> scene with object annotations such as <i>sky, airplane, tree, road, etc.</i> The discovered structure for this scene image is visualized in the right side of the figure. Note that it is intuitive to reason about the scene category, <i>airport</i> , from the discovered semantic scene structure.	4
Figure 2.1	Two videos represented by the latent tag model. Please refer to the text for details about \mathcal{T} and \mathbf{h} . Note that the cluster labels (<i>i.e.</i> , <i>feeding animal</i> and <i>board trick</i>) are unknown beforehand. They are added for a better understanding of the video content and the latent tag representations. . . .	16
Figure 2.2	Four sample clusters from TRECVID MED 11. We label each cluster by the dominating video class, <i>e.g.</i> , <i>woodworking project, parade</i> , and visualize the top-3 scored videos. A “✓” sign indicates that the video label is consistent with the cluster label; otherwise, a “✗” sign is used. The two “mis-clustered” videos are on <i>parkour</i> (left) and <i>feeding animal</i> (right). Below each video, we show the top eight inferred tags sorted by the potential calculated from Eq. (2.8).	21
Figure 3.1	A sample MCF network. The edge settings are formatted as: “[<i>lower capacity, upper capacity</i>], <i>cost</i> ”. See text for details.	29
Figure 3.2	The semantic hierarchies of AWA, VEHICLE and IMAGENET. Note that AWA has 50 fine-grained animal classes, VEHICLE has 20 vehicle classes, and IMAGENET has 20 non-animal, non-vehicle classes.	31
Figure 3.3	HMMC model sparsity. See text for details.	35

Figure 3.4	SP result w.r.t. different settings of F . Here we fix $K = 2$ on all the four datasets.	36
Figure 3.5	SP result w.r.t. different settings of K . Here we fix F as the number of ground-truth classes in each dataset.	37
Figure 3.6	The learned hierarchy on AWA-ATTR with binary branching. Here we show the results on the first four layers. For each node, we visualize three majority image classes (with the number of images from each class listed below the sample image), and the three most discriminative attributes (ranked by the magnitude of the regularization on the corresponding feature dimension).	38
Figure 4.1	An example showing the object matchings between the <i>airport</i> class and a test image. There are four major object categories in the training <i>airport</i> images: <i>sky</i> , <i>airplane</i> , <i>road</i> and <i>tree</i> . We match the dashed objects from the training side to the objects in the test image, from which the class-to-image distance is calculated. Spatial relations, e.g., <i>sky-above-airplane</i> , <i>sky-above-road</i> , and <i>tree-next.to-airplane</i> , are also considered in measuring the distance.	40
Figure 4.2	Sample recognition results using our <i>Full</i> model. Each row corresponds to a scene class, and we show the top four ranked positive images and the top two ranked negative images. The title of an image includes the scene class label and a figure indicating the rank of the image according to our learned distance: the smaller the rank, the smaller the distance. For an image, we plot up to four discriminative objects (as listed in Table 4.3) together with the predicted locations. The color of the bounding box shows the relative importance of the objects in distance calculation (sorted by the unary object distance): <i>red</i> > <i>blue</i> > <i>green</i> > <i>yellow</i>	52
Figure 5.1	A minimum cost flow (MCF) solver for the assignment problem in Eq. (5.6). We send a flow of $(1 - r)M$ units from the starting node s to the end node e , by passing M patch nodes and K cluster nodes, with the assignment cost c_{jt} defined in Eq. (5.7). The edge settings are formatted as: “[<i>lower capacity</i> , <i>upper capacity</i>], <i>cost</i> ”.	60
Figure 5.2	Qualitative visualization of the learned parts on 12 scenes (e.g., <i>church</i> , <i>closet</i> , <i>movietheater</i> , etc). For each scene, we show three parts selected by our scene recognition models. Each part accounts for one row in the visualization, where the left-most plot visualizes the part model, and the rest shows three sample patches from the corresponding cluster. Note how discriminative and representative the discovered parts are for the scenes.	64

Chapter 1

Introduction

Scene understanding is a fundamental and open problem in computer vision. By “scene” we mean a view of a real-world environment that typically arranges objects and surfaces¹ in a meaningful way. A scene could either be indoor (*e.g., kitchen, bedroom*) or outdoor (*e.g., beache, airport*), depicted in 2D images or 3D stereos – see Figure 1.1 for a few examples. There is no universal scope for what to understand in a scene, but the central task of scene understanding is to reason about the scene semantics and natural properties like the category and functionality of a scene. The ability to understand scenes assists computer systems in perceiving the world around us, and it goes to the very heart of “AI-complete” [134].

The focus of this thesis is on scene recognition, a sub-task of scene understanding. Specifically, we study the most dominant and fundamental scene understanding problem in the current computer vision literature – scene recognition in 2D images, which aims at classifying a scene image into a semantic category that best describes and summarizes the scene environment. Scene recognition is an essential component of a variety of real-world applications, including image search, robotics, social media analysis, and many others. For the variety of scene categories, Figure 1.2 (a) shows the set used in the SUN dataset [128, 127] – a state-of-the-art benchmark for scene recognition. Note that we use the term “scene category” and “scene class” interchangeably hereafter in this thesis.

We have observed substantial progresses in scene recognition over the past decade, and a great deal of features and models have been proposed to handle the task. In this thesis, we tackle scene recognition by exploiting scene semantics. We believe that recognizing the scene category of a given image relies heavily on how well we represent and understand the semantics in the scene.

Before diving into the detailed solutions, it always helps to identify the nature of the problem. Thus, we describe our understanding of scene recognition in the following section.

¹For simplicity, in the rest of this thesis, we consider surfaces (*e.g., grass, road, etc.*) as generalized objects, and use the term “object” to represent both surfaces and objects.



Figure 1.1: Visualization of three sample scenes. (a) and (b) are 2D images from [128], and (c) is a 3D stereo from [130].

1.1 Understanding Scene Recognition

In this section, we are interested in a deep understanding of scene recognition, including the motivation for recognizing scenes, the difference from object recognition, and the key characteristics to success. We will address these three aspects by answering the three questions in the following.

1.1.1 Why Should We Care About Scene Recognition?

The goal of scene recognition is to identify the semantic scene category of a given image, and the semantic category encapsulates valuable knowledge of the scene contents. Knowing the scene category helps computer systems to perceive the scene, by understanding what to do with the objects in the scene, and what functions to accomplish within the scene. As highlighted in [85], with scene recognition, “we therefore include the perception of function as a proper – indeed, crucial – subject for vision science”.

1.1.2 What Is the Difference From Object Recognition?

One may easily bias scene recognition as identifying representative objects in the given image. However, the concept of scene category typically have a higher level of abstraction than objects (*e.g.*, *airport* scenes versus *airplane* objects). Specifically, scene recognition looks at the whole picture, while object recognition always focuses on a sub-region of the image. In general, understanding objects assists in scene recognition, since the semantic category of a scene can be derived from the objects present in the scene. Figure 1.2 (b) shows common object categories in our daily experience – notice the correlation between scene categories (Figure 1.2 (a)) and object categories (Figure 1.2 (b)).

1.1.3 What Drives Success in Scene Recognition?

A decade ago, the standard approach for scene recognition was to first extract visual appearance descriptors (such as SIFT [73] and GIST [83]) and then build a classification model (like Support

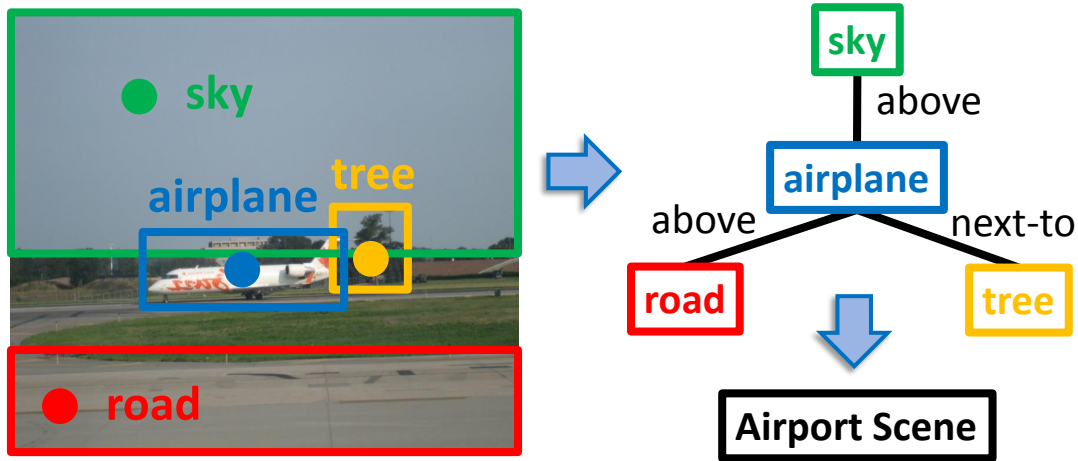


Figure 1.3: A scene recognition pipeline based on semantic scene structure discovery. The left image is a sample *airport* scene with object annotations such as *sky*, *airplane*, *tree*, *road*, etc. The discovered structure for this scene image is visualized in the right side of the figure. Note that it is intuitive to reason about the scene category, *airport*, from the discovered semantic scene structure.

Vector Machine, *i.e.*, SVM [15]) on top of it. However, the bottleneck of this approach comes with the representation power of visual appearance descriptors, which usually summarize low-level pixel information (such as gradient and color) in a small local patch. The low-level descriptors have problems in handling high variations and deformations of visual appearance, resulting in sub-optimal scene recognition performance.

Over the past decade, researchers have changed the focus from learning with low-level descriptors to exploiting semantics in scene images, since it is well aligned with our human’s way of thinking. If we were given a scene image and asked to categorize it, it is likely that we would first find out the objects in the image, and then reason about the scene category based on this collection of objects. This intuition is further evidenced by the need of “focused attention” in natural scene perception [26, 14]. Note that the objects provide invaluable semantic components for us to understand scenes, by removing the complexity in handling the high variations and deformations of scene appearance.

We believe that exploiting semantics is the key for computer vision to success in scene recognition – recognizing the scene category relies heavily on how well we represent and understand the semantics embedded in the scene.

1.2 Exploiting Semantics in Scenes

As emphasized in the last section, understanding semantics can be crucial for scene recognition. We follow this line of research and identify two directions for exploiting the rich semantics in scenes: one by discovering semantic scene structures, and one by discovering semantic scene parts. We describe the details in the following.

1.2.1 Discovering Scene Structures

First, we believe that it is beneficial to organize scene semantics in structured representation. Figure 1.3 shows off our intuition.

Generally speaking, a scene image is typically composed of a number of objects spanning across different regions of the image. The objects contribute to the scene semantics, so as do the spatial relations among these objects. For example, the relation *airplane-below-sky* indicates an *airport* scene, while *airplane-on.top.of-sky* might describe other scenes like *airshow*. We would like to capture the spatial information by representing the scene semantics as a graph, where the graph nodes correspond to objects and graph edges capture object spatial relations. The structured representation summarizes the environment described in the scene image, and is potentially beneficial for reasoning about the scene category.

1.2.2 Discovering Scene Parts

Second, we emphasize the importance of automatic discovery of scene parts for the composition of scene semantics.

Let us revisit Figure 1.3 where structured representation is used to capture scene semantics. An assumption we have made is that the object annotations are readily available on scene images. However, obtaining high-quality and noise-free object annotations is impractical in real-world scenarios due to the following two reasons. On one hand, annotating objects is a very subjective task and people may not agree on the annotations that should be assigned to the same scene image. On the other hand, manually labeling objects in images is tedious and costly, especially when dealing with the large-scale and ever-growing image collections we have in hand.

Given these constraints, we would like to remove the need for object annotation, and automatically discover semantic scene parts in an unsupervised data-driven manner. A scene part is expected to capture a semantic component of the scene, like a surface, an object, or even an object part. The parts are likely non-subjective, as they are discovered from the image data directly. Intuitively, scene recognition benefits from scene part discovery since the parts reveal various semantic components embedded in scenes.

In the following section, we review closely related work in discovering both scene structures and scene parts. Note that we also review state-of-the-art clustering techniques since they are powerful tools for data-driven discovery of visual semantics.

1.3 Related Work

A comprehensive survey of the scene recognition and clustering literature is beyond the scope of this thesis. Here we only review the work most related to ours.

1.3.1 Discovering Scene Structures

It is common to exploit object-level structures for scene understanding. A representative method, call ObjectBank, is proposed by Li *et al.* [68]. Specifically, this method pre-defines a generic collection of labeled objects, and trains an object detector for each object category. For a scene image, it captures object-level structures from a multiple-level spatial pyramid pooling of object detector responses.

Apart from spatial pyramid pooling, scene structures can also be represented by directly modeling object relations, such as co-occurrences, overlaps or spatial layouts. Rabinovich *et al.* [94] exploit pairwise object co-occurrences for contextual object relevance. Malisiewicz and Efros [75] parse scene images by filtering out object pairs with large overlap. Li *et al.* [67] detect groups of objects with Deformable Part Models (DPMs) as the basic elements for scene understanding. Lee and Grauman [66] encode object spatial layout by “object-graph” descriptors. Similarly, Lan *et al.* [62] retrieve images by structured object queries that specify object spatial relations. Note that we have proposed a method following this line of research, and we build semantic structures for scene recognition based on object spatial relations.

1.3.2 Discovering Scene Parts

There has been much work on scene part discovery in recent years. Depending on whether we have supervision on the image-level scene categories, we can roughly divide existing methods into the following two groups.

Part Discovery from Unsupervised Scenes

Part discovery can be performed in a purely unsupervised setting – given a bunch of unlabeled images, the goal is to automatically discover useful parts. As a comprehensive survey, Tuytelaars *et al.* [112] review the detailed literature.

Generally speaking, there are two common techniques to handle this task. The first is clustering. For instance, Kim and Torralba [54] use link analysis to iteratively refine the regions of interest in cluttered images. A recent development by Zhu *et al.* [145] employs a bottom-up, saliency-guided multiple class learning method to perform part localization, discovery, and detector training in an integrated framework. A second technique is latent topic models. To list an example, Russell *et al.* [98] build pLSA [45] and LDA [4] on top of image segmentations to discover semantic categories.

Part Discovery from Supervised Scenes

There is also recent progress in discovering scene parts given only image-level scene categories. Note that the semantic scene categories provide valuable information for identifying useful parts. We group existing work based on the underlying technique used to discover parts.

A first group of methods is inspired by the success of DPMs for part-based object detection [32]. For example, Pandey and Lazebnik [86] discover scene parts by training DPMs directly on scene images. Li *et al.* [67] advocate for automatic detection of groups of parts as the basic elements for scene understanding.

Another line of research starts by training an exemplar part model for every sampled image patch, and then relies on heuristics to refine the final set of scene parts. For instance, Singh *et al.* [108] train exemplar-SVMs [76] for patches, and then iteratively merge similar parts within a scene category. Juneja *et al.* [53] learn exemplar-LDA models [40] for patches, and then select scene parts that occur in some but not many scene categories.

There is also work on clustering patches within each individual scene category, and regarding each cluster as a discovered scene part. Wang *et al.* [120] learn parts via maximum-margin multiple-instance dictionary learning. Doersch *et al.* [24] modify mean-shift clustering to find scene parts that have high density in positive images but low density in negative images. Sun and Ponce [111] initialize parts by k-means clustering, and refined them in a scene recognition model regularized by group sparsity. Note that our method follows this setting, and we propose to discover parts and recognize scenes in a unified learning framework.

1.3.3 Clustering for Semantics Discovery

Clustering is a major task in machine learning that has been actively studied over decades of research [50]. Clustering is typically conducted in an unsupervised manner, with the goal of grouping data instances of similar patterns or structures together. Clustering remains a challenging and active topic of research due to its widespread applicability in a variety of areas, including data-driven discovery, visualization, computer vision, information retrieval, natural language processing, *etc.* Popular clustering methods include k-means clustering [42, 72], k-medoids clustering [51], mixture models [96], normalized cuts [107], spectral clustering [80] and affinity propagation [34].

In this thesis, we focus on two advanced clustering techniques that enable us to explore the rich semantics embedded in data: maximum-margin clustering and hierarchical clustering. We describe the details in the following.

Maximum-margin Clustering

Recent development in maximum-margin techniques has led to the invention of maximum-margin clustering (MMC) [131, 115, 138, 139, 70, 11], which aims to learn both the separating hyperplanes that separate clusters of data, and the label assignments of instances to the clusters. Note that the learned cluster-specific models equip MMC with unique features over conventional clustering methods. Conceptually, MMC is favorable for data-driven semantics discovery since it captures semantics in the learned cluster-specific models. Empirically, MMC usually generates better performance, largely due to the discriminative margin separation criterion imposed among clusters.

MMC has also shown its success in computer vision applications. For example, Zhang *et al.* [138] conduct MMC based image segmentation. Farhadi and Tabrizi [30] find different view points of human activities via MMC. Wang and Cao [121] incorporate MMC to discover geographical clusters of beach images. Hoai and Zisserman [44] form a joint framework of maximum-margin classification and clustering to improve sub-categorization.

Hierarchical Clustering

The clustering techniques we have mentioned above are all flat clustering methods, where the clusters are obtained via one split of the data. Hierarchical clustering methods, on the other hand, are typically built based on hierarchical tree structures of multiple splits. They employ either top-down clustering strategies that recursively split clusters into fine-grained clusters [6, 110, 38], or bottom-up clustering strategies that recursively group smaller clusters into larger ones [77].

Hierarchical clustering has been extensively studied for its benefits over its flat counterparts – it is able to discover hierarchical tree structures (*e.g.*, taxonomies) that better represent semantics in many real-world data distributions.

1.4 Contributions

We identify our contributions in this section. In short, we first propose two data-driven clustering algorithms for discovering semantics in visual data. We then tackle scene recognition by discovering semantic scene structures and parts. In the following, we highlight our main contributions in detail.

1.4.1 Latent Maximum-Margin Clustering

We first present a maximum-margin framework that clusters data using latent variables. Using latent representations enables our framework to model unobserved semantic information embedded in data. We implement our idea by large-margin learning, and develop an alternating descent algorithm to effectively solve the resultant non-convex optimization problem. This work has been reported in [141].

1.4.2 Hierarchical Maximum-Margin Clustering

Second, we present a hierarchical version of maximum-margin clustering to discover semantic hierarchical tree structures. Our method extends beyond flat maximum-margin clustering, and performs clustering recursively in a top-down manner. We propose an effective greedy splitting criteria for selecting which cluster to split next, and employ regularizers to capture the semantics of feature sharing within each split and feature competition among different layers of splits. This work has been reported in [140].

1.4.3 Scene Recognition by Semantic Structure Discovery

Next, we work on the problem of scene recognition. We first learn a class-to-image distance function that matches object-level scene structures. The set of objects in training images for a scene category are treated as a structured collage to represent scene semantics. When presented with a test image, the best matching between this collage of training image objects and those in the test image is found. This work has been reported in [142].

1.4.4 Scene Recognition by Semantic Part Discovery

Finally, we set our goal as discovering semantic parts that are discriminative with each other and representative for recognizing scenes. We address the problem via a joint learning of parts and scenes. For discriminativeness, we cluster image patches using the maximum-margin clustering technique we have introduced above. For representativeness, we build scene recognition models on top of part-based image representation, and apply sparse regularization to select representative parts. We optimize patch clustering, part model learning and scene recognition in a joint framework.

1.5 Thesis Organization

The organization of this thesis is as follows. We first describe latent maximum-margin clustering in Chapter 2, and hierarchical maximum-margin clustering in Chapter 3. Then we present our scene recognition method that leverages semantic scene structure discovery in Chapter 4, followed by Chapter 5 that proposes semantic scene part discovery for scene recognition. Finally, Chapter 6 concludes this thesis, identifies limitations of the proposed methods, and discusses promising directions for future work.

Chapter 2

Latent Maximum-Margin Clustering

In this chapter, we present a maximum-margin framework that clusters data using latent variables. Using latent representations enables our framework to model unobserved information embedded in the data. We implement our idea by large-margin learning, and develop an alternating descent algorithm to effectively solve the resultant non-convex optimization problem. We instantiate our latent maximum-margin clustering framework with tag-based video clustering tasks, where each video is represented by a latent tag model describing the presence or absence of video tags. Experimental results obtained on three standard datasets show that the proposed method outperforms non-latent maximum-margin clustering as well as conventional clustering approaches.

2.1 Overview

Clustering is a major task in machine learning and has been extensively studied over decades of research [50]. Given a set of observations, clustering aims to group data instances of similar structures or patterns together. Popular clustering approaches include the k-means algorithm [42, 72], mixture models [96], normalized cuts [107], and spectral clustering [80]. Recent progress has been made using maximum-margin clustering (MMC) [131], which extends the supervised large margin theory (*e.g.*, SVM) to the unsupervised scenario. MMC performs clustering by simultaneously optimizing cluster-specific models and instance-specific labeling assignments, and often generates better performance than conventional methods [132, 115, 138, 139, 70, 39].

Modeling data with latent variables is common in many applications. Latent variables are often defined to have intuitive meaning, and are used to capture unobserved semantics in the data. As compared with ordinary linear models, latent variable models feature the ability to exploit a richer representation of the space of instances. Thus, they often achieve superior performance in practice. In computer vision, this superiority is best exemplified by the success of deformable part models (DPMs) [32] for object detection. DPMs enhance the representation of an object class by capturing viewpoint and pose variations. They utilize a root template describing the entire object appearance and several part templates. Latent variables are used to capture deformations and appearance varia-

tions of the root template and parts. DPMs perform object detection via search for the best locations of the root and part templates.

Latent variable models are often coupled with supervised learning to learn models incorporating the unobserved variables. For example, DPMs are learned in a latent SVM framework [32] for object detection; similar models have been shown to improve human action recognition [122]. A host of other applications of latent SVMs have obtained state-of-the-art performance in computer vision. Motivated by their success in supervised learning, we believe latent variable models can also help in unsupervised clustering – data instances with similar latent representations should be grouped together in one cluster.

As the latent variables are unobserved in the original data, we need a learning framework to handle this latent knowledge. To implement this idea, we develop a novel clustering algorithm based on MMC that incorporates latent variables – we call this latent maximum-margin clustering (LMMC). The LMMC algorithm results in a non-convex optimization problem, for which we introduce an iterative alternating descent algorithm. Each iteration involves three steps: inferring latent variables for each sample point, optimizing cluster assignments, and updating cluster model parameters.

To evaluate the efficacy of this clustering algorithm, we instantiate LMMC for tag-based video clustering, where each video is modeled with latent variables controlling the presence or absence of a set of descriptive tags. We conduct experiments on three standard datasets: TRECVID MED 11 [84], KTH Actions [104] and UCF Sports [97], and show that LMMC outperforms non-latent MMC and conventional clustering methods.

The rest of this chapter is organized as follows. Section 2.2 reviews related work. Section 2.3 formulates the LMMC framework in detail. We describe tag-based video clustering in Section 2.4, followed by experimental results reported in Section 2.5. Finally, Section 2.6 summarizes this chapter.

2.2 Related Work

Latent variable models: There has been much work in recent years using latent variable models. The definition of latent variables are usually task-dependent. Here we focus on the learning part only. Andrews *et al.* [1] propose multiple-instance SVM to learn latent variables in positive bags. Felzenszwalb *et al.* [32] formulate latent SVM by extending binary linear SVM with latent variables. Yu and Joachims [137] handle structural outputs with latent structural SVM. This model is also known as maximum-margin hidden conditional random fields (MMHCRF) [122]. Kumar *et al.* [59] propose self-paced learning, an optimization strategy that focuses on simple models first. Yang *et al.* [136] kernelize latent SVM for better performance. All of this work demonstrates the power of maximum-margin latent variable models for supervised learning; our framework conducts unsupervised clustering while modeling data with latent variables.

Maximum-margin clustering: MMC was first proposed by Xu *et al.* [131] to extend supervised large-margin methods to unsupervised clustering. Different from the supervised case, where the

optimization is convex, MMC results in non-convex problems. To solve the problems, many approaches are proposed in the past decade. Xu *et al.* [131] and Valizadegan and Rong [115] reformulate the original problem as a semi-definite programming (SDP) problem. Zhang *et al.* [138] employ alternating optimization – finding labels and optimizing a support vector regression (SVR). Li *et al.* [70] iteratively generate the most violated labels, and combine them via multiple kernel learning. Note that the above methods can only solve binary-cluster clustering problems. To handle the multi-cluster case, Xu and Schuurmans [132] extend the SDP method in [131]. Zhao *et al.* [139] propose a cutting-plane method which uses the constrained convex-concave procedure (CCCP) to relax the non-convex constraint. Gopalan and Sankaranarayanan [39] examine data projections to identify the maximum margin. Our framework deals with multi-cluster clustering, and we model data instances with latent variables to exploit rich representations. It is also worth mentioning that MMC leads naturally to the semi-supervised SVM framework [52] by assuming a training set of labeled instances [131, 132]. Using the same idea, we could extend LMMC to semi-supervised learning.

MMC has also shown its success in various computer vision applications. For example, Zhang *et al.* [138] conduct MMC based image segmentation. Farhadi and Tabrizi [30] find different view points of human activities via MMC. Wang and Cao [120] incorporate MMC to discover geographical clusters of beach images. Hoai and Zisserman [44] form a joint framework of maximum-margin classification and clustering to improve sub-categorization.

Tag-based video analysis: Tagging videos with relevant concepts or attributes is common in video analysis. Qi *et al.* [92] predict multiple correlative tags in a structural SVM framework. Yang and Toderici [135] exploit latent sub-categories of tags in large-scale videos. The obtained tags can assist in recognition. For example, Liu *et al.* [71] use semantic attributes (*e.g.*, *up-down motion*, *torso motion*, *twist*) to recognize human actions (*e.g.*, *walking*, *hand clapping*). Izadinia and Shah [49] model low-level event tags (*e.g.*, *people dancing*, *animal eating*) as latent variables to recognize complex video events (*e.g.*, *wedding ceremony*, *grooming animal*).

Instead of supervised recognition of tags or video categories, we focus on unsupervised tag-based video clustering. In fact, recently research collects various sources of tags for video clustering. Schroff *et al.* [103] cluster videos by the capturing locations. Hsu *et al.* [46] build hierarchical clustering using user-contributed comments. Our work uses latent tag models, and our LMMC framework is general enough to handle various types of tags.

2.3 Latent Maximum-Margin Clustering

As stated above, modeling data with latent variables can be beneficial in a variety of supervised applications. For unsupervised clustering, we believe it also helps to group data instances based on latent representations. To implement this idea, we propose the LMMC framework.

LMMC models instances with latent variables. When fitting an instance to a cluster, we find the optimal values for latent variables and use the corresponding latent representation of the instance.

To best fit different clusters, an instance is allowed to flexibly take different latent variable values when being compared to different clusters. This enables LMMC to explore a rich latent space when forming clusters. Note that in conventional clustering algorithms, an instance is usually restricted to have the same representation in all clusters. Furthermore, as the latent variables are unobserved in the original data, we need a learning framework to exploit this latent knowledge. Here we develop a large-margin learning framework based on MMC, and learn a discriminative model for each cluster. The resultant LMMC optimization is non-convex, and we design an alternating descent algorithm to approximate the solution. Next we will briefly introduce MMC in Section 2.3.1, followed by detailed descriptions of the LMMC framework and optimization respectively in Sections 2.3.2 and 2.3.3.

2.3.1 Maximum-Margin Clustering

MMC [131, 138, 139] extends the maximum-margin principle popularized by supervised SVMs to unsupervised clustering, where the input instances are unlabeled. The idea of MMC is to find a labeling so that the margin obtained would be maximal over all possible labelings. If we suppose there are N instances $\{\mathbf{x}_i\}_{i=1}^N$ to be clustered into K clusters, then MMC can be formulated as follows [132, 139]:

$$\begin{aligned}
\min_{\mathcal{W}, \mathcal{Y}, \xi \geq 0} \quad & \frac{1}{2} \sum_{t=1}^K \|\mathbf{w}_t\|^2 + \frac{C}{K} \sum_{i=1}^N \sum_{r=1}^K \xi_{ir}, \\
\text{s.t.} \quad & \sum_{t=1}^K y_{it} \mathbf{w}_t^\top \mathbf{x}_i - \mathbf{w}_r^\top \mathbf{x}_i \geq 1 - y_{ir} - \xi_{ir}, \quad \forall i, r \\
& y_{it} \in \{0, 1\}, \quad \forall i, t \quad \sum_{t=1}^K y_{it} = 1, \quad \forall i
\end{aligned} \tag{2.1}$$

where $\mathcal{W} = \{\mathbf{w}_t\}_{t=1}^K$ are the linear model parameters for each cluster, $\xi = \{\xi_{ir}\}$ ($i \in \{1, \dots, N\}$, $t \in \{1, \dots, K\}$) are the slack variables to allow soft margin, and C is a trade-off parameter. We denote the labeling assignment by $\mathcal{Y} = \{y_{it}\}$ ($i \in \{1, \dots, N\}$, $t \in \{1, \dots, K\}$), where $y_{it} = 1$ indicates that the instance \mathbf{x}_i is clustered into the t -th cluster, and $y_{it} = 0$ otherwise. By convention, we require that each instance is assigned to one and only one cluster, *i.e.*, the last constraint in Eq. (2.1). Moreover, the first constraint in Eq. (2.1) enforces a large-margin between clusters by constraining that the score of \mathbf{x}_i to the assigned cluster is sufficiently larger than the score of \mathbf{x}_i to any other clusters. Note that MMC is an unsupervised clustering method, which jointly estimates the model parameters \mathcal{W} and finds the best labeling \mathcal{Y} .

Enforcing balanced clusters. Unfortunately, solving Eq. (2.1) could end up with trivial solutions where all instances are simply assigned to the same cluster, and we obtain an unbounded margin.

To address this problem, we add cluster balance constraints to Eq. (2.1) that require \mathcal{Y} to satisfy

$$L \leq \sum_{i=1}^N y_{it} \leq U, \forall t \quad (2.2)$$

where L and U are the lower and upper bounds controlling the size of a cluster. Note that we explicitly enforce cluster balance using a hard constraint on the cluster sizes. This is different from [139], a representative multi-cluster MMC method, where the cluster balance constraints are implicitly imposed on the accumulated model scores (*i.e.*, $\sum_{i=1}^N \mathbf{w}_t^\top \mathbf{x}_i$). We found empirically that explicitly enforcing balanced cluster sizes led to better results.

2.3.2 Latent Maximum-Margin Clustering

We now extend MMC to include latent variables. The latent variable of an instance is cluster-specific. Formally, we denote \mathbf{h} as the latent variable of an instance \mathbf{x} associated to a cluster parameterized by \mathbf{w} . Following the latent SVM formulation [32, 137, 122], scoring \mathbf{x} w.r.t. \mathbf{w} is to solve an inference problem of the form:

$$f_{\mathbf{w}}(\mathbf{x}) = \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}), \quad (2.3)$$

where $\Phi(\mathbf{x}, \mathbf{h})$ is the feature vector defined for the pair of (\mathbf{x}, \mathbf{h}) . To simplify the notation, we assume the latent variable \mathbf{h} takes its value from a discrete set of labels. However, our formulation can be easily generalized to handle more complex latent variables (*e.g.*, graph structures [137, 122]).

To incorporate the latent variable models into clustering, we replace the linear model $\mathbf{w}^\top \mathbf{x}$ in Eq. (2.1) by the latent variable model $f_{\mathbf{w}}(\mathbf{x})$. We call the resultant framework latent maximum-margin clustering (LMMC). LMMC finds clusters via the following optimization:

$$\begin{aligned} \min_{\mathcal{W}, \mathcal{Y}, \xi \geq 0} \quad & \frac{1}{2} \sum_{t=1}^K \|\mathbf{w}_t\|^2 + \frac{C}{K} \sum_{i=1}^N \sum_{r=1}^K \xi_{ir}. \\ \text{s.t.} \quad & \sum_{t=1}^K y_{it} f_{\mathbf{w}_t}(\mathbf{x}_i) - f_{\mathbf{w}_r}(\mathbf{x}_i) \geq 1 - y_{ir} - \xi_{ir}, \forall i, r \\ & y_{it} \in \{0, 1\}, \forall i, t \quad \sum_{t=1}^K y_{it} = 1, \forall i \quad L \leq \sum_{i=1}^N y_{it} \leq U, \forall t \end{aligned} \quad (2.4)$$

We adopt the notation \mathcal{Y} from the MMC formulation to denote the labeling assignment. Similar to MMC, the first constraint in Eq. (2.4) enforces the large-margin criterion where the score of fitting \mathbf{x}_i to the assigned cluster is marginally larger than the score of fitting \mathbf{x}_i to any other clusters. Cluster balance is enforced by the last constraint in Eq. (2.4). Note that LMMC jointly optimizes the model parameters \mathcal{W} and finds the best labeling assignment \mathcal{Y} , while inferring the optimal latent variables.

2.3.3 Optimization

It is easy to verify that the optimization problem described in Eq. (2.4) is non-convex due to the optimization over the labeling assignment variables \mathcal{Y} and the latent variables $\mathcal{H} = \{\mathbf{h}_{it}\} (i \in \{1, \dots, N\}, t \in \{1, \dots, K\})$. To solve this problem, we first eliminate the slack variables ξ , and rewrite Eq. (2.4) equivalently as:

$$\min_{\mathcal{W}} \frac{1}{2} \sum_{t=1}^K \|\mathbf{w}_t\|^2 + \frac{C}{K} R(\mathcal{W}), \quad (2.5)$$

where $R(\mathcal{W})$ is the risk function defined by:

$$\begin{aligned} R(\mathcal{W}) &= \min_{\mathcal{Y}} \sum_{i=1}^N \sum_{r=1}^K \max(0, 1 - y_{ir} + f_{\mathbf{w}_r}(\mathbf{x}_i) - \sum_{t=1}^K y_{it} f_{\mathbf{w}_t}(\mathbf{x}_i)). \\ \text{s.t. } &y_{it} \in \{0, 1\}, \forall i, t \quad \sum_{t=1}^K y_{it} = 1, \forall i \quad L \leq \sum_{i=1}^N y_{it} \leq U, \forall t \end{aligned} \quad (2.6)$$

Note that Eq. (2.5) minimizes over the model parameters \mathcal{W} , and Eq. (2.6) minimizes over the labeling assignment variables \mathcal{Y} while inferring the latent variables \mathcal{H} . We develop an alternating descent algorithm to find an approximate solution. In each iteration, we first evaluate the risk function $R(\mathcal{W})$ given the current model parameters \mathcal{W} , and then update \mathcal{W} with the obtained risk value. Next we describe each step in detail.

Risk evaluation: The first step of learning is to compute the risk function $R(\mathcal{W})$ with the model parameters \mathcal{W} fixed. We first infer the latent variables \mathcal{H} and then optimize the labeling assignment \mathcal{Y} . According to Eq. (2.3), the latent variable \mathbf{h}_{it} of an instance \mathbf{x}_i associated to cluster t can be obtained via: $\operatorname{argmax}_{\mathbf{h}_{it}} \mathbf{w}_t^\top \Phi(\mathbf{x}_i, \mathbf{h}_{it})$. Note that the inference problem is task-dependent. For our latent tag model, we present an efficient inference method in Section 2.4.

After obtaining the latent variables \mathcal{H} , we optimize the labeling assignment \mathcal{Y} from Eq. (2.6). Intuitively, this is to minimize the total risk of labeling all instances yet maintaining the cluster balance constraints. We reformulate Eq. (2.6) as an integer linear programming (ILP) problem by introducing a variable ψ_{it} to capture the risk of assigning an instance \mathbf{x}_i to a cluster t . The ILP can be written as:

$$\begin{aligned} R(\mathcal{W}) &= \min_{\mathcal{Y}} \sum_{i=1}^N \sum_{t=1}^K \psi_{it} y_{it}, \\ \text{s.t. } &y_{it} \in \{0, 1\}, \forall i, t \quad \sum_{t=1}^K y_{it} = 1, \forall i \quad L \leq \sum_{i=1}^N y_{it} \leq U, \forall t \end{aligned} \quad (2.7)$$

where $\psi_{it} = \sum_{r=1, r \neq t}^K \max(0, 1 + f_{\mathbf{w}_r}(\mathbf{x}_i) - f_{\mathbf{w}_t}(\mathbf{x}_i))$. This captures the total ‘‘mis-clustering’’ penalties - suppose that we regard t as the ‘‘ground truth’’ cluster label for an instance \mathbf{x}_i , then ψ_{it} measures the sum of hinge losses for all incorrect predictions r ($r \neq t$), which is consistent with the



	Cluster: feeding animal								Cluster: board trick									
video																		
\mathcal{T} :	board	car	dog	food	grass	man	snow	tree	...	board	car	dog	food	grass	man	snow	tree	...
\mathbf{h} :	0	0	1	1	1	1	0	1	...	1	0	0	0	1	1	0	0	...

Figure 2.1: Two videos represented by the latent tag model. Please refer to the text for details about \mathcal{T} and \mathbf{h} . Note that the cluster labels (*i.e.*, *feeding animal* and *board trick*) are unknown beforehand. They are added for a better understanding of the video content and the latent tag representations.

supervised multi-class SVM at a higher level [16]. Eq. (2.7) is a standard ILP problem with $N \times K$ variables and $N + K$ constraints. We use the GNU Linear Programming Kit (GLPK) to obtain an approximate solution to this problem.

Updating \mathcal{W} : The next step of learning is the optimization over the model parameters \mathcal{W} (Eq. (2.5)). The learning problem is non-convex and we use the the non-convex bundle optimization solver in [23]. In a nutshell, this method builds a piecewise quadratic approximation to the objective function of Eq. (2.5) by iteratively adding a linear cutting plane at the current optimum and updating the optimum. Now the key issue is to compute the subgradient $\partial_{\mathbf{w}_t} f_{\mathbf{w}_t}(\mathbf{x}_i)$ for a particular \mathbf{w}_t . Let \mathbf{h}_{it}^* be the optimal solution to the inference problem: $\mathbf{h}_{it}^* = \operatorname{argmax}_{\mathbf{h}_{it}} \mathbf{w}_t^\top \Phi(\mathbf{x}_i, \mathbf{h}_{it})$. Then the subgradient can be calculated as $\partial_{\mathbf{w}_t} f_{\mathbf{w}_t}(\mathbf{x}_i) = \Phi(\mathbf{x}_i, \mathbf{h}_{it}^*)$. Using the subgradient $\partial_{\mathbf{w}_t} f_{\mathbf{w}_t}(\mathbf{x}_i)$, we optimize Eq. (2.5) by the algorithm in [23].

2.4 Tag-Based Video Clustering

In this section, we introduce an application of LMMC: tag-based video clustering. Our goal is to jointly learn video clusters and tags in a single framework. We treat tags of a video as latent variables and capture the correlations between clusters and tags. Intuitively, videos with a similar set of tags should be assigned to the same cluster. We assume the existence of a separate dataset consisting of videos with ground-truth tag labels, from which we train tag detectors independently. The tag detectors are deemed to be weak and noisy because of the high variations of video appearance. During clustering, we are given a set of new videos without the ground-truth tag labels, and our goal is to assign cluster labels to these videos. We run the tag detectors to generate a prediction on how likely a tag appears in a video. Due to the weakness of tag prediction, we incorporate a binary latent variable to indicate whether the tag presents or not in the video.

We employ a latent tag model to represent videos. We are particularly interested in tags which describe different aspects of videos. For example, a video from the cluster *feeding animal* (see Figure 2.1) may be annotated with *dog*, *food*, *man*, *etc.* Assume we collect all the tags in a set \mathcal{T} . For a video being assigned to a particular cluster, we know it could have a number of tags from \mathcal{T} describing its visual content related to the cluster. However, we do not know which tags are

present in the video. To address this problem, we associate latent variables to the video to denote the presence and absence of tags.

Formally, given a cluster parameterized by \mathbf{w} , we associate a latent variable \mathbf{h} to a video \mathbf{x} , where $\mathbf{h} = \{h_t\}_{t \in \mathcal{T}}$ and $h_t \in \{0, 1\}$ is a binary variable denoting the presence/absence of each tag t . $h_t = 1$ means \mathbf{x} has the tag t , while $h_t = 0$ means \mathbf{x} does not have the tag t . Figure 2.1 shows the latent tag representations of two sample videos. We score the video \mathbf{x} according to the model in Eq. (2.3): $f_{\mathbf{w}}(\mathbf{x}) = \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h})$, where the potential function $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h})$ is defined as follows:

$$\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} h_t \cdot \omega_t^\top \phi_t(\mathbf{x}). \quad (2.8)$$

This potential function measures the compatibility between the video \mathbf{x} and tag t associated with the current cluster. Note that $\mathbf{w} = \{\omega_t\}_{t \in \mathcal{T}}$ are the cluster-specific model parameters, and $\Phi = \{h_t \cdot \phi_t(\mathbf{x})\}_{t \in \mathcal{T}}$ is the feature vector depending on the video \mathbf{x} and its tags \mathbf{h} . Here $\phi_t(\mathbf{x}) \in \mathbb{R}^d$ is the feature vector extracted from the video \mathbf{x} , and the parameter ω_t is a template for tag t . In our current implementation, instead of keeping $\phi_t(\mathbf{x})$ as a high dimensional vector of video features, we simply represent it as a scalar score of detecting tag t on \mathbf{x} by a pre-trained binary tag detector. To learn biases between different clusters, we append a constant 1 to make $\phi_t(\mathbf{x})$ two-dimensional.

Now we describe how to infer the latent variable $\mathbf{h}^* = \operatorname{argmax}_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h})$. As there is no dependency between tags, we can infer each latent variable separately. According to Eq. (2.8), the term corresponding to tag t is $h_t \cdot \omega_t^\top \phi_t(\mathbf{x})$. Considering that h_t is binary, we set h_t to 1 if $\omega_t^\top \phi_t(\mathbf{x}) > 0$; otherwise, we set h_t to 0.

2.5 Experiments

We evaluate the performance of our method on three standard video datasets: TRECVID MED 11 [84], KTH Actions [104] and UCF Sports [97]. We briefly describe our experimental setup before reporting the experimental results in Section 2.5.1.

TRECVID MED 11 dataset [84]: This dataset contains web videos collected by the Linguistic Data Consortium from various web video hosting sites. There are 15 complex event categories including *board trick*, *feeding animal*, *landing fish*, *wedding ceremony*, *woodworking project*, *birthday party*, *changing tire*, *flash mob*, *getting vehicle unstuck*, *grooming animal*, *making sandwich*, *parade*, *parkour*, *repairing appliance*, and *sewing project*. TRECVID MED 11 has three data collections: Event-Kit, DEV-T and DEV-O. DEV-T and DEV-O are dominated by videos of the null category, *i.e.*, background videos that do not contain the events of interest. Thus, we use the Event-Kit data collection in the experiments. By removing 13 short videos that contain no visual content, we finally have a total of 2,379 videos for clustering.

We use tags that were generated in Vahdat and Mori [113] for the TRECVID MED 11 dataset. Specifically, this dataset includes “judgment files” that contain a short one-sentence description for each video. A sample description is: “A man and a little boy lie on the ground after the boy has

fallen off his bike”. This sentence provides us with information about presence of objects such as *man*, *boy*, *ground* and *bike*, which could be used as tags. In [113], text analysis tools are employed to extract binary tags based on frequent nouns in the judgment files. Examples of the 74 most frequent tags used in this work are: *music*, *person*, *food*, *kitchen*, *bird*, *bike*, *car*, *street*, *boat*, *water*, *etc.* The complete list of tags are available on our website.

To train tag detectors, we use the DEV-T and DEV-O videos that belong to the 15 event categories. There are 1675 videos in total. We extract HOG3D descriptors [56] and form a 1,000 word codebook. Each video is then represented by a 1,000-dimensional feature vector. We train a linear SVM for each tag, and predict the detection scores on the Event-Kit videos. To remove biases between tag detectors, we normalize the detection scores by z-score normalization. Note that we make no use of the ground-truth tags on the Event-Kit videos that are to be clustered.

KTH Actions dataset [104]: This dataset contains a total of 599 videos of 6 human actions: *walking*, *jogging*, *running*, *boxing*, *hand waving*, and *hand clapping*. Our experiments use all the videos for clustering.

We use Action Bank [100] to generate tags for this dataset. Action Bank has 205 template actions with various action semantics and viewpoints. Randomly selected examples of template actions are: *hula1*, *ski5*, *clap3*, *fence2*, *violin6*, *etc.* In our experiments, we treat the template actions as tags. Specifically, on each video and for each template action, we use the set of Action Bank action detection scores collected at different spatiotemporal scales and correlation volumes. We perform max-pooling on the scores to obtain the corresponding tag detection score. Again, for each tag, we normalize the detection scores by z-score normalization.

UCF Sports dataset [97]: This dataset consists of 140 videos from 10 action classes: *diving*, *golf swinging*, *kicking*, *lifting*, *horse riding*, *running*, *skating*, *swinging (on the pommel horse)*, *swinging (at the high bar)*, and *walking*. We use all the videos for clustering. The tags and tag detection scores are generated from Action Bank, in the same way as KTH Actions.

Baselines: To evaluate the efficacy of LMMC, we implement three conventional clustering methods for comparison, including the k-means algorithm (KM) [42, 72], normalized cut (NC) [107], and spectral clustering (SC) [80]. For NC, the implementation and parameter settings are the same as [107], which uses a Gaussian similarity function with all the instances considered as neighbors. For SC, we use a 5-nearest neighborhood graph and set the width of the Gaussian similarity function as the average distance over all the 5-nearest neighbors. Note that these three methods do not use latent variable models. Therefore, for a fair comparison with LMMC, they are directly performed on the data where each video is represented by a vector of tag detection scores. We have also tried KM, NC and SC on the 1,000-dimensional HOG3D features. However, the performance is worse and is not reported here. Furthermore, to mitigate the effect of randomness, KM, NC and SC are run 10 times with different initial seeds and the average results are recorded in the experiments.

In order to show the benefits of incorporating latent variables, we further develop a baseline called MMC by replacing the latent variable model $f_{\mathbf{w}}(\mathbf{x})$ in Eq. (2.4) with a linear model $\mathbf{w}^T \mathbf{x}$. This is equivalent to running an ordinary maximum-margin clustering algorithm on the video data

Table 2.1: Clustering results (in %) on the three datasets. The figures boldfaced are the best performance among all the compared methods.

	TRECVID MED 11				KTH Actions				UCF Sports			
	PUR	NMI	RI	FM	PUR	NMI	RI	FM	PUR	NMI	RI	FM
LMMC	39.0	28.7	89.5	22.1	92.5	87.0	95.8	87.2	76.4	71.2	92.0	60.0
MMC	36.0	26.6	89.3	20.3	91.3	86.5	95.2	85.5	63.6	62.2	89.2	46.1
SC	28.6	23.6	87.1	20.3	61.0	60.8	75.6	58.2	69.9	70.8	90.6	58.1
KM	27.0	23.8	85.9	20.4	64.8	60.7	84.0	60.6	63.1	66.2	87.9	58.7
NC	12.9	5.7	31.6	12.7	48.0	33.9	72.9	35.1	60.7	55.8	83.4	41.8

represented by tag detection scores. For a fair comparison, we use the same solver for learning MMC and LMMC. The trade-off parameter C in Eq. (2.4) is selected as the best from the range $\{10^1, 10^2, 10^3\}$. The lower bound and upper bounds of the cluster-balance constraint (*i.e.*, L and U in Eq. (2.4)) are set as $0.9\frac{N}{K}$ and $1.1\frac{N}{K}$ respectively to enforce balanced clusters.

Performance measures: Following the convention of maximum-margin clustering [131, 132, 115, 138, 139, 70, 39], we set the number of clusters to be the ground-truth number of classes for all the compared methods. The clustering quality is evaluated by four standard measurements including purity (PUR) [131], normalized mutual information (NMI) [60], Rand index (RI) [95] and balanced F-measure (FM). They are employed to assess different aspects of a given clustering: PUR measures the accuracy of the dominating class in each cluster; NMI is from the information-theoretic perspective and calculates the mutual dependence of the predicted clustering and the ground-truth partitions; RI evaluates true positives within clusters and true negatives between clusters; and FM considers both precision and recall. The higher the four measures, the better the performance.

2.5.1 Results

The clustering results are listed in Table 2.1. It shows that LMMC consistently outperforms the MMC baseline and conventional clustering methods on all three datasets. Specifically, by incorporating latent variables, LMMC improves the MMC baseline by 3% on TRECVID MED 11, 1% on KTH Actions, and 13% on UCF Sports respectively, in terms of PUR. This demonstrates that learning the latent presence and absence of tags can exploit rich representations of videos, and boost clustering performance. Specifically, adopting the latent tag model in LMMC helps to handle tag ambiguity and discover hidden knowledge about the tags embedded in video data. Moreover, LMMC performs better than the three conventional methods, SC, KM and NC, showing the efficacy of the proposed LMMC framework for unsupervised data clustering.

Note that MMC runs on the same non-latent representation as the three conventional methods, SC, KM and NC. However, MMC outperforms them on the two largest datasets, TRECVID MED 11 and KTH Actions, and is comparable with them on UCF Sports. This provides evidence for the effectiveness of maximum-margin clustering as well as the proposed alternating descent algorithm for optimizing the non-convex objective.

Table 2.2: Runtime results (in seconds) on the three datasets. Specifically, we report the processing time for running each clustering algorithm on the pre-prepared features. The results are collected on a machine with Intel Xeon 2.8GHz CPU and 16GB memory.

	TRECVID MED 11	KTH Actions	UCF Sports
LMMC	297.270	9.060	8.480
MMC	5209.510	15.820	82.010
SC	1.215	0.208	0.065
KM	0.112	0.016	0.004
NC	2.550	0.169	0.078

We also collect the clustering runtime results for all the compared methods in Table 2.2. On all three datasets, LMMC and MMC take more time than the three conventional clustering methods. This is because LMMC and MMC need an optimization process to learn model parameters. Besides, it is interesting to note that LMMC costs less time than MMC. The reason we suspect is that learning MMC involves all the tags on all the videos, but learning LMMC only deals with a sparse set of latent tags.

Visualization: We select four clusters from TRECVID MED 11, and visualize the results in Figure 2.2. Please refer to the caption for more details.

2.6 Summary

We have presented a latent maximum-margin framework for unsupervised clustering. By representing instances with latent variables, our method features the ability to exploit the unobserved information embedded in data. We formulate our framework by large-margin learning, and an alternating descent algorithm is developed to solve the resultant non-convex objective. We instantiate our framework with tag-based video clustering, where each video is represented by a latent tag model with latent presence and absence of video tags. Our experiments conducted on three standard video datasets validate the efficacy of the proposed framework. We believe our solution is general enough to be applied in other applications with latent representations, *e.g.* video clustering with latent key segments, image clustering with latent region-of-interest, *etc.* It would also be interesting to extend our framework to semi-supervised learning by assuming a training set of labeled instances.



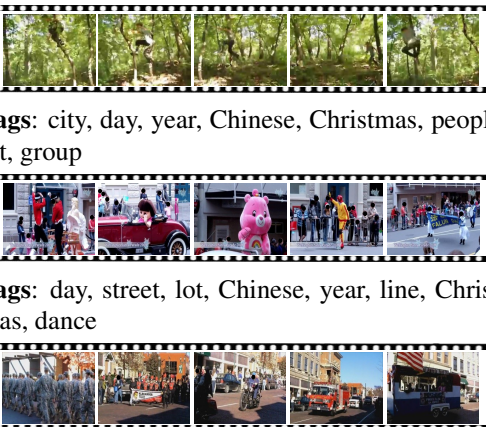
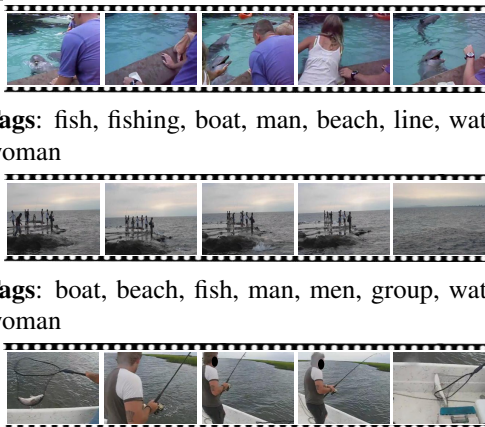
<p style="text-align: center;">Cluster: woodworking project</p>  <p>Tags: piece, wood, machine, lady, indoors, man, kitchen, baby ✓</p> <p>Tags: piece, man, wood, baby, hand, machine, lady, kitchen ✓</p> <p>Tags: wood, piece, baby, indoors, hand, man, lady, bike ✓</p>	<p style="text-align: center;">Cluster: birthday party</p>  <p>Tags: party, birthday, restaurant, couple, wedding ceremony, wedding, ceremony, indoors ✓</p> <p>Tags: birthday, party, restaurant, family, child, wedding ceremony, wedding, couple ✓</p> <p>Tags: party, birthday, restaurant, child, family, wedding ceremony, chicken, couple ✓</p>
<p style="text-align: center;">Cluster: parade</p>  <p>Tags: city, day, year, Chinese, Christmas, people, lot, group ✗</p> <p>Tags: day, street, lot, Chinese, year, line, Christmas, dance ✓</p> <p>Tags: street, day, lot, Chinese, line, year, dancing, dance ✓</p>	<p style="text-align: center;">Cluster: landing fish</p>  <p>Tags: fish, fishing, boat, man, beach, line, water, woman ✗</p> <p>Tags: boat, beach, fish, man, men, group, water, woman ✓</p> <p>Tags: fish, beach, boat, men, man, chicken, truck, move ✓</p>

Figure 2.2: Four sample clusters from TRECVID MED 11. We label each cluster by the dominating video class, *e.g.*, *woodworking project*, *parade*, and visualize the top-3 scored videos. A “✓” sign indicates that the video label is consistent with the cluster label; otherwise, a “✗” sign is used. The two “mis-clustered” videos are on *parkour* (left) and *feeding animal* (right). Below each video, we show the top eight inferred tags sorted by the potential calculated from Eq. (2.8).

Chapter 3

Hierarchical Maximal-Margin Clustering

In this chapter, we present a hierarchical maximum-margin clustering method for unsupervised data analysis. Our method extends beyond flat maximum-margin clustering, and performs clustering recursively in a top-down manner. We propose an effective greedy splitting criteria for selecting which cluster to split next, and employ regularizers that enforce feature sharing/competition for capturing data semantics. Experimental results obtained on four standard image datasets show that our method outperforms flat and hierarchical clustering baselines, while forming clean and semantically meaningful cluster hierarchies.

3.1 Overview

Clustering is an important topic in machine learning that, after decades of research, remains a challenging and active topic of research. Clustering aims to group instances together based on their underlying similarity in an unsupervised manner. Clustering remains an active topic of research due to its widespread applicability in the areas of data analysis, visualization, computer vision, information retrieval, and natural language processing. Popular clustering methods include k-means clustering [42, 72] and spectral clustering [80].

Recent progress in maximum-margin methods has led to the development of maximum-margin clustering (MMC) techniques [131], which aim to learn both the separating hyperplanes that separate clusters of data, and the label assignments of instances to the clusters. MMC outperforms traditional clustering methods in many cases, largely due to the discriminative margin separation criterion imposed among clusters.

However, MMC also has limitations. First, MMC is not particularly efficient. While efficient MMC methods have been proposed [138, 139], even in such cases the time complexity is at least linear or quadratic with respect to the number of samples and clusters. This scalability issue is a significant problem when considering the scale of modern datasets. Second, MMC has difficulty

identifying clusters with small margins, which are particularly useful for fine-grained data. Consider clustering images of commercial vehicles. In such data the major source of dissimilarity among samples is the viewpoint and this is what MMC is likely to focus on. The variations in the make of the vehicle, which are semantically more meaningful, would result in more local fine-grained differences and may be ignored by MMC’s flat-clustering criterion.

Hierarchical clustering methods, which are typically based on a tree structure, have been extensively studied for their benefits over their flat clustering counterparts. These hierarchical clustering methods can discover hierarchical structures in data that better represent many real-world data distributions. Computationally, hierarchical clustering methods are also often more efficient, because one can reduce a single large clustering problem into a set of smaller subproblems to be recursively solved. Since within each sub-problem the data only needs to be clustered into a small number of clusters, and for lower levels of the hierarchy only a small subset of the data participates in each clustering step, this procedure tends to be a lot more efficient.

To leverage such benefits, we propose a hierarchical extension to MMC that recursively performs k-way clustering in a top-down manner. However, instead of naively performing MMC at each clustering step, we further leverage the observation from human-defined taxonomies that each grouping/splitting decision typically focuses on different features of the data.

Suppose, again, that we want to cluster different types of commercial vehicles. Assuming we can cluster the data hierarchically, it is sensible to assume that first we should cluster the data based on the vehicle type (*e.g.*, truck, SUV, sedan). Once we know which sub-group each instance belongs to, we may want to employ other criteria to separate them, *e.g.*, according to the price range or the make. We want to leverage a similar intuition to learn clusters that focus on maximizing the margin along different directions at different levels in the hierarchy. Here, directions are defined by subsets of features from the much larger feature vectors describing each instance. More specifically, we employ regularization that allows clusters to group and compete for the features at different levels. Such regularization has been made popular in semantic supervised learning in recent years [129, 47], but here we apply the idea in an unsupervised hierarchical clustering framework.

We test our hierarchical maximum-margin clustering (HMMC) method on several image datasets, and show that HMMC is able to outperform flat clustering methods like MMC. More significantly, it is able to discover clean and semantically meaningful cluster hierarchies, outperforming other hierarchical clustering alternatives.

Our contributions are threefold: (i) we present a novel hierarchical clustering algorithm based on maximum-margin clustering with an effective greedy splitting criterion for selecting which cluster to split next, (ii) we employ regularization that enforces feature sharing/competition to learn clusters that can focus on important features during clustering, and (iii) we empirically validate that our HMMC can learn semantically meaningful clusters without any human supervision.

The rest of this chapter is organized as follows. Section 3.2 reviews related work. Section 3.3 formulates HMMC, followed by the optimization algorithm presented in Section 3.4. We report experimental results in Section 3.5. Finally, Section 3.6 summarizes this chapter.

3.2 Related Work

Maximum-margin clustering: MMC was first proposed by Xu *et al.* [131]. It is a maximum-margin method for clustering, analogous to support vector machines (SVMs) for supervised learning problems, that learns both the maximum-margin hyperplane for each cluster and the clustering assignment of instances to clusters. Since this joint learning results in a non-convex formulation, unlike SVMs, it is often solved by a semidefinite relaxation [131, 115] or alternating optimization [138]. While most of the MMC methods focus on efficient optimization of the non-convex problems, the MMC formulation was also extended to handle the case of multi-cluster clustering problems [132, 139], and to include latent variables (as in Chapter 2).

Hierarchical clustering methods: Most hierarchical clustering methods employ either top-down clustering strategies that recursively split clusters into fine-grained clusters, or bottom-up clustering strategies that recursively group the smaller clusters into larger ones [77]. Our method is a top-down clustering method, and the canonical example of such a method is hierarchical k-means clustering, which performs k-means recursively in a top-down manner (*e.g.*, the bisecting k-means method [110]). Variations on this idea include hierarchical spectral clustering (*e.g.*, PDDP [6]) which performs the hierarchical clustering on the graph Laplacian of the similarity matrix, and model-based hierarchical clustering [114, 8, 38] which fits probabilistic models at each split. To the best of our knowledge, this is the first work using a maximum-margin approach for hierarchical clustering.

Sharing/competing for features: Regularization methods that promote certain structures in the parameter or feature spaces have been extensively studied in the context of regression, classification, and sparse coding. The group lasso [79] employs a mixed $\ell_{1,2}$ -norm to promote sparsity among groups of features, identifying the groups that are most important for the task. This has been applied to classification tasks like multi-task learning and multi-class classification, where it encourages the classifier(s) to share features across the tasks/classes. A generalization of the group lasso is the sparse group lasso [35], that further encourages sparsity within each individual model.

However, in some cases it makes more sense to have models fit to exclusive sets of features. The exclusive lasso [143] encourages two models to use different features, by minimizing the ℓ_2 -norm of their ℓ_1 -norms. This discourages different models from having non-zero values along the same feature dimensions, encouraging each model to use features that are exclusive to their tasks. Orthogonal transfer [129] focuses on such exclusiveness between parent and child models in a taxonomy, and enforces the exclusivity through “orthogonal regularization” where we minimize the inner product of the SVM weights for parent and child nodes. The tree of metrics approach [47] employs similar intuition, but learns Mahalanobis metrics instead of SVM weights, and focuses on selecting sparse and disjoint features. Tree-guided group lasso [55] employ both sharing and exclusive regularizations, to promote sharing between the labels that belong to the same parent, while also enforcing exclusive fitting between them, guided by a predefined taxonomy. These meth-

ods consider supervised learning scenarios, while our method utilizes the grouping and exclusive regularizers for unsupervised clustering.

3.3 Hierarchical Maximum-Margin Clustering

We propose a hierarchical clustering method based on the maximum-margin criterion. We aim to find groups of data points with a large separation between them, while forming a cluster hierarchy. The proposed method builds on the standard flat MMC clustering [131], but extends MMC in the following two aspects: (i) we introduce regularizers to encourage the different layers of the hierarchy to focus on the use of different feature subsets, and (ii) we build the hierarchy iteratively from coarse clusters to fine-grained clusters (rather than forming all clusters in one split) using a greedy top-down algorithm with a novel splitting criterion. We first introduce the HMMC formulation in this section, and then describe the optimization method in Section 3.4.

Suppose there are T non-leaf nodes $\{n_t\}_{t=1}^T$ in the learned hierarchy. We use \mathcal{D}_t to denote the data on n_t , and HMMC splits \mathcal{D}_t into K_t clusters by learning a linear model \mathbf{w}_{tk} for each cluster k . We collect the K_t cluster models in $\mathbf{w}_t = \{\mathbf{w}_{tk}\}_{k=1}^{K_t}$. We split the data \mathcal{D}_t on node n_t using the MMC idea – finding a clustering assignment such that the resultant margin between clusters is maximal over all possible assignments. By summing over all the non-leaf splits, our global HMMC objective is formulated as:

$$\begin{aligned} \min_{\substack{\mathbf{w}, \mathbf{y} \\ \xi \geq 0}} \quad & \sum_{t=1}^T \left(\alpha G(\mathbf{w}_t) + \beta E(\mathbf{w}_t) + \frac{1}{|\mathcal{D}_t| K_t} \sum_{\substack{\mathbf{x}_i \in \mathcal{D}_t \\ y \neq y_{ti}}} \xi_{tiy}^2 \right), \\ \text{s.t.} \quad & \mathbf{w}_{ty_{ti}}^\top \mathbf{x}_i - \mathbf{w}_{ty}^\top \mathbf{x}_i \geq 1 - \xi_{tiy}, \quad \forall t, \mathbf{x}_i \in \mathcal{D}_t, y \neq y_{ti} \\ & y_{ti} \in \{1, \dots, K_t\}, \quad \forall t, \mathbf{x}_i \in \mathcal{D}_t \\ & L_t \leq \sum_{\mathbf{x}_i \in \mathcal{D}_t} \Delta(y_{ti} = y) \leq U_t, \quad \forall t, y \in \{1, \dots, K_t\} \end{aligned} \quad (3.1)$$

where $\mathbf{w} = \{\mathbf{w}_t\}$ are the cluster model parameters, y_{ti} denotes the cluster label of an instance \mathbf{x}_i on node n_t , ξ 's are slack variables to allow margin violations, $G(\cdot)$ and $E(\cdot)$ are regularizers, and α and β are trade-off parameters. Our algorithm uses MMC for each data split, where we enforce the maximum-margin criterion by constraining the score of fitting \mathbf{x}_i to its assigned cluster to be sufficiently larger than to any other cluster, using the squared hinge loss (whose smoothness simplifies the optimization). The last constraint enforces the clusters to be balanced, to avoid degenerate solutions with empty clusters and infinite margins. Here $\Delta(\cdot)$ is an indicator function, while L_t and U_t are the lower and upper bounds controlling the size of the clusters. As suggested in Chapter 2, we set L_t and U_t to $0.9 \frac{|\mathcal{D}_t|}{K_t}$ and $1.1 \frac{|\mathcal{D}_t|}{K_t}$, respectively, to achieve roughly balanced clusters at each split. Note that HMMC jointly optimizes the model parameters \mathbf{w} and clustering assignments $\mathbf{y} = \{y_{ti}\}$ over all splits.

The two regularizers $G(\mathbf{w}_t)$ and $E(\mathbf{w}_t)$ promote learning of a semantically meaningful cluster hierarchy. These regularizers encourage splitting on a sparse group of features at each node, but encoding a preference towards using different features at different levels of the hierarchy. While the grouping and competition among features have proved useful for encoding semantic taxonomies in supervised learning problems [129, 47], we apply these ideas for discovering semantically meaningful cluster hierarchies in an entirely unsupervised setting.

Group sparsity: In the hierarchy, we would like different splits to focus on different subsets of features. Thus, in splitting a non-leaf node, we encourage the clustering process to only use a sparse set of relevant features. Considering that there are K_t cluster models at node n_t , we enforce group sparsity [79] over different feature dimensions so that the K_t models are using the same subset of features. Formally, we have the following regularizer on the split of n_t :

$$G(\mathbf{w}_t) = \frac{1}{PK_t} \sum_{p=1}^P \sqrt{\sum_{k=1}^{K_t} \mathbf{w}_{tk,p}^2}, \quad (3.2)$$

where P is the feature dimension, and $\mathbf{w}_{tk,p}$ is the p -th element in \mathbf{w}_{tk} . Mathematically, this term encodes a mixed $\ell_{1,2}$ -norm to enforce sparsity among the feature dimensions. Thus, if a feature is irrelevant, then it is zero-weighted in all the K_t cluster models.

Exclusive sparsity: We also want the cluster hierarchy to use different subsets of features in different layers, so that we consider different factors when traversing the hierarchy. In other words, a split is expected to explore features that are different from its ancestors and descendants, and thus the splits compete for features at different layers. We will denote a node n_t 's ancestors by \mathcal{A}_t , which formally is the set of nodes on the path from the root to n_t . With this notation the exclusive regularizer for node n_t is defined by [143]:

$$E(\mathbf{w}_t) = \frac{1}{K_t |\mathcal{A}_t| P} \sum_{k=1}^{K_t} \sum_{n_a \in \mathcal{A}_t} \sum_{p=1}^P |\mathbf{w}_{tk,p}| \cdot |\mathbf{w}_{ak_a,p}|, \quad (3.3)$$

where k_a indexes the child of n_a ($n_a \in \mathcal{A}_t$) on the path to n_t . Thus, \mathbf{w}_{ak_a} is the parameter vector for the ancestral cluster to which n_t belongs. Eq. (3.3) penalizes ‘‘cooperation’’ (using the same features) and encourages ‘‘competition’’ (using different features) between a cluster model \mathbf{w}_{tk} and each of its ancestor models $\{\mathbf{w}_{ak_a}\}_{n_a \in \mathcal{A}_t}$. The degree of competition is calculated as the element-wise multiplication of the absolute weight values. Intuitively, this means that there is no penalty if two models use different features, but using the same features results in a high penalty. Consequently, minimizing the exclusive sparsity as we split nodes will encourage nodes to use features different from those used by their ancestors and descendants. In [129, 117], it is shown that Eq. (3.3) becomes convex when combined with a sufficiently large ℓ_2 -regularizer.

Algorithm 1 HMMC: A greedy algorithm for building hierarchy

Input: n_1 and \mathcal{D} ▷ n_1 is the root node carrying all data in \mathcal{D}
Output: \mathcal{H} ▷ the cluster hierarchy including all non-leaf nodes

- 1: **Initialize:** $\mathcal{L} \leftarrow \{n_1\}$; ▷ the current set of leaf nodes
- 2: **while** the stopping criterion is not met **do**
- 3: **for** $n_t \in \mathcal{L}$ **do**
- 4: cluster the data on n_t ; ▷ details in Section 3.4.2
- 5: compute the splitting score $S(n_t)$; ▷ using Eq. (3.4)
- 6: $n_* \leftarrow \operatorname{argmax}_{n_t \in \mathcal{L}} S(n_t)$; ▷ greedily find the next split
- 7: $\mathcal{L} \leftarrow \mathcal{L} \setminus n_*$; $\mathcal{H} \leftarrow \mathcal{H} \cup n_*$; ▷ move n_* from \mathcal{L} to \mathcal{H}
- 8: **for** each cluster in n_* **do**
- 9: create a leaf node n_c carrying the data in that cluster;
- 10: link n_c as a child of n_* ;
- 11: $\mathcal{L} \leftarrow \mathcal{L} \cup n_c$; ▷ add n_c to the current set of leaf nodes

3.4 Optimization

The objective of Eq. (3.1) is non-convex due to the unknown hierarchical structure, and because we do not know the split on each node that jointly optimizes \mathbf{w} and \mathbf{y} . To solve the problem, we propose a greedy top-down algorithm to build the hierarchy (Section 3.4.1), and an alternating descent algorithm for splitting a node (Section 3.4.2).

3.4.1 Building the Hierarchy

We build the cluster hierarchy in a top-down manner, where the challenge is to iteratively find the next leaf to split. Algorithm 1 gives an overview of our greedy method. We start from the root node n_1 containing all the data. Note that n_1 starts as a leaf node since it has no children. Each iteration tries to split the data on each leaf node n_t (Step 4), and we define the splitting score (Step 5) as:

$$S(n_t) = \frac{\sum_{\mathbf{x}_i \in \mathcal{D}_t} \mathbf{w}_{ty_i}^\top \mathbf{x}_i}{G(\mathbf{w}_t) + E(\mathbf{w}_t)}. \quad (3.4)$$

The splitting score measures how well, and how easily, the data on node n_t can be clustered. The numerator of Eq. (3.4) summarizes the scores of fitting each instance to its assigned cluster. A high value in the numerator indicates compact clusters where the instances are well-fit by the assigned cluster models. The denominator of Eq. (3.4) is the regularization term indicating the complexity of the cluster models, where a small value implies a simple model. Thus, a higher splitting score means the node is a better candidate to be split.

The leaf node to split is chosen to greedily maximize the splitting score (Step 6). We fix the cluster models on this node, mark it as a non-leaf node, and move it to the hierarchy (Step 7). Moreover, since we are splitting this node, we generate its child nodes according to the clustering result and add the child nodes to the leaf node set for the next iteration (Steps 8 to 11). We iterate this process until the stopping criterion is satisfied, which could test whether (i) a given number of

leaf nodes are found, (ii) whether the sizes of all leaf nodes are sufficiently small, or (iii) whether the hierarchy reaches a height limit. To speed up this process, we cache the clustering result on each leaf node, so that we do not have to rerun the clustering once the leaf node is selected to grow the hierarchy.

3.4.2 Splitting A Node

The clustering on a given node n_t is formulated as:

$$\min_{\substack{\mathbf{w}_t, \mathbf{y}_t \\ \xi \geq 0}} \alpha G(\mathbf{w}_t) + \beta E(\mathbf{w}_t) + \frac{1}{|\mathcal{D}_t|K_t} \sum_{\mathbf{x}_i \in \mathcal{D}_t} \sum_{y \neq y_{ti}} \xi_{tiy}^2, \quad (3.5)$$

where we omit the constraints (from Eq. (3.1)) for brevity. Note that the cluster models of the ancestors of n_t have been fixed in the greedy top-down learning process. Thus, the exclusive regularizer $E(\mathbf{w}_t)$ becomes a weighted ℓ_1 -norm (sparsity-inducing) regularizer on \mathbf{w}_t , where the weight on each model parameter $w_{tk,p}$ is set based on the ancestor nodes to $\frac{\sum_{n_a \in \mathcal{A}_t} |w_{ak_a,p}|}{K_t |\mathcal{A}_t| P}$. Together with the group sparsity $G(\mathbf{w}_t)$, this yields a weighted sparse group lasso regularizer, generalizing the sparse group lasso regularizer of Friedman *et al.* [35].

Eq. (3.5) is still a non-convex problem due to the joint optimization over \mathbf{w}_t and \mathbf{y}_t . We use an alternating descent algorithm to reach a solution. In each iteration we fix the model parameters \mathbf{w}_t and optimize \mathbf{y}_t by solving a clustering assignment problem, and then we update \mathbf{w}_t while keeping \mathbf{y}_t fixed using a proximal quasi-Newton algorithm [65, 102]. The algorithm stops when the objective converges to a local optimum with respect to these steps.

Clustering assignment: With \mathbf{w}_t fixed, the problem in Eq. (3.5) turns out to be an assignment problem, which minimizes the total cost for labeling all instances while maintaining balanced clusters:

$$\begin{aligned} \min_{\mathbf{y}_t} \quad & \sum_{\mathbf{x}_i \in \mathcal{D}_t} \Delta(y_{ti} = y) \cdot \overbrace{\sum_{y' \neq y} [1 - \mathbf{w}_{ty'}^\top \mathbf{x}_i + \mathbf{w}_{ty'}^\top \mathbf{x}_i]_+^2}^{C_{tiy}}, \\ \text{s.t.} \quad & y_{ti} \in \{1, \dots, K_t\}, \quad \forall \mathbf{x}_i \in \mathcal{D}_t \\ & L_t \leq \sum_{\mathbf{x}_i \in \mathcal{D}_t} \Delta(y_{ti} = y) \leq U_t, \quad \forall y \in \{1, \dots, K_t\} \end{aligned} \quad (3.6)$$

where C_{tiy} is the cost for assigning an instance \mathbf{x}_i into a cluster y . Following Chapter 2, we could solve Eq. (3.6) by constructing an integer linear programming (ILP) problem with $O(|\mathcal{D}_t| \cdot K_t)$ variables and $O(|\mathcal{D}_t| + K_t)$ constraints. However, this ILP is time-consuming since in the worst case the complexity of existing ILP solvers is exponential in the number of variables. To efficiently solve this problem, we formulate it as a minimum cost flow (MCF) problem.

We re-write the clustering assignment as the problem of sending an MCF through an appropriately designed network, illustrated in Figure 3.1. The flow capacity of an edge from the starting node s to an instance node \mathbf{x}_i is set to 1 since we are assigning every instance into a cluster. This

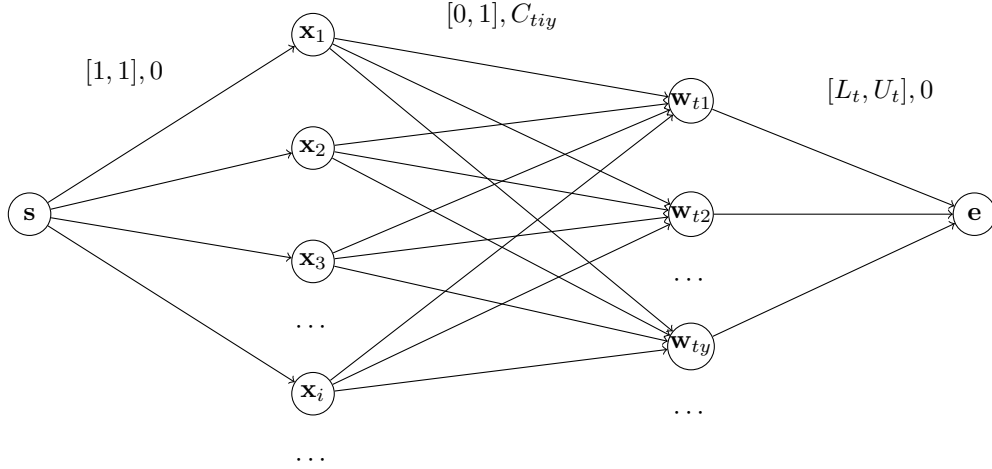


Figure 3.1: A sample MCF network. The edge settings are formatted as: “[lower capacity, upper capacity], cost”. See text for details.

one unit of flow is sent from \mathbf{x}_i to a cluster node \mathbf{w}_{ty} , to which the instance is assigned, at cost C_{tiy} . Finally, each cluster node sends its receiving flows to the end node \mathbf{e} , where we limit the flow capacity in the range $[L_t, U_t]$ to take the cluster balance constraints into account. It can be shown that clustering the $|\mathcal{D}_t|$ instances into K_t clusters (under the cluster balance constraints) is equivalent to sending $|\mathcal{D}_t|$ units of flow from \mathbf{s} to \mathbf{e} , and the optimal network flow corresponds to the minimum total cost of Eq. (3.6). To find this optimal flow, we apply the capacity scaling algorithm [25] implemented in the LEMON library [22], which is an efficient dual solution method running in $O(|\mathcal{D}_t| \cdot K_t \cdot \log(|\mathcal{D}_t| + K_t) \cdot \log(U_t \cdot |\mathcal{D}_t| \cdot K_t))$ complexity. In practice, our MCF solver speeds up the ILP solver in Chapter 2 by 10 to 100 times.

Updating \mathbf{w}_t : With fixed \mathbf{y}_t , we solve for \mathbf{w}_t (a convex problem) using a proximal quasi-Newton method [65, 102]. This method is designed to efficiently minimize smooth losses with non-smooth but simple regularizers, and on each iteration it computes a new estimate \mathbf{w}_t by solving:

$$\min_{\mathbf{w}_t} \alpha G(\mathbf{w}_t) + \beta E(\mathbf{w}_t) + H(\mathbf{w}_t^{old}) + H'(\mathbf{w}_t^{old})^\top (\mathbf{w}_t - \mathbf{w}_t^{old}) + \frac{1}{2s} \|\mathbf{w}_t - \mathbf{w}_t^{old}\|_B^2, \quad (3.7)$$

where s is a step-size set using a backtracking line-search, $H(\mathbf{w}_t^{old})$ is the squared hinge-loss (*i.e.*, the last term of Eq. (3.5) after using the constraints to eliminate the slack variables) estimated with \mathbf{w}_t^{old} from the fixed \mathbf{y}_t , $H'(\mathbf{w}_t^{old})$ is the derivative of $H(\mathbf{w}_t^{old})$ w.r.t. \mathbf{w}_t^{old} , and $\|\mathbf{z}\|_B^2 = \mathbf{z}^\top B \mathbf{z}$ is a divergence formed using the L-BFGS matrix B [7, 81].

A spectral proximal-gradient method is used to compute an approximate minimizer of this objective. This algorithm requires the proximal operator. For our weighted sparse group lasso regularizer, we can show that solving this minimizing problem involves a two-step procedure. First, we incorporate the weighted ℓ_1 -norm penalty by applying the soft-threshold operator $\mathbf{w}_{tk,p} =$

$\frac{\mathbf{w}_{tk,p}}{|\mathbf{w}_{tk,p}|} [|\mathbf{w}_{tk,p}| - s\beta\lambda_E]_+$ to each model parameter individually, where $\lambda_E = \frac{\sum_{n_a \in \mathcal{A}_t} |\mathbf{w}_{ak_a,p}|}{K_t |\mathcal{A}_t| P}$ is the weights coming from the ancestor models in $E(\mathbf{w}_t)$. This operator returns 0 if $\mathbf{w}_{tk,p} = 0$. Second, we incorporate the group sparsity using the group-wise soft-threshold operator $\mathbf{w}_{t:,p} = \frac{\mathbf{w}_{t:,p}}{\|\mathbf{w}_{t:,p}\|_2} [\|\mathbf{w}_{t:,p}\|_2 - s\alpha\lambda_G]_+$, where $\mathbf{w}_{t:,p} = [\mathbf{w}_{t1,p}, \dots, \mathbf{w}_{tK_t,p}]^\top$ is the grouping of K_t cluster models on a feature dimension p , and $\lambda_G = \frac{1}{PK_t}$ is the normalization term from $G(\mathbf{w}_t)$. Note that this operator returns $\mathbf{0}$ if $\mathbf{w}_{t:,p} = \mathbf{0}$.

Convergence analysis: We now show that this alternating descent algorithm converges to a local optimum. The optimization consists of two alternating steps: updating the discrete \mathbf{y}_t and the continuous \mathbf{w}_t . In the \mathbf{w}_t update, we fix the clustering \mathbf{y}_t and use a method that is guaranteed to find a global optimum [65, 102]. The \mathbf{y}_t update (with \mathbf{w}_t fixed) also seeks the global optimum by using MCF to solve the clustering assignment problem. Thus, the procedure guarantees convergence to a local minimum with respect to updating \mathbf{w}_t and \mathbf{y}_t .

3.5 Experiments

Datasets: We evaluate the performance of HMMC on four datasets from two public image collections: Animal With Attributes (AWA) [61] and ImageNet [19]. Both collections have natural hierarchies consisting of fine-grained image classes that can be grouped into more general classes.

AWA contains 30,475 images from 50 animal classes (*e.g.*, *bat* and *deer*). We use two datasets following the practice of [47]. The first one, AWA-ATTR, has 85 features consisting of the outputs of 85 linear SVMs trained to predict the presence/absence of the 85 nameable properties annotated by [61], like *red* and *furry*. The second dataset, AWA-PCA, uses the provided features (SIFT, rgSIFT, PHOG, SURF, LSS, RGB) after being concatenated, normalized, and PCA-reduced to 100 dimensions. The ground-truth hierarchy of AWA is shown in Figure 3.2(a).

We use two datasets collected from ImageNet: VEHICLE contains 20 vehicle classes (*e.g.*, *cab* and *canoe*) and 26,624 images [47], and IMAGENET consists of 28,957 images spanning 20 non-animal, non-vehicle classes (*e.g.*, *lamp* and *drum*) [48]. The raw image features are the provided bag-of-words histograms obtained by SIFT [19, 18]. We also project them down to 100 dimensions with PCA. The semantic hierarchies of VEHICLE and IMAGENET are given in Figure 3.2(b) and Figure 3.2(c), respectively.

Baselines: We compare HMMC with four sets of baselines. The first set is the flat clustering methods k-means (KM) [42, 72], spectral clustering (SC) [80], and an MMC approach implemented in Chapter 2.

The second set is hierarchical bottom-up clustering (HBUC). We have tested a variety of methods including Single-Link (SL), Average-Link (AL) and Complete-Link (CL) [77]. The pairwise dissimilarity between two images is measured by Euclidean distance.

The third set is hierarchical top-down clustering methods (HTDC). We derive variants of hierarchical k-means (HKM) and hierarchical spectral clustering (HSC) directly from our HMMC approach. HKM and HSC apply the same greedy top-down approach as HMMC, but split a given

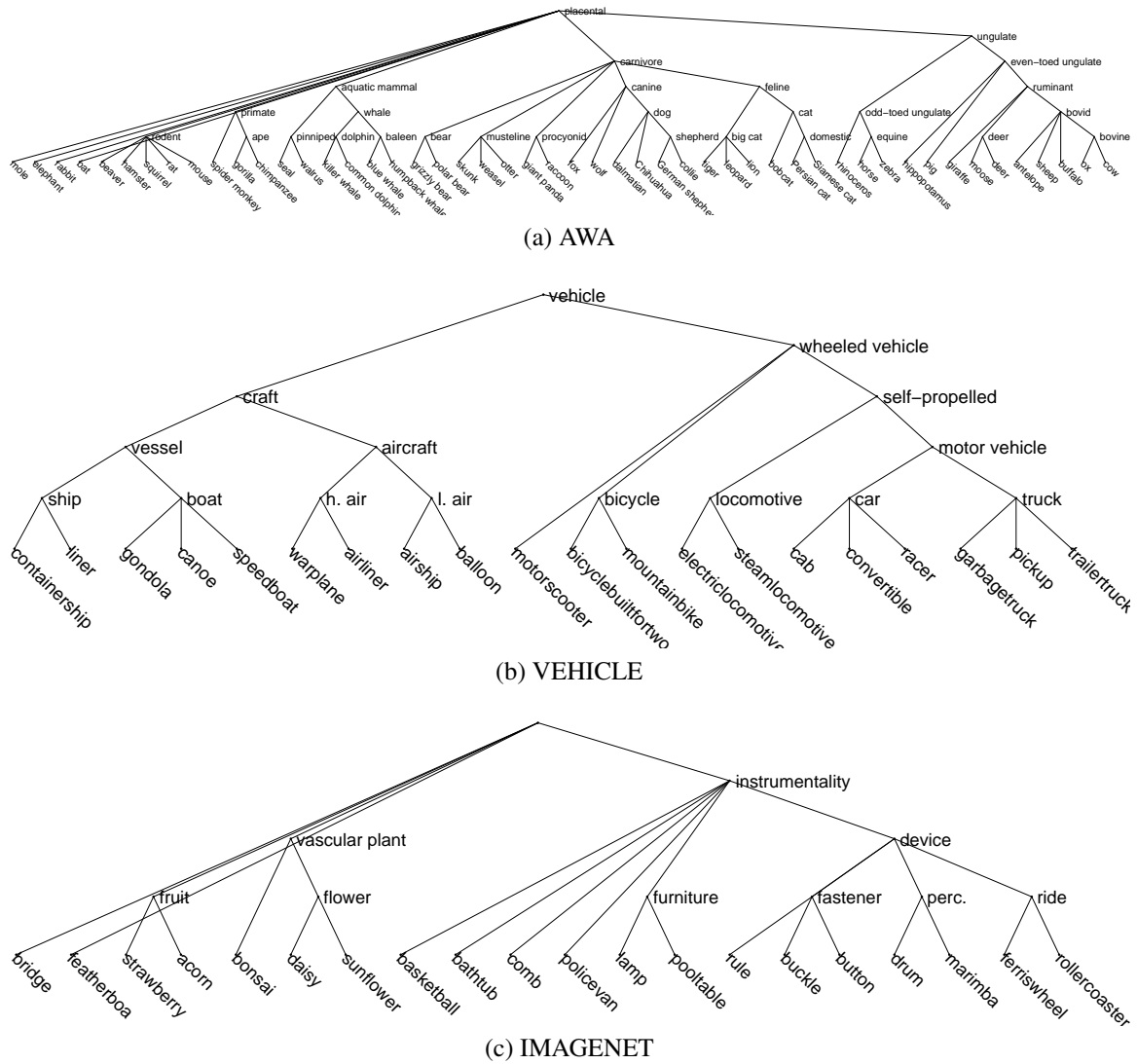


Figure 3.2: The semantic hierarchies of AWA, VEHICLE and IMAGENET. Note that AWA has 50 fine-grained animal classes, VEHICLE has 20 vehicle classes, and IMAGENET has 20 non-animal, non-vehicle classes.

node using k-means and spectral clustering, respectively. Similar to HMMC, HKM and HSC first try splitting all the current leaf nodes, and then greedily grow the leaf with the best splitting. The splitting score on a leaf node is defined as the average within-cluster distance – minimizing this gives the most compact clusters. We also considered two other baselines, HKM-D and HSC-D. Instead of growing the leaf with the most compact clusters, HKM-D and HSC-D grow the leaf with the most scattered data, which is defined as the total distance of all instances to their center.

The fourth set of baselines are variants of HMMC. We change the regularization to derive HMMC-G (group sparsity only), HMMC-E (exclusive sparsity only), HMMC-1 (basic ℓ_1 -norm), and HMMC-2 (squared ℓ_2 -norm).

Parameters: For a fair comparison of all the hierarchical top-down clustering methods, we apply the same stopping criterion: we test if the number of leaf nodes exceeds a fixed limit F . Empirically, we set F as 1, 1.5 and 2 times the number of ground-truth classes in each dataset. The number of splits on each node also has a great impact on the learned hierarchy. To compare different hierarchical clustering methods, we simply use K -nary branching for all splits in all hierarchies. We experiment with K set as 2, 3, 4 and 5, respectively. With a particular setting of F and K , we can fairly compare different hierarchical clustering methods since they perform the same number of splits and obtain the same number of leaf nodes. We use the same solver for learning HMMC and its variants, and report the best performance with both α and β selected from the range $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^0\}$.

For the HBUC baselines, we apply the same F parameter as above. However, all the HBUC methods use binary branching and there is no result for K larger than 2.

For the flat clustering methods (*i.e.*, KM, SC and MMC), we set the number of clusters to F to fairly compare performance with hierarchical methods. For SC, we use a 5-nearest neighborhood graph and set the width of the Gaussian similarity function as the average distance over all the 5-nearest neighbors. This also applies in HSC and HSC-D which use SC for splitting a node.

Performance measures: We evaluate all the methods by three performance measures. The first two are semantic measures focusing on how well the learned hierarchy captures the semantics in the ground-truth hierarchy. The motivation is that two semantically similar images should be grouped in the same or nearby clusters in the learned hierarchy, and two semantically dissimilar images should be split into clusters that are far away from each other.

For each pair of images, we compute their semantic similarity from the ground-truth hierarchy, using the following two metrics. The first “shortest path” metric [41] finds the shortest path linking the two image classes in the ground-truth hierarchy, normalizes the path distance by the maximum distance, and subtracts the distance from 1 as the semantic similarity. The second “path sharing” metric [33] counts the number of nodes shared by the parent branches of the two image classes, normalized by the length of the longest of the two branches. Note that we can similarly define the shortest path similarity and the path sharing similarity using the learned hierarchy, for any pair of images, by checking the leaf node(s) where the two images are clustered. For flat clustering with no hierarchy, we simply set the similarity as 1 if two images are from the same cluster, and 0 otherwise.

To measure the goodness of the learned hierarchy in capturing semantics, we compute the mean squared error of the learned similarity and the ground-truth semantic similarity over all pairs of images, and subtract the mean squared error from 1 as our semantic measure. Note that we have two semantic measures, the shortest path similarity (SP) and the path sharing similarity (PS). The higher the values, the better the performance.

Moreover, we also report the Rand Index (RI) [95], which evaluates the percentage of true positives within clusters and true negatives between clusters. Note that RI is a commonly-used measure for flat clustering. For hierarchical clustering methods, we simply ignore the hierarchy and evaluate RI on the leaf node clustering (allowing direct comparisons with flat clustering methods).

3.5.1 Results

Comparing flat and hierarchical methods: We report the clustering results with F equal to the number of ground-truth classes and $K = 2$ (binary splitting) in Table 3.1. Note that we observe similar results with other settings of F and K so we omit the numbers here. Table 3.1 shows that HMMC achieves the best performance on AWA-PCA, VEHICLE and IMAGENET, and competitive results on AWA-ATTR. Specifically, HMMC improves over the second best by 0.2% on AWA-ATTR, 2% on AWA-PCA, 6% on VEHICLE and 4% on IMAGENET, respectively, in terms of the semantic measure SP. This verifies that HMMC better captures the semantics in the clustered hierarchies.

Moreover, HMMC outperforms other HTDC baselines in most cases, showing the effectiveness of our greedy top-down algorithm for hierarchy building and our alternating descent algorithm for splitting data on a given node. Note that the HBUC baselines tend to perform worse since they typically produced extremely unbalanced clusters at the top levels (*e.g.*, a child contains only one sample). They also did not lead to semantically meaningful hierarchies.

Comparing the variants of HMMC: Table 3.1 shows that HMMC gets slightly better performance over the four variants of HMMC. This is reasonable since HMMC produces sparse models that may better capture semantics. We also compare the model sparsity (*i.e.*, the percentage of zeros in the learned models) in Figure 3.3. Here we omit HMMC-2 since the model is always non-sparse. For a fair comparison, we fix the trade-off parameters to 1 in all models. Note that by combining the grouping and exclusive regularizers, HMMC is sparser than HMMC-G and HMMC-E. HMMC-1 sometimes has slightly better sparsity than HMMC, but the performance is limited as it does not explicitly model group and exclusive sparsity to capture data semantics.

Runtime comparison: Table 3.1 also reports the runtime results. Our implementation of HMMC is between 1.4 to 8 times faster than MMC, showing the efficiency of the hierarchical method. Note that HMMC is more expensive than other hierarchical and flat methods. This is reasonable since HMMC needs to solve a more expensive optimization problem during clustering.

Using different F and K : We also vary the parameters F (*i.e.*, the number of leaf nodes) and K (*i.e.*, the number of splits), and plot the SP performance in Figure 3.4 and Figure 3.5, respectively. Here we have omitted the poor results of hierarchical bottom-up methods for better visualizations.

Table 3.1: Clustering performance on the four datasets. SP, PS and RI are reported in percentage, and the boldfaced numbers achieve the best performance among flat and hierarchical methods (excluding HMMC variants). The runtime (in seconds) is measured on a machine with Intel Xeon 2.8GHz CPU and 16GB memory.

		AWA-ATTR				AWA-PCA			
Methods		SP	PS	RI	runtime	SP	PS	RI	runtime
FLAT	KM	77.95	92.83	96.04	5.2	77.48	91.34	94.50	7.4
	SC	77.90	92.54	95.71	209.2	77.16	88.72	91.21	172.2
	MMC	77.08	83.67	83.71	15957.7	77.32	90.59	93.54	6077.0
HBUC	SL	63.97	16.24	2.65	88.3	64.00	16.24	2.69	80.9
	AL	74.55	38.99	32.84	72.1	64.37	17.00	3.94	80.9
	CL	92.60	87.54	93.33	81.8	68.14	22.63	34.27	47.6
HTDC	HKM	71.95	40.46	30.02	1.6	85.00	76.86	79.77	3.2
	HSC	81.59	69.84	67.43	247.0	79.47	47.69	57.25	873.2
	HKM-D	92.59	91.01	95.97	5.8	91.43	88.24	95.02	2.4
	HSC-D	94.18	90.38	95.98	293.4	79.94	48.02	57.97	873.4
	HMMC	94.40	91.03	95.96	1986.9	93.69	89.66	95.65	1550.1
VARIANT	HMMC-G	94.36	90.83	95.87	1389.6	93.77	89.59	95.56	1254.2
	HMMC-E	93.81	90.74	95.45	788.2	93.11	89.39	94.79	1408.8
	HMMC-1	87.77	77.49	77.84	558.1	92.01	89.69	95.49	769.9
	HMMC-2	92.70	90.92	95.99	893.2	93.65	89.47	95.25	1449.1

		VEHICLE				IMAGENET			
Methods		SP	PS	RI	runtime	SP	PS	RI	runtime
FLAT	KM	75.44	76.76	78.08	2.9	79.66	82.03	87.14	4.0
	SC	74.15	74.00	74.12	112.0	69.39	67.79	61.25	137.3
	MMC	78.03	84.23	88.49	1366.6	79.98	82.74	89.24	2634.3
HBUC	SL	58.21	29.12	5.30	61.2	45.85	32.97	5.24	84.5
	AL	58.24	29.17	5.45	47.8	45.87	33.00	5.29	49.0
	CL	58.30	29.26	5.58	54.2	46.46	33.62	6.59	69.3
HTDC	HKM	77.68	65.75	59.75	2.2	82.85	80.93	84.67	1.9
	HSC	68.59	45.84	36.62	745.4	64.76	52.69	53.64	913.6
	HKM-D	84.89	74.77	85.37	1.9	81.42	80.87	86.60	3.2
	HSC-D	69.29	46.07	37.11	316.9	48.19	37.01	11.00	892.9
	HMMC	90.48	85.08	90.16	994.3	86.94	84.63	90.59	1411.6
VARIANT	HMMC-G	90.40	85.03	90.10	883.6	86.69	84.33	90.56	1016.4
	HMMC-E	87.82	83.00	86.26	616.3	85.77	83.98	89.06	1658.4
	HMMC-1	89.94	84.72	89.54	486.9	86.81	84.52	90.52	690.1
	HMMC-2	90.05	84.71	89.68	541.3	86.13	84.08	89.94	1158.3

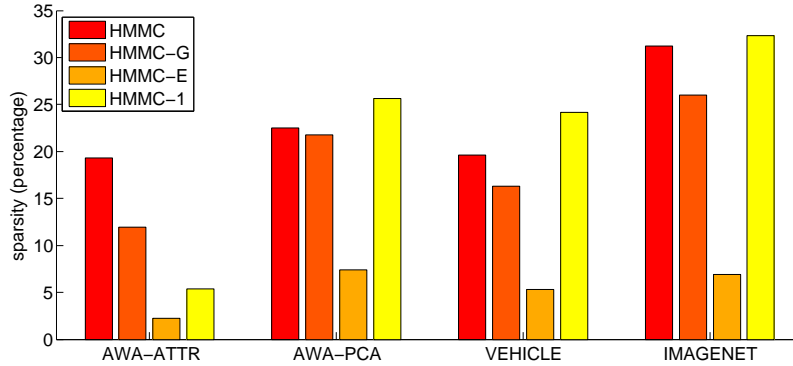


Figure 3.3: HMMC model sparsity. See text for details.

HMMC consistently outperforms the other baselines on AWA-PCA, VEHICLE and IMAGENET, and is comparable with HKM-D and HSC-D on AWA-ATTR. Note that the performance of HMMC is stable with regard to the different settings of F and K .

Visualizations: Figure 3.6 visualizes the learned hierarchy on AWA-ATTR. Our model captures semantically meaningful attributes in building the hierarchy – note how the attribute *quadrappedal* is used to separate *whales* and *polar bears*, and how *longneck* is used to divide *rhinos* and *giraffes*.

3.6 Summary

We have presented a hierarchical clustering method for unsupervised construction of semantic hierarchies. We develop a greedy top-down splitting criterion, and use the grouping and exclusive regularizers for building semantically meaningful hierarchies from unsupervised data. Our method makes use of maximum-margin learning, and we propose effective algorithms to solve the resultant non-convex objective. We test our method on four standard image datasets, showing the efficacy of our method in clustering, and the ability to capture semantics via the hierarchies. As future directions, it is interesting to leverage the tree structure building in a unified optimization framework.

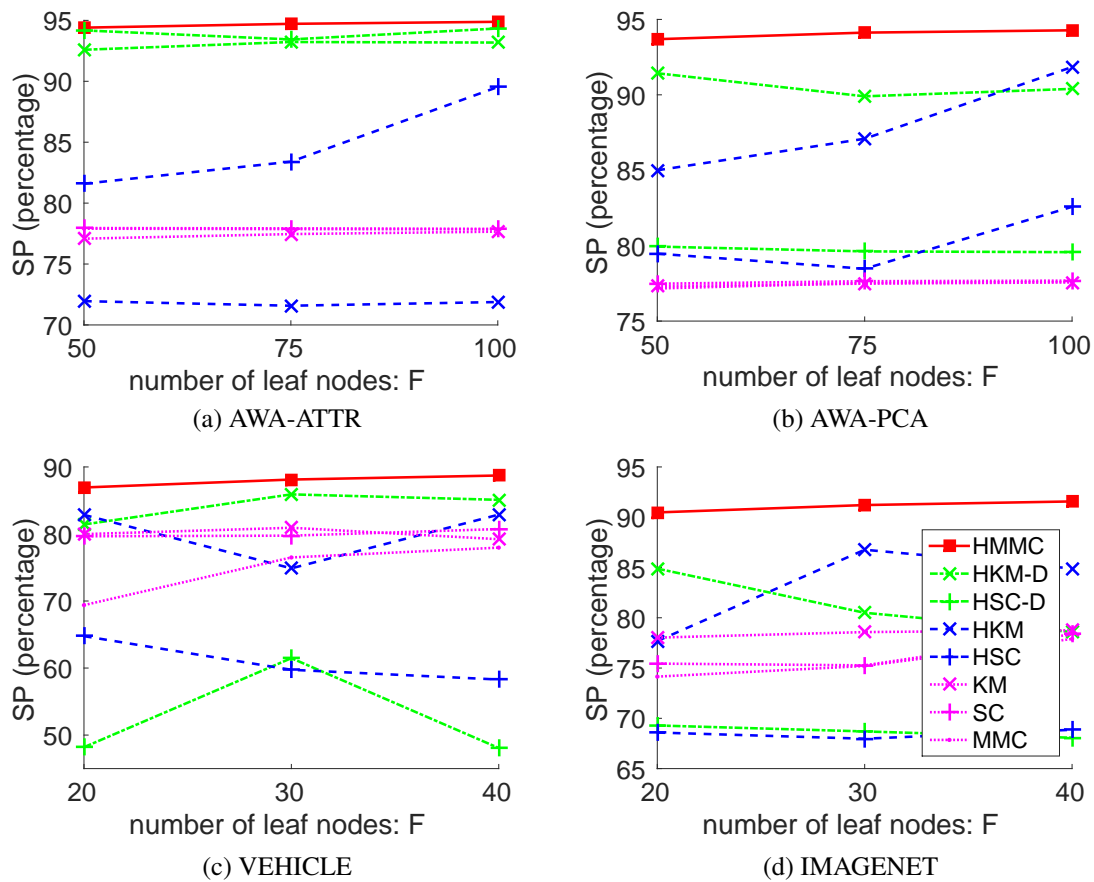


Figure 3.4: SP result w.r.t. different settings of F . Here we fix $K = 2$ on all the four datasets.

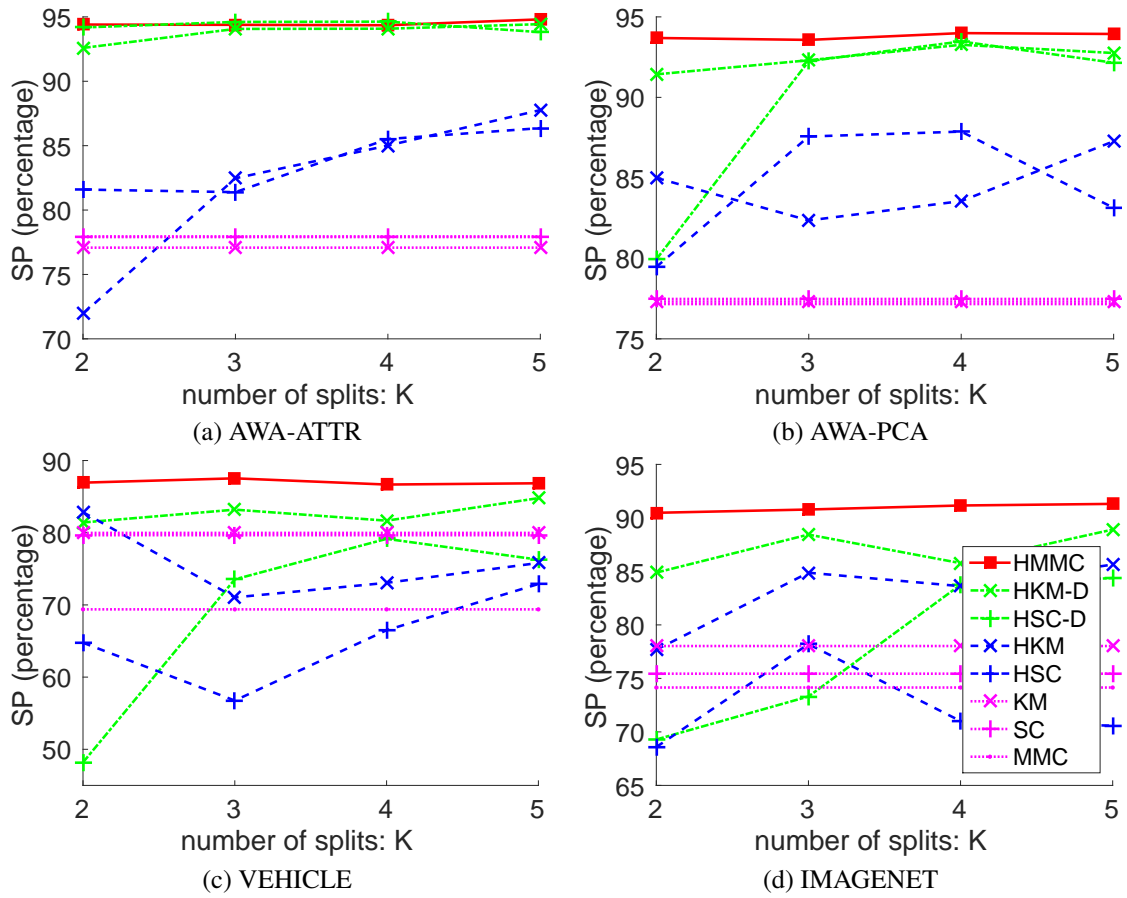


Figure 3.5: SP result w.r.t. different settings of K . Here we fix F as the number of ground-truth classes in each dataset.

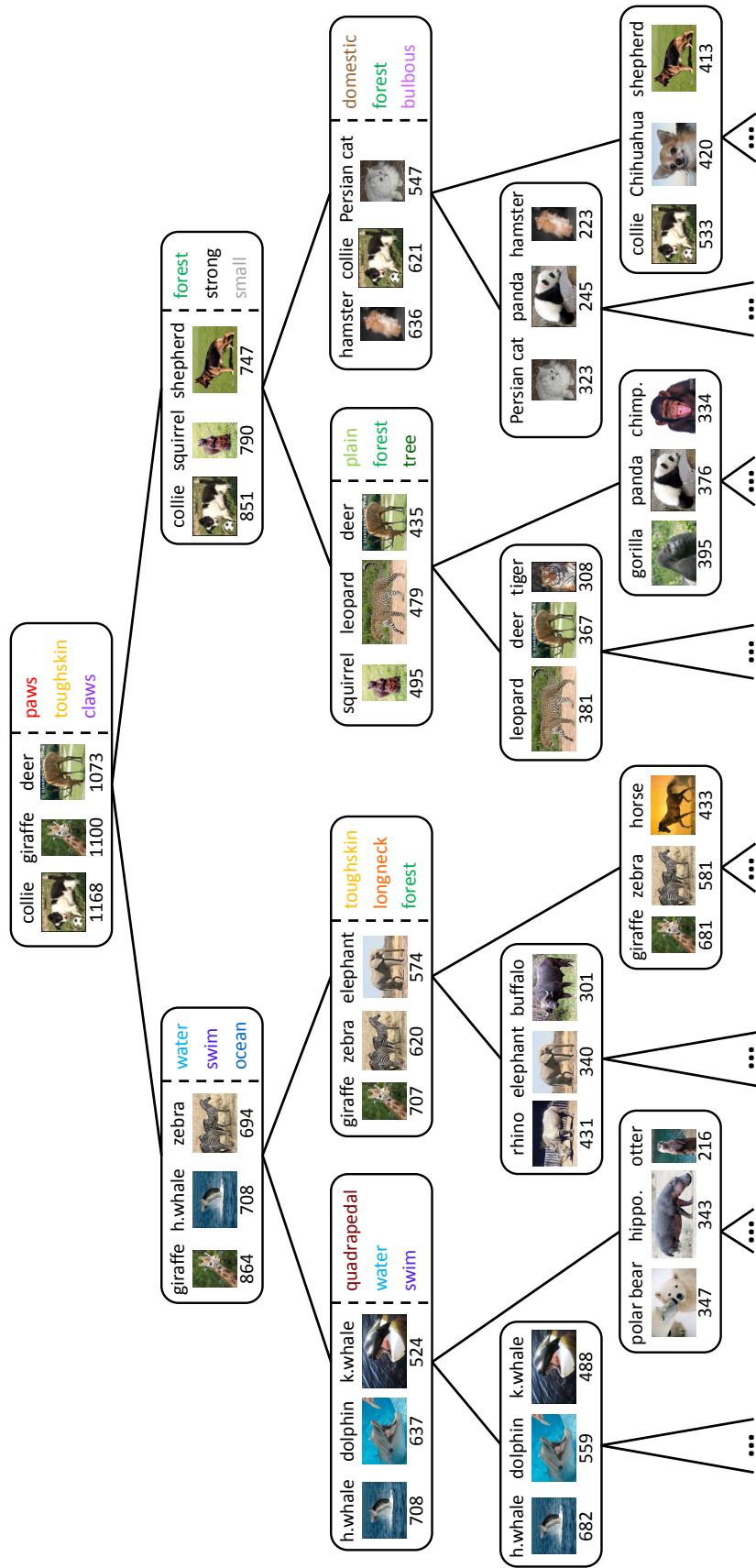


Figure 3.6: The learned hierarchy on AWA-ATTR with binary branching. Here we show the results on the first four layers. For each node, we visualize three majority image classes (with the number of images from each class listed below the sample image), and the three most discriminative attributes (ranked by the magnitude of the regularization on the corresponding feature dimension).

Chapter 4

Scene Recognition by Semantic Structure Discovery

In this chapter, we present our recognition method that leverages semantic scene structure discovery. Specifically, we conduct image recognition by learning a class-to-image distance function that matches objects. The set of objects in training images for an image class are treated as a structured collage. When presented with a test image, the best matching between this collage of training image objects and those in the test image is found. We validate the efficacy of the proposed model on the PASCAL 07 and SUN 09 datasets, showing that our structured model is effective for both object and scene recognition tasks. State-of-the-art recognition results are obtained, and qualitative results demonstrate that objects can be accurately localized and matched.

4.1 Overview

We present a method for image recognition that matches sets of objects. We aim to recognize an input image into classes, such as those containing a specific object (*e.g.*, PASCAL VOC [27, 28]) or coming from a certain scene (*e.g.*, SUN [128, 13, 127]). Our representation focuses on the set of objects found in an image class. An image class is represented using the set of objects contained in its image instances. We formulate a class-to-image distance for matching to an unseen image that looks for a set of similar objects in similar spatial arrangements to those found in a set of training images. The distance between this collage of objects and a test image is used to recognize the test image.

Image recognition is a well-studied problem in computer vision. An important question is choosing an appropriate representation for recognition. Standard approaches in the vision literature span a gamut of potential answers for this representation question. Purely statistical measures based on local features are common, *e.g.*, Lazebnik *et al.* [63]. Direct exemplar matching methods are also well-studied, *e.g.*, Berg *et al.* [3]. Detailed reasoning about object segmentation can also assist in image recognition [9]. Higher-level semantic reasoning about object context is another

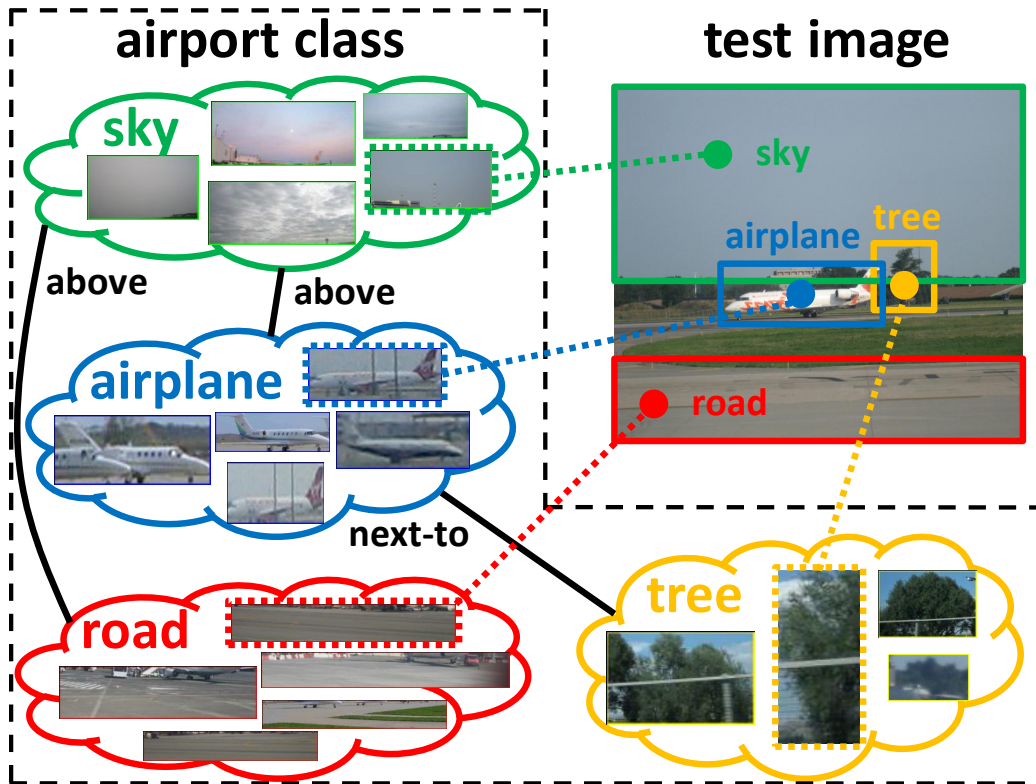


Figure 4.1: An example showing the object matchings between the *airport* class and a test image. There are four major object categories in the training *airport* images: *sky*, *airplane*, *road* and *tree*. We match the dashed objects from the training side to the objects in the test image, from which the class-to-image distance is calculated. Spatial relations, e.g., *sky-above-airplane*, *sky-above-road*, and *tree-next.to-airplane*, are also considered in measuring the distance.

important cue for image recognition, e.g., Li *et al.* [67]. The focus of this work is on object-level representations, though a solution to image recognition likely requires integration of all these sources of information.

In this chapter we develop a method that matches the object structure present in an image. We learn a distance from an image class to a given image that examines a higher-level semantic representation using objects. Figure 4.1 shows an example of the object matching. We are inspired by two recent lines of work – Object Bank [68], which takes a statistical view of object presence, and exemplar SVM [75] which considers matching individual exemplar objects. The Object Bank work of Li *et al.* [68] shows that a large bank of object detectors is an effective feature for image recognition – building a feature vector that captures the statistics of object detector responses. On the other hand, Malisiewicz and Efros [75] advocate for an exemplar matching approach – each image is its own island of uniqueness. Our work bridges these two approaches, leveraging the strength of many objects as a representation for images, but using a matching framework that considers structured collages of objects across an entire training class.

Our main contribution is the development of this image recognition method. We present a novel latent variable distance function learning framework that considers matchings of objects between a test image and a set of training images from one class. We develop efficient representations for the relationships between objects in this latent variable framework. We show empirically that this method is effective, and that reasoning about objects and their relations in images can lead to high quality recognition performance.

The rest of this chapter is organized as follows. We first review related work in Section 4.2. Then we describe our distance model in Section 4.3. The inference and learning algorithms are described in Sections 4.4 and 4.5, respectively. Later we report experimental results in Section 4.6. Finally, Section 4.7 summarizes this chapter.

4.2 Related Work

Object-level representations: Image understanding with object-level representations is common in computer vision research. We divide the literature into three categories. First, object-level representations supply rich information to assist detection and recognition. Malisiewicz and Efros [75] learn per-exemplar distance functions for data association based object detection. Li *et al.* [68] tackle scene recognition by representing an image as Object Bank – a feature vector that captures the statistics of object detectors. Second, object-level representations can be combined with other information sources. Wang and Mori [123] model object-tag correspondences in a latent variable framework. Wang and Forsyth [122] jointly learn object categories and visual attributes in a multiple instance learning framework. Third, groups of objects can provide useful contextual information. Rabinovich *et al.* [94] exploit contextual relevance of objects by modeling object co-occurrences. Lee and Grauman [66] encode the layout of object-level patterns by object-graph descriptors. Li *et al.* [67] model groups of objects as the basic elements for scene understanding. Lan *et al.* [62] retrieve images for structured object queries, and show that contextually-related objects are helpful even if they are not present in the given queries.

Distance function learning: There has been much work in recent years learning distance functions for image recognition. An early representative study by Frome *et al.* [36, 37] builds image-to-image distance on top of local patch-based distances, where each patch is localized by a geometric blur descriptor. Boiman *et al.* [5] compute nearest-neighbor based image-to-class distance based on local SIFT descriptors [73]. Wang *et al.* [125] also measure image-to-class distance by learning Mahalanobis distance metrics. Recent work by Wang *et al.* [124] regularizes class-to-image distance via ℓ_1 -norm. Wang *et al.* [119] define a class-to-bag distance for multiple instance learning. Our method also learns class-to-image distance, but the key difference is that we focus on object-level representations and explicitly reason about objects and their relations in images. In contrast, existing methods always operate in the space of local descriptor features.

4.3 The Object Matching Based Distance Model

Our goal is to learn a class-to-image distance function that jointly captures object matchings, object pairwise interactions, as well as global image appearance. We start with an example (Figure 4.1) that illustrates calculating the class-to-image distance from the *airport* class to a test image. The airport class is represented as a collage of object sets (*i.e.*, *sky*, *airplane*, *road* and *tree*) from training images, arranged in certain spatial layout, such as *sky-above-airplane*. In essence, our distance model matches to a test image with a set of similar objects in similar spatial arrangements from training images.

Our structured model consists of three components: the unary object distance, the pairwise object distance, and the global image appearance distance. The unary object distance measures the object-level distance from an image class to a test image. In our example, we match one object from each of the four object sets (*sky*, *airplane*, *road* and *tree*) to the test image. We calculate the distance between the matched pair of objects. The unary object distance is a summation over the four distances calculated from the four object matchings. The pairwise object distance measures the distance of spatial arrangements of objects from an image class to a test image. In our example, the matched objects in the test image meet the three popular spatial relations in the training airport images. Thus, we further pull the test image close to the airport scene. Finally, our distance model takes the global image features into account and calculates the global image appearance distance accordingly.

4.3.1 Model Formulation

We first introduce the notations used in this chapter before defining our distance model. We assume the ground-truth object bounding boxes are available in the training images. Note that this assumption reasonable because object annotation is becoming more and more prevalent with the help of online annotation tools such as LabelMe [99] and Amazon Mechanical Turk¹. Our two experimental datasets, PASCAL 07 [27] and SUN 09 [13], are both fully annotated.

For an image class C , we gather together all the objects in the training images belonging to this class to make up the object sets $\mathcal{O} = \{\mathcal{O}_i\}_{i \in \mathcal{V}}$, where \mathcal{V} denotes all the object categories in \mathcal{O} , and \mathcal{O}_i is the set of objects annotated with category $i \in \mathcal{V}$. We use \mathcal{O}_i^u to represent the u -th object in \mathcal{O}_i . Given an image \mathbf{x} , our model is a distance function $D_\theta(C, \mathbf{x})$ (here θ are the parameters of this function) that measures the class-to-image distance from C to \mathbf{x} based on object matchings. Ideally, $D_\theta(C, \mathbf{x})$ will have a small value if the image \mathbf{x} belongs to the class C , and a large value if \mathbf{x} comes from a class other than C .

There are two major challenges in defining $D_\theta(C, \mathbf{x})$. First, even though the ground-truth object bounding boxes are readily available in the training images, we do not have annotated objects on the test image set. To resolve this problem, we assume \mathbf{x} is associated with a set of “hypothesized” objects. We model the location/scale configurations of the hypothesized objects as latent variables

¹Available online at www.mturk.com/mturk/.

and infer them implicitly in our model. The latent variables are denoted as $\mathcal{H} = \{\mathcal{H}_i\}_{i \in \mathcal{V}}$, where \mathcal{H}_i is the set of hypothesized object configurations in category i . We use \mathcal{H}_i^v to denote the v -th configuration in \mathcal{H}_i and the corresponding hypothesized object interchangeably. Note that \mathcal{H} is normally smaller than \mathcal{O} in size because \mathcal{O} gathers all the objects in class- C images and \mathcal{H} only includes the objects in the image \mathbf{x} .

A second challenge lies in finding the optimal object matchings from \mathcal{O} to \mathcal{H} . If we only consider the unary object distance, we can find the optimal object matching separately within each object category by choosing the closest pair over the bipartite matchings between \mathcal{O}_i and \mathcal{H}_i . However, we believe that the pairwise spatial relations can also deliver useful information for measuring distance (as shown in Figure 4.1). Therefore, we need to jointly consider the unary object distance as well as the pairwise interactions. To address the problem, we model the object matchings as a set of latent variables $\mathcal{M} = \{(u_i, v_i)\}_{i \in \mathcal{V}}$, where u_i and v_i are both object indices, and the pair (u_i, v_i) indicates that object $\mathcal{O}_i^{u_i}$ is matched to object $\mathcal{H}_i^{v_i}$ for category i .

Given the class C and the image \mathbf{x} , we can find the optimal settings of \mathcal{H} and \mathcal{M} by minimizing the distance over all possible object configurations and all possible object matchings. Then the minimum distance is treated as the class-to-image distance $D_\theta(C, \mathbf{x})$. Formally, we have

$$D_\theta(C, \mathbf{x}) = \min_{\{\mathcal{H}, \mathcal{M}\}} \theta^\top \Phi(\mathcal{O}, \mathcal{H}, \mathcal{M}, \mathbf{x}), \quad (4.1)$$

where $\theta^\top \Phi(\mathcal{O}, \mathcal{H}, \mathcal{M}, \mathbf{x})$ is a linear function measuring the distance from C to \mathbf{x} accordingly to putative object configurations \mathcal{H} and putative object matchings \mathcal{M} . We define

$$\theta^\top \Phi(\mathcal{O}, \mathcal{H}, \mathcal{M}, \mathbf{x}) = \alpha^\top \psi(\mathcal{O}, \mathcal{H}, \mathcal{M}) + \beta^\top \rho(\mathcal{H}, \mathcal{M}) + \gamma^\top \phi(\mathbf{x}), \quad (4.2)$$

where $\theta = \{\alpha, \beta, \gamma\}$ are the model parameters, and $\Phi = \{\psi, \rho, \phi\}$ is the feature vector defined on $(\mathcal{O}, \mathcal{H}, \mathcal{M}, \mathbf{x})$. Next we describe in detail each component in Eq. (4.2).

Unary object distance $\alpha^\top \psi(\mathcal{O}, \mathcal{H}, \mathcal{M})$: This function measures the unary object distance between \mathcal{O} and \mathcal{H} based on the object matchings \mathcal{M} . To compute the distance between a pair of matched objects, we consider five base distance measures calculated from five local object features including color histograms, HOG [17], LBP [82], texon [74], and location histograms (more details in Section 4.6). The unary object distance is then calculated as a weighted summation over all base distances. Formally, we parameterize this function as:

$$\alpha^\top \psi(\mathcal{O}, \mathcal{H}, \mathcal{M}) = \sum_{i \in \mathcal{V}} \sum_t \alpha_{it} \cdot \psi_t(\mathcal{O}_i^{u_i}, \mathcal{H}_i^{v_i}), \quad (4.3)$$

where $\psi_t(\mathcal{O}_i^{u_i}, \mathcal{H}_i^{v_i})$ is a scalar distance between $\mathcal{O}_i^{u_i}$ and $\mathcal{H}_i^{v_i}$ measured by the type- t features. Note that α_{it} is a scalar parameter that weights the t -th distance measure for all the category- i objects – high weights indicate discriminative object categories. Similar to [36, 37, 75], we restrict α_{it} to be non-negative.

Pairwise object distance $\beta^\top \rho(\mathcal{H}, \mathcal{M})$: This function captures the pairwise spatial relations among certain object categories. Here we follow [20] to define four spatial relations including *on.top.of*, *above*, *below* and *next.to*. Given two object categories (i, j) and the matched objects $(\mathcal{H}_i^{v_i}, \mathcal{H}_j^{v_j})$ in the image \mathbf{x} , we define $\rho_k(\mathcal{H}_i^{v_i}, \mathcal{H}_j^{v_j}) = -1$ if the spatial relation between $\mathcal{H}_i^{v_i}$ and $\mathcal{H}_j^{v_j}$ is consistent with a spatial relation k , and $\rho_k(\mathcal{H}_i^{v_i}, \mathcal{H}_j^{v_j}) = 0$ otherwise. The pairwise object distance is parameterized as:

$$\beta^\top \rho(\mathcal{H}, \mathcal{M}) = \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} \sum_k \beta_{ijk} \cdot \rho_k(\mathcal{H}_i^{v_i}, \mathcal{H}_j^{v_j}), \quad (4.4)$$

where β_{ijk} is a scalar parameter that weights the spatial relation k between object categories i and j – high weights indicate discriminative spatial relations. We also require β_{ijk} to be non-negative. This function implements the idea that we should pull the image \mathbf{x} close to the class C if the spatial relations between the matched objects in the image \mathbf{x} are discriminative for the class C .

Global image appearance distance $\gamma^\top \phi(\mathbf{x})$: This function models the distance based on the global image features $\phi(\mathbf{x})$. It is parameterized as:

$$\gamma^\top \phi(\mathbf{x}) = \sum_g \gamma_g \cdot \phi_g(\mathbf{x}), \quad (4.5)$$

where γ_g is a scalar parameter that weights the g -th global feature $\phi_g(\mathbf{x})$. In fact, the choice of $\phi(\mathbf{x})$ is task-dependent and any robust features can be flexibly encoded in the model. In our experiments, we use the bag-of-words features [10] for object recognition on PASCAL 07, and the GIST descriptors [83] for scene recognition on SUN 09.

4.4 Inference

During testing, we are given the model parameters $\theta = \{\alpha, \beta, \gamma\}$ as well as a collection of unannotated test images. For each test image \mathbf{x} , we need to compute the class-to-image distance $D_\theta(C, \mathbf{x})$. The final decision is made by recognizing images with small distances as positive, and images with large distances as negative. Here the key computational issue is to solve the inference problem in Eq. (4.1).

It is computationally intensive to solve the inference problem because we need to examine all the possible configurations (*i.e.*, locations and scales) for each object category, search over all the possible object matchings, and find the complete configurations and object matchings that jointly minimize the objective function. If we only consider the unary object distance, this results in inferring the optimal object configuration and object matching within each object category independently. We can try each object’s configuration in a sliding window manner, and then examine all the possible object matchings. With our full structured model defined in Eq. (4.2), the inference problem in Eq. (4.1) is computationally infeasible.

To speed up the inference process, we employ several approximation strategies. First, we reduce the search space of location/scale configurations for the objects in an object category. This is

achieved by running an object detector [32] on all locations/scales in \mathbf{x} in a standard sliding window manner, followed by non-maximum suppression to obtain the candidate configurations. In our experiments, we use respectively 5 and 10 candidate configurations for each object category per PASCAL 07 and SUN 09 image. We keep using the notation \mathcal{H}_i to denote the candidate configurations of object category i . When solving the inference problem in Eq. (4.1), we restrict the selected object for object category i to one of its corresponding candidate configurations in \mathcal{H}_i .

The second approximation strategy is for object matchings. Given the candidate configurations \mathcal{H}_i , there are $|\mathcal{O}_i| \times |\mathcal{H}_i|$ possible object matchings for the object category i . It is costly to consider all of them, especially since we need to jointly regard all the object categories in finding the optimal set of object matchings. Here we reduce the search space for category i by only considering $|\mathcal{H}_i|$ candidate object matchings. In detail, for each candidate object configuration $\mathcal{H}_i^v \in \mathcal{H}_i$, we compute the distance from all the objects in \mathcal{O}_i to it. We then assign a candidate object matching by pairing \mathcal{H}_i^v to its closest object $\mathcal{O}_i^{u^*}$ in \mathcal{O}_i . Formally, we identify the candidate object matching by solving the following optimization problem:

$$u^* = \operatorname{argmin}_u \sum_t \alpha_{it} \cdot \psi_t(\mathcal{O}_i^u, \mathcal{H}_i^v). \quad (4.6)$$

Note that the candidate object matchings are still latent (*i.e.*, not observed in the original data) because they change with the model parameters α during learning. When solving the inference problem in Eq. (4.1), we require each object category to select one object matching from the candidate set.

Provided the above approximations, it is easy to show that the inference problem in Eq. (4.1) is now equivalent to the energy minimization problem in a Markov Random Field (MRF) with $|\mathcal{V}|$ nodes [57]. Each node in the MRF corresponds to an object category. The node i has $|\mathcal{H}_i|$ possible states, where the unary energy for each state is the distance calculated by Eq. (4.6) for the corresponding candidate object matching. An edge (i, j) in the MRF corresponds to the relation between object categories i and j .

The optimization problem in Eq. (4.1) is still hard to solve if we have to consider the relation between all pairs of object categories, *i.e.*, when the relation between object categories is represented by a complete graph. To further speed up the process, we prune the graph into a tree structure by considering only frequent spatial relations in the class- C images. In detail, we first assume that only one spatial relation matters for a given pair of object categories, and we choose it as the most frequent spatial relation. The selected spatial relations are then used to construct an undirected weighted (by frequency) graph. We take the maximum spanning tree of this graph as our pruned tree structure for class C . Putting everything together, we can now solve the inference problem in Eq. (4.1) efficiently with Belief Propagation [89].

4.5 Learning

We now describe how to learn the distance function for the class C . Given a set of positive training images $\{\mathbf{x}_p\}_{p=1}^P$ and a set of negative images $\{\mathbf{x}_n\}_{n=1}^N$ of class C , we would like to train the model parameters $\theta = \{\alpha, \beta, \gamma\}$ that tend to associate a small distance to a new test image \mathbf{x} if \mathbf{x} belongs to class C , and a large distance otherwise. A natural way of learning the model is to adopt the latent SVM formulation [32, 23] as follows:

$$\begin{aligned} \min_{\{\alpha, \beta, \xi\} \geq 0} \quad & \frac{1}{2} \|\theta\|^2 + \frac{c}{P} \sum_{pn} \xi_{pn} \\ \text{s.t.} \quad & D_\theta(C, \mathbf{x}_n) - D_\theta(C, \mathbf{x}_p) \geq 1 - \xi_{pn}, \quad \forall p, n. \end{aligned} \quad (4.7)$$

Note that each constraint in Eq. (4.7) constrains that the class-to-image distance from class C to a negative image \mathbf{x}_n should be larger than the distance to a positive image \mathbf{x}_p by a large margin. ξ_{pn} is a slack variable to allow soft-margin. With the constraints, the learned model can discriminate positive and negative images for the class C .

The constrained optimization problem in Eq. (4.7) can be equivalently written as an unconstrained problem:

$$\min_{\{\alpha, \beta\} \geq 0} \frac{1}{2} \|\theta\|^2 + \frac{c}{P} \sum_{pn} (1 + D_\theta(C, \mathbf{x}_p) - D_\theta(C, \mathbf{x}_n)). \quad (4.8)$$

We use the non-convex bundle optimization (NRBM) in [23] to solve Eq. (4.8). The key issue is to compute the subgradient $\partial_\theta D_\theta(C, \mathbf{x})$ for a particular θ . Let $(\mathcal{H}^*, \mathcal{M}^*)$ be the optimal solution to the inference problem we have solved in Section 4.4: $\min_{\{\mathcal{H}, \mathcal{M}\}} \theta^\top \Phi(\mathcal{O}, \mathcal{H}, \mathcal{M}, \mathbf{x})$. Then it can be shown that the subgradient can be calculated as $\partial_\theta D_\theta(C, \mathbf{x}) = \Phi(\mathcal{O}, \mathcal{H}^*, \mathcal{M}^*, \mathbf{x})$. Note that to keep α and β non-negative, we project the negative values in α and β to zeros after each iteration of the NRBM learning.

It is also possible to learn our distance model by using the ground-truth object bounding boxes annotated in the training images without inferring the latent “hypothesized” configurations. However, our experiments suggest that this approach does not perform as well as the learning method defined in Eq. (4.7). This is because the learning of Eq. (4.7) simulates the testing process when unannotated test images are provided for distance calculation.

4.6 Experiments

We evaluate the performance of our method on two image datasets: PASCAL 07 [27] and SUN 09 [13]. We briefly describe our experimental setup before reporting the experimental results in Section 4.6.1. **PASCAL 07 dataset** [27]: The PASCAL Visual Object Challenge provides a standard platform for object recognition. We use the PASCAL 07 dataset for a comparison with previous work. This dataset contains 9,963 annotated images, 5,011 for training and 4,052 for testing. There are 20

image classes, each corresponds to an object category, *e.g.*, *bus*, *table*, *person*, *etc.* The goal is to predict the presence of an object category in a test image. A typical image has around 3 object instances in 2 object categories. On average, an object category contains 783 object instances in the training image set.

SUN 09 dataset [13]: This dataset consists of 12,000 annotated scene images. Similar to [13], we use 4,367 images for training and 4,317 images for testing. There are 111 object categories each containing at least 5 object instances. We filter out small object instances sized less than 20 by 20 pixels, and finally, we have a training set of 4,356 images and a testing set of 4,305 images. A typical image has around 11 object instances in 5 object categories. On average, there are 417 object instances per object category in the training image set. We perform recognition tasks on 58 scene classes each containing at least 10 training and 10 test images². The other small scene classes are only considered as negative data in the experiments.

Note that, as a superset of SUN 09, the SUN dataset [128, 127] also provides a standard benchmark for scene recognition. However, we choose SUN 09 for two reasons. First, the number of object instances per category in SUN 09 is significantly larger than that in SUN (417 as compared to around 65). Second, our method requires ground-truth object bounding boxes on the training set, but only one tenth of the SUN images are annotated.

Local object features: When we perform the experiment, we select or design several state-of-the-art features that are potentially useful for representing object categories. We build color histograms in RGB space. Our histograms have 11 bins in each channel. HOG descriptors [17] provide excellent performance for object recognition. We resize each object instance to 80×100 pixels (which is the average object size), and extract HOG on a regular grid at steps of 8 pixels. In order to characterize image textures, we further use two powerful texture features: texton [74] and LBP [82]. We construct a 128 entry texton dictionary by clustering the responses of a filter bank with 8 orientations, 2 scales, and 2 elongations. A 128-dimensional texton histogram is built for each object instance. LBP are computed using 8 sampling points on a circle of radius 1 together with a uniform mapping of 59 patterns. In this way, we produce a 59-dimensional LBP histogram for each object instance. To represent an object’s absolute location in an image, we partition the image into 5×5 cells, and compute the area of the object instance in each cell. We normalize all the histograms by ℓ_1 norm, and use the histogram intersection distance (*i.e.*, one minus the histogram intersection) to measure the base distance on each feature type.

Global image features: For PASCAL 07, dense SIFT with improved Fisher encoding [90] are shown to outperform the other encoding methods in a fair comparison [10]. We use the implementation of [10] with suggested parameters to extract a 327,680-dimensional feature vector for each image. To improve the learning efficiency, we pre-train 20 SVM classifiers for the 20 image classes based on a kernel calculated from the high-dimensional feature vectors. For an image, the output scores of the 20 SVM classifiers are used to construct a 20-dimensional global appearance feature

²We manually extract the scene labels for the SUN 09 images as they are not included in the original release. The scene labels are available on our website.

vector. For SUN 09, we simply extract the 512-dimensional GIST descriptors [83] with filters tuned to 8 orientations at 4 different scales.

Baselines: We design five baselines by considering different components of our *Full* model. The first one is the *Global* model using Eq. (4.5) only. The second one is our *Unary* model with Eq. (4.3). The third one is the *Unary+Pair* model that incorporates Eqs. (4.3) and (4.4). We further develop two unary models based on Eqs. (4.5) and (4.3): *Global+Unary*, where object matchings are inferred using Eq. (4.6); and *Global+Unary-Latent*, where object matchings are fixed by setting $\alpha_{it} = 1$ in Eq. (4.6). The two unary models are designed to test the efficacy of latent object matchings.

For a fair comparison, we use the same solver for learning all these methods. The learning parameter c in Eq. (4.7) is selected as the best from the range $\{10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\}$. We perform one-vs-all recognition for each image class. Following the PASCAL challenge criterion, the recognition performance on both datasets is measured by average precision (AP) and mean average precision over all classes (mAP).

4.6.1 Results

PASCAL 07: The recognition results on PASCAL 07 are listed in Table 4.1. We first compare *Full* with several state-of-the-art approaches. Our model has significant performance gains over various methods, including similar approaches that operate on object-level representations and explore contextual information in groups of objects [94], a latent SVM model for region-based recognition [133], the winner of PASCAL VOC 2007 using multiple kernel learning on bag-of-word features [78], and the “Dense SIFT + Fisher Encoding” approach which is shown to outperform the other encoding methods [10]. *Full* is comparable with [43] which combines detection and recognition into a unified learning framework, and [12] which is a recent top result on PASCAL 07. We also build our own object bank representations for PASCAL 07. For an image, the representation is a 20-dimensional feature vector, where each dimension corresponds to an object category in PASCAL 07, and its value is the maximum response of an object detector. We train linear SVMs based on the object bank features, leading to OB+SVM in Table 4.1. Our model significantly improves over this method (by 14% mAP). These results validate the effectiveness of the proposed method.

We compare *Full* with *Global*, *Unary* and *Unary+Pair*. Table 4.1 shows that, as a simple combination of these models, *Full* significantly outperforms *Global*, *Unary* and *Unary+Pair* by 4% mAP, 10% mAP and 10% mAP, respectively. This demonstrates that the object matchings learned by local object models (*i.e.*, *Unary* and *Unary+Pair*) provide complementary information to the global image features, and our full model can effectively combine these two sources to build stronger classifiers.

Now we consider *Global+Unary-Latent*, *Global+Unary* and *Full* to evaluate the efficacy of latent object matchings. As shown in Table 4.1, the two latent models (*i.e.*, *Full* and *Global+Unary*) only perform slightly better than the non-latent model *Global+Unary-Latent*, indicating that the latent object matching method does not contribute much to recognition, when the latent variables are inferred by either the unary object distance or the combination of unary and pairwise object distance.

Table 4.1: Object recognition performance (AP and mAP in %) on PASCAL 07. The figures boldfaced are the best performance among *Full* and state-of-the-art methods. Paired *t*-tests are also conducted on the AP values to examine *Full* against all the other methods. We list the returned *p*-values in the last column, where the boldfaced figures indicate no significance between *Full* and the compared methods under 5% significance level.

	plane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	table
Rabinovich <i>et al.</i> [94]	63.0	22.0	18.0	28.0	43.0	46.0	62.0	32.0	37.0	19.0	30.0
Yakhnenko <i>et al.</i> [133]	66.9	43.3	32.4	59.5	16.0	39.2	68.9	38.0	38.5	27.7	27.6
<i>OB+SVM</i>	67.4	79.5	28.7	49.9	47.5	69.4	88.0	51.0	8.6	37.2	19.8
Marszalek <i>et al.</i> [78]	77.5	63.6	56.1	71.9	33.1	60.6	78.0	58.8	53.5	42.6	54.9
Chatfield <i>et al.</i> [10]	79.0	67.4	51.9	70.9	30.8	72.2	79.9	61.4	56.0	49.6	58.4
Harzallah <i>et al.</i> [43]	77.2	69.3	56.2	66.6	45.5	68.1	83.4	53.6	58.3	51.1	62.2
Chen <i>et al.</i> [12]	76.7	74.7	53.8	72.1	40.4	71.7	83.6	66.5	52.5	57.5	62.8
<i>Full</i>	79.2	69.9	48.9	73.2	36.0	75.6	83.7	63.8	55.4	50.0	64.7
<i>Global</i>	77.8	64.8	47.9	71.0	27.9	70.3	81.2	61.0	54.3	46.2	59.5
<i>Unary</i>	62.6	74.9	17.4	34.2	44.3	68.0	86.8	45.6	46.7	39.4	47.3
<i>Unary+Pair</i>	62.9	75.3	17.6	34.4	44.9	66.7	87.6	45.6	46.6	39.4	48.4
<i>Global+Unary-Latent</i>	78.9	69.6	48.7	73.0	35.4	75.7	83.7	63.3	55.5	48.9	64.5
<i>Global+Unary</i>	78.9	69.7	48.9	72.7	35.8	75.5	83.7	63.3	55.7	49.1	64.7

	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP	<i>t</i>-test
Rabinovich <i>et al.</i> [94]	32.0	12.0	31.0	43.0	33.0	41.0	37.0	29.0	62.0	36.0	4.1E-6
Yakhnenko <i>et al.</i> [133]	31.7	66.7	45.8	77.0	12.5	28.8	28.5	61.1	35.0	42.3	3.2E-10
<i>OB+SVM</i>	8.5	78.2	70.5	41.3	33.9	42.4	45.9	75.2	63.6	50.3	2.4E-3
Marszalek <i>et al.</i> [78]	45.8	77.5	64.0	85.9	36.3	44.7	50.6	79.2	53.2	59.4	1.5E-3
Chatfield <i>et al.</i> [10]	44.8	78.8	70.8	85.0	31.7	51.0	56.4	80.2	57.5	61.7	7.1E-4
Harzallah <i>et al.</i> [43]	45.2	78.4	69.7	86.1	52.4	54.4	54.3	75.8	62.1	63.5	7.2E-1
Chen <i>et al.</i> [12]	51.1	81.4	71.5	86.5	36.4	55.3	60.6	80.6	57.8	64.7	5.1E-1
<i>Full</i>	43.7	82.9	74.2	86.3	31.5	52.1	62.2	83.6	64.4	64.1	N/A
<i>Global</i>	41.9	80.3	70.1	85.4	28.3	45.0	53.5	82.1	54.8	60.2	3.7E-6
<i>Unary</i>	22.0	77.0	66.8	90.3	31.0	43.8	43.8	68.9	57.8	53.4	8.7E-4
<i>Unary+Pair</i>	22.8	77.1	68.3	90.3	30.2	40.7	44.9	68.4	59.4	53.6	9.7E-4
<i>Global+Unary-Latent</i>	43.3	82.9	74.2	86.2	31.6	50.4	62.3	83.7	64.7	63.8	4.1E-2
<i>Global+Unary</i>	43.3	82.9	74.2	86.3	31.7	50.7	62.4	83.7	64.5	63.9	5.5E-2

Table 4.2: Recognition results (AP and mAP in %) on SUN 09. We only report AP on the 15 largest scene classes due to space limitations. The mAP results are averaged over all 58 classes. See the caption of Table 4.1 for more details.

	bedroom	skyscraper	street	building	snowy mtn.	kitchen	highway	field
<i>OB+SVM</i>	41.6	50.6	59.1	24.5	55.3	46.1	63.2	40.7
<i>GIST+SVM</i>	24.9	71.9	74.7	30.1	43.5	17.8	78.0	39.3
<i>Full</i>	38.0	67.8	82.3	42.9	54.8	44.8	78.9	54.4
<i>Global</i>	26.7	71.8	76.7	29.0	46.5	23.6	73.3	43.7
<i>Unary</i>	30.8	12.0	51.8	23.6	43.1	28.3	66.3	40.2
<i>Unary+Pair</i>	31.4	15.2	53.3	30.8	46.2	34.6	64.5	50.8
<i>Global+Unary-Latent</i>	38.0	65.4	73.4	27.4	50.0	40.3	74.9	47.2
<i>Global+Unary</i>	37.0	64.6	73.6	32.1	47.7	41.1	74.5	47.3

	bathroom	livingroom	forest	coast	mountain	office	airport	mAP	t-test
<i>OB+SVM</i>	51.7	19.7	60.0	27.6	9.7	10.0	3.8	13.9	1.4E-6
<i>GIST+SVM</i>	22.0	3.9	76.4	17.0	11.6	6.0	10.8	14.2	3.6E-4
<i>Full</i>	50.2	18.9	74.1	31.5	15.7	7.9	9.1	19.2	N/A
<i>Global</i>	23.1	4.1	78.4	28.7	20.4	6.5	7.6	15.3	2.9E-3
<i>Unary</i>	29.4	11.8	17.9	28.0	17.5	6.4	4.3	11.5	1.3E-5
<i>Unary+Pair</i>	34.9	15.2	20.7	31.1	17.3	6.6	4.4	13.0	2.0E-4
<i>Global+Unary-Latent</i>	48.0	13.7	69.1	30.6	19.6	7.0	6.6	17.1	2.7E-4
<i>Global+Unary</i>	46.4	19.4	70.2	32.3	22.4	7.5	8.1	17.6	3.6E-3

This experimental observation is reasonable since the goal of PASCAL 07 object recognition is to decide the presence of an object category in a given test image. Once the object detector fires on the test image, matching the detected object to a particular object in the class does not significantly affect the overall recognition performance. The next dataset, SUN 09, has scenes with multiple objects, for which this ambiguity is more important.

SUN 09: We summarize the recognition results on SUN 09 in Table 4.2. For comparison, we implement two state-of-the-art scene recognition methods. The first is *OB+SVM*, which is the exactly same as the one designed for PASCAL 07. The only difference is that here we employ a 111-dimensional object bank representation, where each dimension corresponds to an object category in SUN 09. We also extract 512-dimensional GSIT descriptors [83] and train a linear SVM for each scene class, *i.e.*, *GIST+SVM*. Our *Full* model significantly outperforms the two methods, and is effective for scene recognition. It is worth noting that our *Global* model operates on the same GIST features as *GIST+SVM*, but achieves better performance by targeting on distance function learning.

Similar to PASCAL 07, our *Full* model significantly outperforms *Global*, *Unary* and *Unary+Pair*, by 4%, 8% and 6% respectively. This result again validates that we can build a strong *Full* model by taking advantage of both global image appearance and local object matchings.

Now we evaluate the efficacy of latent object matchings. Recall that *Global+Unary-Latent* uses fixed object matchings, *Global+Unary* uses latent object matchings based on the unary object distance, and our *Full* model uses latent object matchings inferred by the combination of unary and pairwise object distance. Although we do not see a big performance leap from *Global+Unary-*

Table 4.3: We list the five most discriminative object categories (*i.e.*, highly weighed by α in Eq. (4.3)) with respect to each local object feature on sample scene classes. We also provide the five most discriminative spatial relations (*i.e.*, highly weighed by β in Eq. (4.4)) among these object categories.

	airport	highway
color	airplane, sky, person, truck, streetlight	sky, road, sign, car, tree
HOG	sky, airplane, road, van, door	sky, road, car, sign, tree
texton	tree, door, streetlight, truck, van	sign, car, tree, road, building
LBP	door, truck, streetlight, window, van	sign, car, building, bus, fence
location	tree, truck, van, window, person	sign, car, tree, sky, building
spatial relations	airplane-below-sky person-on.top.of-road truck-on.top.of-road van-on.top.of-road tree-on.top.of-sky	tree-on.top.of-car car-on.top.of-building car-on.top.of-fence bus-on.top.of-car sky-above-road

	bedroom	kitchen
color	bed, wall, curtain, drawer, television	cupboard, stove, cabinet, oven, microwave
HOG	wall, bed, floor, curtain, table	wall, stove, cupboard, floor, oven
texton	bed, drawer, curtain, television, flowers	stove, oven, cabinet, countertop, refrigerator
LBP	drawer, bed, television, flowers, bottle	stove, oven, cabinet, countertop, microwave
location	bed, wall, drawer, television, microwave	stove, cupboard, oven, countertop, cabinet
spatial relations	bottle-next.to-bed television-on.top.of-wall bed-on.top.of-wall table-on.top.of-wall microwave-on.top.of-floor	cupboard-above-floor stove-on.top.of-wall wall-above-floor cabinet-on.top.of-wall refrigerator-on.top.of-wall

Latent to *Global+Unary*, our *Full* model does perform significantly better than *Global+Unary-Latent*. This shows the efficacy of our latent object matching method on scene recognition. Moreover, *Full* also significantly outperforms *Global+Unary*, by exploiting pairwise spatial relations.

As compared to object recognition on PASCAL 07, where the class label is purely determined by one object in the image, scene recognition on SUN 09 is more complicated because we need to consider a collection of objects and their correlations to correctly recognize a test image. To this end, our model explores object-level representations and various contextual information among objects, and the experimental results show that our model is highly effective.

Visualization: We select four scene classes in SUN 09, and view the learned *Full* model in Table 4.3. Sample recognition results are visualized in Figure 4.2. Please refer to the captions for more details.

4.7 Summary

We have presented a discriminative model to learn class-to-image distances for image recognition by considering the structured object matchings between a test image and a set of training images

	top-ranked positive images				top-ranked negative images		
airport	airport @1	airport @2	airport @3	airport @4	arrival gate @6	railway yard @9	
	highway	highway @1	highway @2	highway @3	highway @4	street @51	country road @52
bedroom	bedroom @2	bedroom @3	bedroom @4	bedroom @5	childroom @1	hotelroom @6	
kitchen	kitchen @1	kitchen @2	kitchen @3	kitchen @4	bedroom @12	atrium @24	

Figure 4.2: Sample recognition results using our *Full* model. Each row corresponds to a scene class, and we show the top four ranked positive images and the top two ranked negative images. The title of an image includes the scene class label and a figure indicating the rank of the image according to our learned distance: the smaller the rank, the smaller the distance. For an image, we plot up to four discriminative objects (as listed in Table 4.3) together with the predicted locations. The color of the bounding box shows the relative importance of the objects in distance calculation (sorted by the unary object distance): $red > blue > green > yellow$.

from one class. The model integrates three types of complementary distance including the unary object distance, the pairwise object distance and the global image appearance distance. We formulate a latent variable framework and have proposed efficient inference and effective learning methods. Our experiments validates the efficacy of our model in object recognition and scene recognition tasks. We believe our solution is general enough to be applied in other applications with elementary “object”-level representations, *e.g.*, image retrieval with structured object matchings or video recognition/retrieval with structured action matchings.

Chapter 5

Scene Recognition by Semantic Part Discovery

In this chapter, we present a method to discover parts for scenes. We set our goal as finding semantically meaningful scene parts that are discriminative with each other and representative for recognizing scenes. For discriminativeness, we cluster image patches in a discriminative large-margin learning framework, and treat each cluster as a candidate part. For representativeness, we build scene recognition models on top of part-based image representation, and apply sparse regularization to select representative parts. We formulate patch clustering, part model learning and scene recognition model learning in a joint framework. An alternating descent algorithm is developed to effectively solve the resultant non-convex optimization problem. Experimental results obtained on a standard scene recognition dataset show the efficacy of the proposed method.

5.1 Overview

In computer vision, it has been an active solution to use part-based models for recognition and detection tasks. The famous pictorial structures are proposed to model body parts for human detection and pose estimation [2]. A recent success in object detection learns object parts by deformable part models (DPMs) [32]. Similarly, Pandey and Lazebnik [86] conduct part-based scene recognition by running DPMs on scene images. Part-based modeling usually generates favorable recognition and detection performance, since the parts capture semantic components of the visual concepts.

Despite of the wide usages, however, a problem with part-based modeling is that we need heuristic initialization or detailed annotations to obtain parts. For example, in DPMs, object parts are heuristically initialized to fix locations around each object bounding box. The heuristic initialization is sensitive to highly-deformable objects such as cats and dogs [88], limiting the applicability of DPMs. On the other hand, obtaining accurate and noise-free part annotations is unlikely in any realistic setting.

To address this problem, we are interested in the “automatic” discovery of visual parts. We set our task as understanding scene images in this work. Note that a scene (*e.g.*, *kitchen*, *bedroom*) contains various semantic components corresponding to surfaces (*e.g.*, *wall*, *floor*), objects (*e.g.*, *bed*, *chair*), or even object parts (*e.g.*, *bed frame*, *chair leg*). Thus, the goal of part discovery is to group image patches of the same scene component, and learn a part model to identify that scene component.

We assume a weakly supervised setting where only the image-level scene class are available. It is worth noting that a semantic scene class is usually characterized by a collection of visual parts embedded in scene images, while different scene classes may also favor different part configurations. Therefore, to well capture scene semantics, we would like to discover “useful” parts that are discriminative with each other and representative for recognizing the scene of interest.

To implement the idea we have to address two key issues. The first is: how to learn discriminative part models while the image patches for each part are not yet available? A straightforward solution for this problem is to first cluster image patches (via k-means [42, 72], for example), and then train a discriminative part model for each cluster. However, this solution is sub-optimal since it treats patch clustering and part model learning as two separate steps. To handle this problem, we adopt the maximum-margin clustering (MMC) [131, 138, 139] technique introduced in Chapters 2 and 3 to jointly cluster patches and learn part models.

Note that the parts learned from MMC might be useless for recognizing scenes. For example, the part of *floor* is likely useless for indoor scene recognition since it appears in almost all images. This observation rises another key issue: how shall we select representative parts? To answer this question, we represent an image using part-based representation, where each feature dimension captures the response of a part model on that image. We then train a scene recognition model with sparse regularization. The learned sparse model indicates the representativeness of candidate parts: zero weights are always assigned to useless parts.

The MMC clustering and scene recognition are mutually beneficial. On one hand, MMC generates parts for representing images, thus affecting scene recognition. On the other hand, the scene recognition models determine the relative importance of parts – representative parts should be emphasized in clustering and unrepresentative parts should be discarded. To capture the mutual interactions, we propose to jointly optimize MMC clustering and scene recognition.

The resultant learning objective is non-convex. Thus, we introduce an alternating descent algorithm to solve the problem. Each iteration of the algorithm involves three steps: clustering patches, updating part models, and learning scene recognition models. We evaluate the efficacy of our method on the MIT-Indoor Scene dataset [93]. Qualitative results show that our method learns discriminative and representative parts, and quantitative results validate the efficacy of learned parts for scene recognition.

The rest of this chapter is organized as follows. Section 5.2 reviews related work. Section 5.3 formulates our framework. We describe our optimization algorithm in Section 5.4, followed by experimental results reported in Section 5.5. Finally, Section 5.6 summarizes this chapter.

5.2 Related Work

There has been much work on part discovery in recent years. Depending on the given supervision, we can roughly divide existing approaches into the following three categories.

Unsupervised discovery: Part discovery is typically performed in a purely unsupervised setting. Tuytelaars *et al.* [112] provides a comprehensive survey. Generally speaking, there are two popular directions to address this problem. The first relies on clustering. For instance, Kim and Torralba [54] use link analysis to iteratively refine the regions of interest in cluttered images; Zhu *et al.* [145] develop a bottom-up, saliency-guided multiple class learning method, which performs part localization, discovery, and detector training in an integrated framework. The second line of research utilizes latent topic models. For example, Russell *et al.* [98] build pLSA [45] and LDA [4] on top of image segmentations to discover semantic parts.

Supervised discovery with part labels: Another set of related work assumes that part labels are available on images, and the task is to detect and localize the parts within images. Deselaers *et al.* [21] propose conditional random field that starts from generic knowledge and progressively adapts to new parts. Siva and Xiang [109] achieve favorable performance by properly initializing the part locations and actively detecting the model drift. Prest *et al.* [91] learn part detectors from videos known only to contain frame patches of a target part.

Supervised discovery with image labels: There is also recent progress in discovering parts given only image-level labels. Note that the image labels are not necessarily the same as the part labels. For example, a *bedroom* image could contain parts like *bed*, *window*, *etc.* Our proposed methods follows this setting.

Pandey and Lazebnik [86] discover common scene structures using deformable part-based models [32]. The learned parts for a scene class usually do not correspond to any semantically meaningful components. Li *et al.* [67] advocate for automatic discovery of groups of parts as the basic elements for scene understanding. In contrast to our method, they assume individual object instances are localized and annotated for training.

Some recent work starts by training an exemplar part model for each image patch, and then relies on heuristics to refine the obtained part models. For example, Singh *et al.* [108] train an exemplar-SVM [76] for each patch, and iteratively merge similar part models in a clustering process. Juneja *et al.* [53] train and refine exemplar-LDA model [40] for each patch. To select distinctive part models, they define a heuristic “entropy-rank” measure, which prefers parts occurring in some but not many scene classes. There is also work clustering patches to discover parts. Wang *et al.* [120] advocate for maximum-margin multiple-instance dictionary learning. Doersch *et al.* [24] modify mean-shift clustering to find part models that have high density in positive images but low density in negative images. The most similar work to ours is by Sun and Ponce [111]. It initializes part models by k-means clustering, and then refine them in a latent SVM model regularized by group sparsity. Our method is different since we model patch clustering, part model learning, and scene recognition in a joint framework.

5.3 Part Discovery for Scene Recognition

In this section, we describe our joint framework for learning parts and scenes. Our goal is a unified framework that clusters patches, learns part models, and recognizes scenes. To break down the problem, we consider learning for each scene class independently.

Suppose we have extracted M patches $\{\mathbf{p}_j\}_{j=1}^M$ from the images of a given scene, and we would like to identify K part models from them. A naive approach is to cluster the M patches into K cluster, and then learn a part model for each cluster. However, this naive method has two major drawbacks: (i) the patch clustering and part model learning are disjoint, leading to sub-optimal results, and (ii) due to imperfect patch sampling strategies, the M patches usually include uninformative patches (*i.e.*, background or noise) that damage clustering.

To address the problems, we adopt the idea of MMC to jointly cluster patches and learn part models. In detail, we learn K cluster-specific part models $\mathbf{w} = \{\mathbf{w}_t\}_{t=1}^K$ by enforcing large margins among clusters. We handle uninformative patches by allowing a portion of the M patches unclustered – the uninformative patches do not contribute to form any cluster. We use a variable $t_j \in \{0, 1, \dots, K\}$ to denote the assignment of patch \mathbf{p}_j , where $t_j = 0$ if patch \mathbf{p}_j is unclustered, and $t_j > 0$ if patch \mathbf{p}_j is an instance of the t_j -th part model. We write the cluster assignments of all patches as $\mathbf{t} = \{t_j\}_{j=1}^M$.

We also learn scene recognition models \mathbf{v} with sparse regularization to select representative parts. Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be the set of training scene images, where $y_i = 1$ indicates that the image \mathbf{x}_i is from the scene of interest, and $y_i = -1$ if \mathbf{x}_i belongs to any other scene classes. We represent an image \mathbf{x}_i following the principle of ObjectBank [68]. Specifically, we use a K -dimensional feature vector, where each dimension corresponds to a part model \mathbf{w}_t , and the feature value takes the highest response of \mathbf{w}_t on image \mathbf{x}_i . We enforce sparsity in \mathbf{v} to filter out unrepresentative parts with zero weights. Note that the learned scene recognition model indicates the relative importance of selected parts – irrelevant parts are always assigned with zeros weights. We then consider the relative importance in patch clustering to emphasize or degrade these parts.

Putting everything together, our learning objective is formulated as follows:

$$\begin{aligned}
 \min_{\substack{\mathbf{w}, \mathbf{t}, \mathbf{v} \\ \xi \geq 0, \zeta \geq 0}} \quad & \frac{1}{2}\alpha\|\mathbf{w}\|_2^2 + \frac{1}{K}\sum_{j=1}^M\sum_{t=1, t \neq t_j}^K \xi_{jt} + \beta\|\mathbf{v}\|_1 + \gamma\sum_{i=1}^N \zeta_i^2, & (5.1) \\
 \text{s.t.} \quad & \mathbf{w}_{t_j}^\top \mathbf{p}_j - \mathbf{w}_t^\top \mathbf{p}_j \geq \ell(\mathbf{v}, t_j) - \xi_{jt}, \quad \forall j, t_j > 0, \forall t > 0, t \neq t_j \\
 & L \leq \sum_{j=1}^M \Delta(t_j = t) \leq U, \quad \forall t > 0 \\
 & \sum_{j=1}^M \Delta(t_j = 0) = rM, \\
 & y_i \mathbf{v}^\top \phi(\mathbf{x}_i, \mathbf{w}) \geq 1 - \zeta_i, \quad \forall i
 \end{aligned}$$

where α , β and γ are trade-off parameters, ξ and ζ are the slack variables respectively for patch clustering and scene recognition, and $\Delta(\cdot)$ is an indicator function. Note that $\mathbf{w} = \{\mathbf{w}_t\}_{t=1}^K$ are the part models corresponding to each patch cluster, and \mathbf{v} is the scene recognition model regularized by ℓ_1 sparsity. Next we introduce each constraint in Eq. (5.1) in detail.

The first constraint in Eq. (5.1) enforces that scoring a patch \mathbf{p}_j to its assigned cluster t_j is larger than any other cluster by a margin of $\ell(\mathbf{v}, t_j)$. We define $\ell(\mathbf{v}, t_j)$ to reflect the relative importance of the t_j -th part model:

$$\ell(\mathbf{v}, t_j) = K \frac{|v_{t_j}|}{\|\mathbf{v}\|_1}. \quad (5.2)$$

The second constraint in Eq. (5.1) enforces balanced clusters where L and U are the lower and upper bounds controlling the size of each cluster. The same constraint is used in Chapters 2 and 3 to prevent trivial solution that assigns all patches to one cluster.

The third constraint in Eq. (5.1) accounts for the unclustered patches. These patches are believed to be uninformative background or noise. We simply ignore them when learning part models. We use the parameter $r \in (0, 1]$ to control the portion of unclustered patches: the lower the r , the more uninformative patches.

The last constraint in Eq. (5.1) is for scene recognition. $\phi(\mathbf{x}_i, \mathbf{w})$ is a part-based representation for image \mathbf{x}_i built on top of the K part models. In detail, we follow the practice of ObjectBank [68], and run each part model densely at multiple image locations and scales. We apply max-pooling over each part model’s responses, and concatenate them to form a vector of image features. Formally, it is defined as:

$$\phi(\mathbf{x}_i, \mathbf{w}) = [\max_{\mathbf{p} \in \mathbf{x}_i} \mathbf{w}_1^\top \mathbf{p}, \quad \max_{\mathbf{p} \in \mathbf{x}_i} \mathbf{w}_2^\top \mathbf{p}, \quad \dots, \quad \max_{\mathbf{p} \in \mathbf{x}_i} \mathbf{w}_K^\top \mathbf{p}]^\top, \quad (5.3)$$

where $\mathbf{p} \in \mathbf{x}_i$ is an image window in \mathbf{x}_i at any location or scale. Also note that we use squared hinge loss (*i.e.*, ζ_i^2) in the learning objective for smoothness.

5.4 Optimization

It is easy to verify that the optimization problem defined in Eq. (5.1) is non-convex due to the joint optimization over the part model parameters \mathbf{w} , the patch cluster assignment \mathbf{t} , and the scene recognition model parameters \mathbf{v} . To optimize it, we first eliminate the slack variables ξ and ζ , and rewrite Eq. (5.1) equivalently as:

$$\min_{\mathbf{w}} \frac{1}{2} \alpha \|\mathbf{w}\|_2^2 + R(\mathbf{w}) + A(\mathbf{w}), \quad (5.4)$$

where $R(\mathbf{w})$ and $A(\mathbf{w})$ are the risk functions related to the scene recognition model \mathbf{v} and the cluster assignments \mathbf{t} , respectively.

Algorithm 2 Optimization: An alternating descent algorithm

Input: patches $\{\mathbf{p}_j\}_{j=1}^M$, scene images $\{\mathbf{x}_i, y_i\}_{i=1}^N$, # clusters K , α, β, γ, r , and ϵ

Output: part models \mathbf{w} , patch cluster assignment \mathbf{t} , and the scene recognition model \mathbf{v}

- 1: **Initialize:** part models $\mathbf{w} = \mathbf{w}^1$;
 - 2: **for** $\tau = 1$ to τ_{max} **do**
 - 3: Solve \mathbf{v}^τ in the recognition problem of Eq. (5.5) with \mathbf{w}^τ ;
 - 4: Solve \mathbf{t}^τ in the assignment problem of Eq. (5.6) with \mathbf{w}^τ and \mathbf{v}^τ ;
 - 5: Compute $\frac{\partial R(\mathbf{w})}{\partial \mathbf{w}}|_{\mathbf{w}^\tau}$ and $\frac{\partial A(\mathbf{w})}{\partial \mathbf{w}}|_{\mathbf{w}^\tau}$;
 - 6: Compute $\mathbf{w}^{\tau+1}$, $\mathbf{w}^{\tau*}$, and gap^τ by Algorithm 1 of [23];
 - 7: **if** $gap^\tau \leq \epsilon$ or $\tau == \tau_{max}$ **then**
 - 8: **break;**
 - 9: **Return:** $\mathbf{w} = \mathbf{w}^{\tau*}$, and \mathbf{t} and \mathbf{v} solved from Eqs. (5.5) and (5.6) with $\mathbf{w}^{\tau*}$;
-

In detail, $R(\mathbf{w})$ is a recognition problem defined by:

$$\min_{\mathbf{v}} \beta \|\mathbf{v}\|_1 + \gamma \sum_{i=1}^N \left(\max(0, 1 - y_i \mathbf{v}^\top \phi(\mathbf{x}_i, \mathbf{w})) \right)^2. \quad (5.5)$$

$A(\mathbf{w})$ is an assignment problem defined by:

$$\begin{aligned} \min_{\mathbf{t}, \mathbf{v}} \quad & \frac{1}{K} \sum_{j=1}^M \sum_{t=1, t \neq t_j}^K \max(0, \ell(\mathbf{v}, t_j) - \mathbf{w}_{t_j}^\top \mathbf{p}_j + \mathbf{w}_t^\top \mathbf{p}_j). \\ \text{s.t.} \quad & L \leq \sum_{j=1}^M \Delta(t_j = t) \leq U, \quad \forall t > 0 \\ & \sum_{j=1}^M \Delta(t_j = 0) = rM \end{aligned} \quad (5.6)$$

In order to solve the optimization problem in Eq. (5.4), we develop an alternating descent algorithm shown in Algorithm 2. This algorithm alternates between finding the part model parameters \mathbf{w} and optimizing the other two sets of variables \mathbf{t} and \mathbf{v} . The algorithm mainly consists of three steps performed iteratively. First, the recognition risk is computed from Eq. (5.5) with the current part models \mathbf{w} . Then the assignment risk is evaluated by Eq. (5.6) given \mathbf{w} and \mathbf{v} fixed. Finally, the part model parameters \mathbf{w} are updated from the estimated risk values and gradient. We explain each step in detail in the following.

Recognition risk: Given \mathbf{w} fixed, Eq. (5.5) becomes a standard recognition problem with sparse regularization and squared hinge loss. We use LIBLINEAR [29] to optimize it. To enhance positive data, we flip positive images and add them to training. To speed up the computation of part-based image representation, we run part models on a set of pre-extracted salient patches and collect the maximum response. The details of the salient patch proposal strategy is described in Section 5.5.

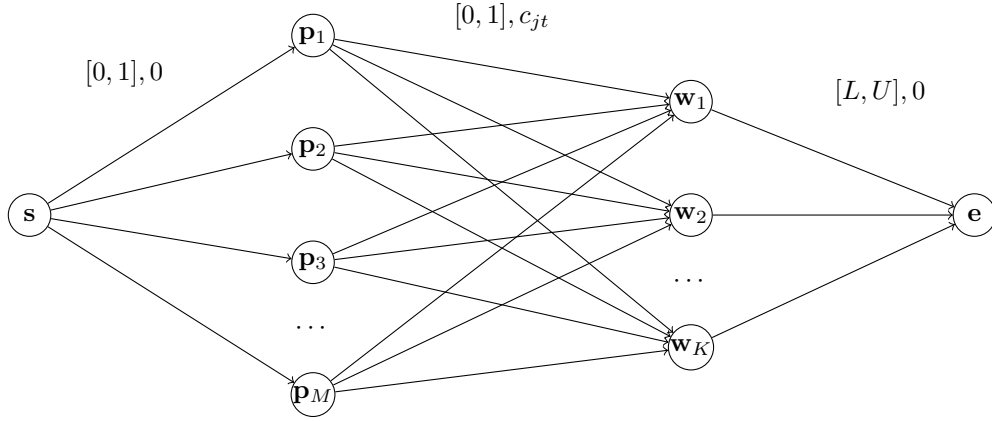


Figure 5.1: A minimum cost flow (MCF) solver for the assignment problem in Eq. (5.6). We send a flow of $(1 - r)M$ units from the starting node s to the end node e , by passing M patch nodes and K cluster nodes, with the assignment cost c_{jt} defined in Eq. (5.7). The edge settings are formatted as: “[lower capacity, upper capacity], cost”.

Assignment risk: With w and v fixed, Eq. (5.6) turns out to be an assignment problem. Note that the assignment cost for assigning patch p_j to a cluster $t > 0$ is computed by

$$c_{jt} = \sum_{t'=1, t' \neq t}^K \max(0, \ell(v, t_j) - \mathbf{w}_t^\top \mathbf{p}_j + w_{t'}^\top \mathbf{p}_j). \quad (5.7)$$

The objective of Eq. (5.6) is to minimize the total assignment cost while maintaining the cluster balance constraints and the constraint on the portion of unclustered patches.

Following Chapter 3, we develop a minimum cost flow (MCF) solver to solve this assignment problem (see Figure 5.1). Please note that the cluster assignment problem in Eq. (5.6) is slightly different from the assignment problem in Chapter 3, as Eq. (5.6) allows the patches to be unclustered. To adopt for this change, we bound the flow capacity from the starting node s to a patch node in the range $[0, 1]$: a flow of 0 indicates an unclustered patch, while a flow of 1 means the patch is assigned to some cluster. The rest of the MCF network is the same as that in Chapter 3. A patch node sends its flow (if any) to a cluster node at the cost of c_{jt} (defined in Eq. (5.7)). Then a cluster node receives flows and sends forward to the end node e , where the edge capacity is limited in between L and U for balanced clusters. It can be proved that the optimal solution of sending $(1 - r)M$ units of flow in this MCF corresponds to the minimum total assignment cost in Eq. (5.6). To find this optimal flow, we use the capacity rescaling algorithm [25] implemented in the LEMON library [22], which is an efficient dual solution method running in $O(MK \cdot \log(M + K) \cdot \log(UMK))$ complexity.

Updating w : The next step of learning is to optimize the part model parameters w in Eq. (5.4). The learning problem is non-convex and we use the non-convex bundle optimization solver in [23]. In a nutshell, this method builds a piecewise quadratic approximation to the objective function of Eq. (5.4) by iteratively adding a linear cutting plane at the current optimum and updating the

optimum. Now the key issue is to compute the subgradients $\frac{\partial R(\mathbf{w})}{\partial \mathbf{w}}$ and $\frac{\partial A(\mathbf{w})}{\partial \mathbf{w}}$ for a particular \mathbf{w} . First, $R(\mathbf{w})$ is related to \mathbf{w} in $\phi(\mathbf{x}_i, \mathbf{w})$. Let \mathbf{p}^{t*} be the highest responding patch in image \mathbf{x}_i for part model \mathbf{w}_t . Then the subgradient $\frac{\partial \phi(\mathbf{x}_i, \mathbf{w})}{\partial \mathbf{w}}$ is calculated as: $[\mathbf{p}^{1*}, \mathbf{p}^{2*}, \dots, \mathbf{p}^{K*}]$. Second, in $A(\mathbf{w})$, the terms related to \mathbf{w} are all in the form of $\mathbf{w}_t^\top \mathbf{p}_j$, whose subgradient w.r.t. \mathbf{w}_t is \mathbf{p}_j . We can easily assemble $\frac{\partial R(\mathbf{w})}{\partial \mathbf{w}}$ and $\frac{\partial A(\mathbf{w})}{\partial \mathbf{w}}$ from the above elementary subgradients. With the obtained $\frac{\partial R(\mathbf{w})}{\partial \mathbf{w}}$ and $\frac{\partial A(\mathbf{w})}{\partial \mathbf{w}}$, we optimize Eq. (5.4) by the algorithm in [23].

5.5 Experiments

We evaluate the performance of our method on the MIT Indoor Scene dataset [93]. We briefly describe our experimental setup before reporting the experimental results in Sections 5.5.1 and 5.5.2. **MIT-Indoor Scene dataset:** This dataset includes 67 indoor scenes ranging from stores (*e.g.*, *deli*, *florist*), home (*e.g.*, *bedroom*, *kitchen*), public spaces (*e.g.*, *church*, *museum*), leisure (*e.g.*, *bar*, *movie theater*), to working place (*e.g.*, *art studio*, *computer room*). Following the setting of [93], we use 80 images for training and 20 images for test within each scene class. For better efficiency, we rescale large images (sized larger than 600×450) to around 600×450 pixels. Quantitative recognition performance is measured by average recognition accuracy as in [93].

Proposal salient patches: To start with, we need to propose a pool of image patches for clustering. A naive solution is to consider all possible image windows, but it is highly inefficient since most image windows are uninformative background or noise. To identify a set of promising patches, we follow the strategy of [53] to perform image over-segmentation on low-level image cues [31]. This process selects an average of 40 patches per images, with each patch taking 64×64 pixels. We build 8×8 HOG cells for each patch and extract 1984-dimensional HOG features [17, 32].

Initialization of patch clusters: We initialize the patch clusters by affinity propagation [34]. Unlike k-means, affinity propagation does not require specifying the desired number of clusters. Instead, the number of clusters is automatically determined from the given data. Also, there is no need for initializing the cluster centers in affinity propagation. The only input parameters are the similarities between each pair of patches. We use the suggested whitened histogram of orientations (WHO) transformation and cosine similarity for measuring patch similarities [40]. We perform affinity propagation for each scene class, and remove small clusters containing no more than 5 patches. The rest (182 per scene) of clusters are kept as the initial set of patches clusters.

Parameters: We fix the portion of unclustered patches as $r = 0.25$ for all experiments. It is an empirical estimation of the percentage of uninformative patches in our salient patch proposal method. The number of clusters K is set as the number of initial clusters obtained above. Furthermore, the upper and lower bounds for cluster balance constraints are computed by $L = 0.5 \frac{rM}{K}$ and $U = 1.5 \frac{rM}{K}$, respectively. The trade-off parameters are set as the best from $\alpha = \{10^{-3}, 10^{-2}, 10^{-1}\}$ and $\beta = \{10^{-1}, 10^0, 10^1\}$, while keeping $\gamma = 1$ for balanced recognition risk and assignment risk. To set the best performing α and β parameters, we use 64 out of 80 training images for model

Table 5.1: Per-class recognition accuracy (%) of the joint learning framework.

APC+SVM	MMC+SVM	Ours
13.53	13.68	20.60

training and the rest 16 images for validation, and select the parameters that achieve the highest recognition accuracy [93] on validation images.

5.5.1 Evaluating the Joint Learning Framework

The proposed method jointly optimizes patch clusters, part models and scene recognition models in a unified framework. To evaluate the benefit of the joint learning framework for scene recognition, we design the following two baselines. The first baseline, APC+SVM, conducts patch clustering (by affinity propagation), part model learning (by learning an SVM for each cluster) and scene recognition (with sparse regularization) in three separate steps. The second baseline, MMC+SVM, jointly clusters patches and learns part models as we do in the proposed method, but recognizes scenes in a separated step with the part models fixed.

The scene recognition performance is reported in Table 5.1. It clearly shows that our method boosts the two baselines considerably, by improving 7% in per-class recognition accuracy. This result validates the efficacy of our joint learning framework.

5.5.2 Evaluating Part Discovery

Now we evaluate the “goodness” of the discovered parts. We first construct image features using the learned part models, and then build scene recognition models on top of the obtained features. This enables quantitative comparison with other state-of-the-art part discovery methods. We also evaluate the representativeness of the selected part models, and visualize the sample parts we have discovered in the MIT-Indoor Scene dataset.

Comparing with state-of-the-art part discovery methods: We follow the principle of ObjectBank [68] to quantitatively evaluate the discovered parts. In detail, we construct part-based representation with the learned part models, where each part model runs densely at every image location at multiple scales (with rescaling factors of $2^{-\frac{i}{3}}$, $i = 0, 1, 2, 3$). We apply max-pooling over each part model’s responses, and concatenate them together as a feature vector. We also capture spatial information via spatial pyramid (1×1 , 2×2 grids), where encodings of each spatial region are stacked together to form the final image features. Note that the part-based image representation we have introduced in our learning process *i.e.*, Eq (5.3), is a simplified version of this strategy for computational efficiency.

For quantitative comparison, we learn one-vs-all linear SVMs on the features and report the recognition performance. Following [53], we use non-linear additive kernels to simulates kernel SVM. It is defined by χ^2 explicit feature mapping [116], and increases the feature dimension by three times. We select the best SVM trade-off parameter in the range of $\{10^0, 10^1, 10^2, 10^3, 10^4\}$

Table 5.2: Per-class recognition accuracy (%) of part discovery methods.

ROI+GIST [93]	26.05	RBoW [87]	37.93	MMDL [120]	50.51
MM-Scene [144]	28.00	D-Patches [108]	38.10	D-Parts [111]	51.40
DPM [86]	30.40	LPR [101]	44.84	IFV [53]	60.77
CENTRIST [126]	36.90	BoP [53]	46.10	DMS [24]	64.03
ObjectBank [68]	37.60	miSVM [69]	46.40	Ours	38.67

Table 5.3: Per-class recognition accuracy (%) using various sets of parts.

Top-10	Top-20	Top-30	Top-40	Top-50	Ours-82	All-182
33.60	34.24	35.70	36.46	37.06	38.67	38.30

via 4-fold cross validation. To enrich positive data, we add flipped positive images for training. We also calibrate different one-vs-all SVM scores by fitting a sigmoid on the validation set, a similar process as in [53].

We compare our discovered parts with state-of-the-art methods. The recognition performance is presented in Table 5.2. Note that it is not a fair comparison since different part discovery methods apply different strategies/parameters in extracting image patches, constructing image features and learning scene recognition models. But the result shows that our discovered parts achieve favorable scene recognition performance. Note that our method uses a smaller number of salient patches than recent methods (*e.g.*, [53, 24]), due to the high computational cost of MMC. We suspect this is a reason for the performance loss of our method as compared to state-of-the-art algorithms in Table 5.2.

Representativeness: We also evaluate the representativeness of the selected parts. Note that our method keeps representative parts based on the sparse scene recognition models. On average, there are 82 out of 182 representative parts selected per scene class.

To validate the selected set of parts, we design the following baseline sets for comparison. We first pick up the top-ranked parts (*e.g.*, top-10, top-20, top-30, top-40, top-50, ranked by the weights learned in the scene recognition models). We also include a baseline set with all the available parts (182 per scene). Similarly as above, we extract image features using each set of parts, and quantitatively compare the scene recognition performance.

The results are reported in Table 5.3, which clearly shows that our selected part set outperforms the other baseline sets. Specifically, compared with using all the available parts, our method achieves better recognition performance while removing more than half of the parts. Also note that the top-ranked parts are always representative for scene recognition, *e.g.*, we only observe a performance drop of 5% by reducing from 82 parts to 10 top-ranked parts. These results validate the effectiveness of our strategy for selecting representative parts.

Visualizations: We visualize the learned parts on sample scene classes in Figure 5.2. Please refer to the caption for detailed explanation.

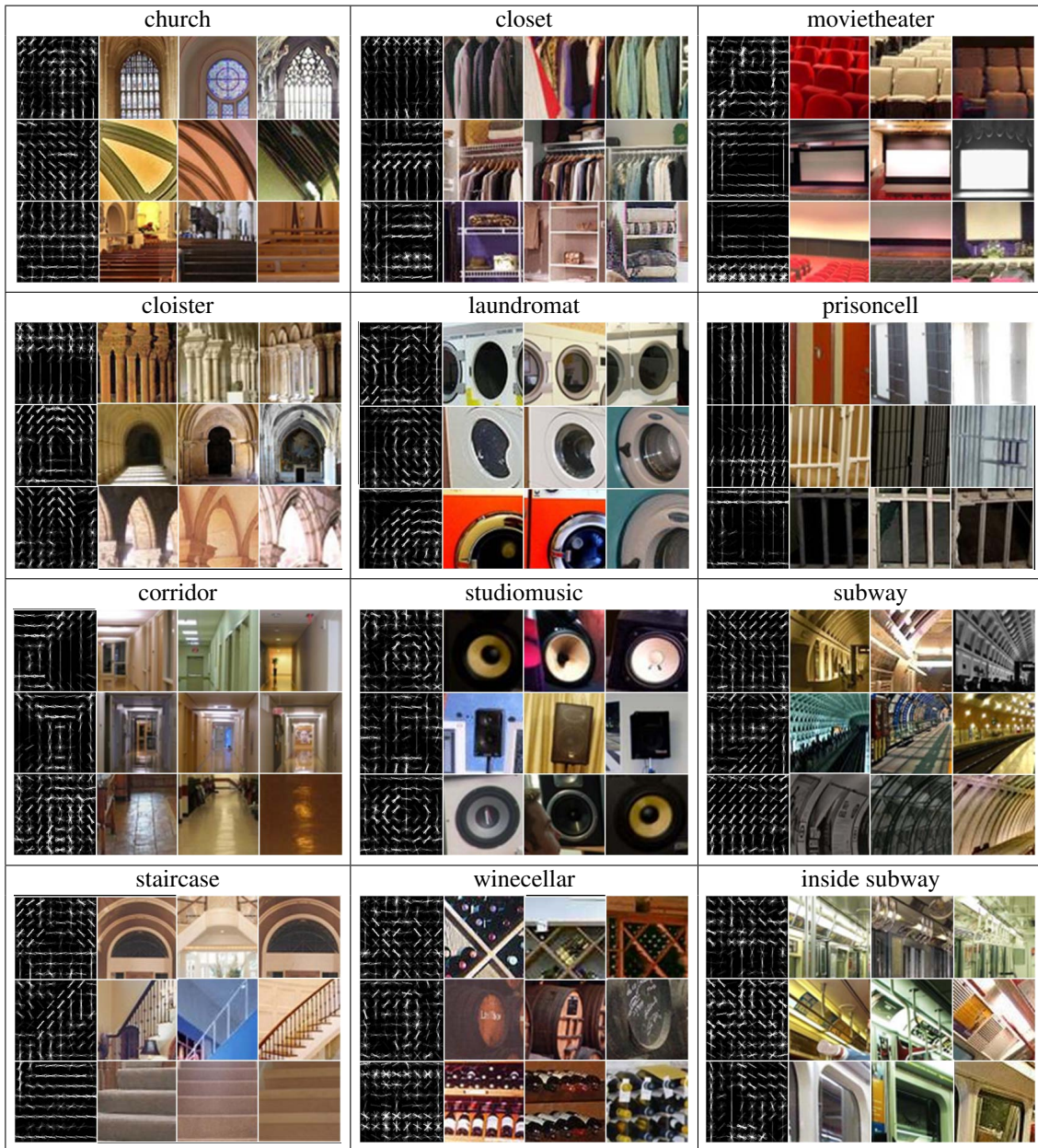


Figure 5.2: Qualitative visualization of the learned parts on 12 scenes (e.g., *church*, *closet*, *movietheater*, etc). For each scene, we show three parts selected by our scene recognition models. Each part accounts for one row in the visualization, where the left-most plot visualizes the part model, and the rest shows three sample patches from the corresponding cluster. Note how discriminative and representative the discovered parts are for the scenes.

5.6 Summary

In this chapter, we have presented a joint learning framework for automatic part discovery in scene images. In detail, we model patch clustering, part model learning, and scene recognition model learning in a unified framework. Our method is able to learn discriminative and representative part models for the scene of interest. We formulate our framework and solve the non-convex objective using an alternating descent algorithm. Experiments conducted on the MIT-Indoor Scene dataset validate the efficacy of our method. As a further extension to our work, it is interesting to apply hierarchical maximum-margin clustering for exploration of hierarchical structures among parts.

Chapter 6

Conclusions, Limitations and Future Directions

Scene recognition is a challenging task in computer vision, and a key to success is to exploit the rich semantics embedded in scenes. Following up this line of research, we raise two major questions in this thesis: *How to discover semantics in visual data?* and *How to recognize scenes by leveraging the discovered semantics?* To answer these questions, we have focused on four directions, as we gradually proceeded from the task of semantics discovery to the task of scene recognition:

Latent maximum-margin clustering. In Chapter 2, we presented a latent maximum-margin clustering (LMMC) framework that clusters data with latent variables. We explore the rich latent representation space when forming clusters, and learn a model for each cluster to reveal and summarize unobserved semantic information embedded in data. The method is implemented by a joint learning of latent variables, cluster assignment, and cluster-specific models.

A major limitation of LMMC is that the learning objective is non-convex and thus the algorithm is sensitive to model initialization. It is beneficial to develop smart initialization strategies for better clustering results. Other than that, LMMC requires extra parameters to control cluster balance. These parameters are task-dependent – we might need expert knowledge or cross validations to set them properly.

Hierarchical maximum-margin clustering. In Chapter 3, we proposed a hierarchical maximum-margin clustering (HMMC) framework for the unsupervised discovery of tree-structured hierarchies. Our method performs flat maximum-margin clustering recursively in a greedy top-down manner. We also enforce sparse and exclusive regularizations to capture the semantics of feature sharing (within each split) and feature competition (among different layers of splits).

A drawback of HMMC is that the tree growing process is not globally optimized. Instead, we apply a greedy splitting criterion that is “optimal” at each local step. On the other hand, the discovered tree structure depends crucially on the branching factor at each split. Other than setting a fixed number as we did in our experiments, it is interesting to design approaches to automatically determine the branching factor for each individual split.

Scene recognition by semantic structure discovery. In Chapter 4, we developed a scene recognition framework by leveraging semantic scene structure discovery. We represent a scene by a structured collage of objects. When presented with a test image, the best matching between this structure and that in the test image is found. We learn class-to-image distance from the object matchings, and recognize the test image based on the learned distance.

The proposed method is limited by extracting only one structure per scene. However, it is intuitive to discover multiple structures within each scene to handle intra-class variations. Also, our model is simplified to localize one object instance per object category on an image. But in practice, there could be multiple object instances of the same category. Thus, it is interesting to consider all object instances in learning and inference.

Scene recognition by semantic part discovery. Finally, in Chapter 5, we designed a method that discovers semantic scene parts for scene recognition. We set our goal as learning discriminative parts that are representative for recognizing scenes. We adopted the maximum-margin clustering technique introduced in Chapters 2 and 3, and proposed a generic framework for jointly learning patch clusters, part models and scene recognition models.

Note that the performance of part discovery relies heavily on the quality of proposed salient patches. A clean and informative set of patches assists in patch clustering and part model learning. Moreover, our method is designed to discover parts for each scene separately. Considering that a part can be shared by multiple scenes, it is promising to model part discovery for all scenes in a unified framework. Furthermore, exploring structures among parts (such as the spatial relations as we discussed in Chapter 4, or the hierarchical structures as we explored in Chapter 3) is another interesting direction.

6.1 Future Directions

To extend the methods and results discussed in this thesis, we plan to take the following directions as future work:

Semantics discovery with supervision: Note that the latent and hierarchical maximum-margin clustering methods are fully unsupervised. The formed clusters might not be exactly the ones we desire. To further refine the results, it is promising to inject supervision in the clustering process.

In the future, we plan to include human in the loop, and ask users to provide feedbacks for clustering. The idea can be implemented by active learning [106] where the clustering algorithm is able to interactively query the user to obtain supervision on certain data instances. It is also interesting to design semi-supervised algorithms [146] if the supervision of partial data is given beforehand. The provided supervision could be formatted in various ways – either we know the cluster assignment of the data, or we have pairwise constraints like must links and cannot links [118], or triplet constraints of relative comparisons [105].

Discovering video semantics for event analysis: The results obtained in this thesis show that it is beneficial to discover semantic structures and parts for recognizing scene images. We believe that the same idea and solution can be directly adopted in the video domain for complex event analysis.

In detail, we are interested in analyzing the rich contents of web videos (*e.g.*, YouTube videos), where each video records a complex event like *birthday party*, *fishing*, *wedding ceremony*, *etc.* To understand the video event, it is essential to discover the semantic video “parts”, including the objects present in the videos, the actions happening in the videos, or the scenes shown in the videos. It is also beneficial to explore the spatial-temporal “structures” among video parts in modeling the rich semantics of video events.

Handling the big data challenge: With the advent of the big data era, numerous scene images are collected in our daily experience. The solution presented in this thesis is being challenged in the following two aspects. On one hand, it is computationally intensive for the semantics discovery and scene recognition algorithms to handle the large-scale image data. Thus, designing efficient and effective optimization methods is highly demanded. On the other hand, the hand-crafted features for describing scene parts and structures are insufficient to model the ever-growing and high dynamic scene collections. We plan to learn powerful features with deep learning techniques, *e.g.*, convolutional neural networks [64, 58].

Bibliography

- [1] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *NIPS*, 2002.
- [2] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *CVPR*, 2009.
- [3] Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, 2005.
- [4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, 2003.
- [5] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.
- [6] Daniel Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery (DMKD)*, 2(4):325–344, 1998.
- [7] Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming (MP)*, 63(2):129–156, 1994.
- [8] Rui M. Castro, Mark Coates, and Robert D. Nowak. Likelihood based hierarchical clustering. *IEEE Transactions on Signal Processing (TSP)*, 52(8):2308–2321, 2004.
- [9] Yuning Chai, Victor S. Lempitsky, and Andrew Zisserman. BiCoS: A bi-level co-segmentation method for image classification. In *ICCV*, 2011.
- [10] Ken Chatfield, Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: An evaluation of recent feature encoding methods. In *BMVC*, 2011.
- [11] Changyou Chen, Jun Zhu, and Xinhua Zhang. Robust bayesian max-margin clustering. In *NIPS*, 2014.
- [12] Qiang Chen, Zheng Song, Yang Hua, ZhongYang Huang, and Shuicheng Yan. Hierarchical matching with side information for image classification. In *CVPR*, 2012.
- [13] Myung Jin Choi, Joseph J. Lim, Antonio Torralba, and Alan S. Willsky. Exploiting hierarchical context on a large database of object categories. In *CVPR*, 2010.
- [14] Michael A. Cohen, George A. Alvarez, and Ken Nakayama. Natural-scene perception requires attention. *Psychological Science (PSS)*, 22(9):1165–1172, 2011.

- [15] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning (ML)*, 20(3):273–297, 1995.
- [16] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research (JMLR)*, 2:265–292, 2001.
- [17] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
- [18] Jia Deng, Alexander C. Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010.
- [19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.
- [20] Chaitanya Desai, Deva Ramanan, and Charless Fowlkes. Discriminative models for multi-class object layout. In *ICCV*, 2009.
- [21] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. Localizing objects while learning their appearance. In *ECCV*, 2010.
- [22] Balázs Dezs, Alpár Jüttner, and Péter Kovács. LEMON - an open source C++ graph template library. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 264(5):23–45, 2011.
- [23] Trinh Minh Tri Do and Thierry Artières. Large margin training for hidden Markov models with partially observed states. In *ICML*, 2009.
- [24] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Mid-level visual element discovery by discriminative mean-shift. In *NIPS*, 2013.
- [25] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of ACM (JACM)*, 19(2):248–264, 1972.
- [26] Karla K. Evans and Anne Treisman. Perception of objects in natural scenes: Is it really attention free? *Journal of Experimental Psychology: Human Perception and Performance (JEPHPP)*, 31(6):1476–1492, 2005.
- [27] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes challenge 2007 (VOC2007) results. <http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html>, 2007.
- [28] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.
- [29] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research (JMLR)*, 9:1871–1874, 2008.
- [30] Ali Farhadi and Mostafa Kamali Tabrizi. Learning to recognize activities from the wrong view point. In *ECCV*, 2008.

- [31] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004.
- [32] Pedro F. Felzenszwalb, David A. McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.
- [33] Robert Fergus, Hector Bernal, Yair Weiss, and Antonio Torralba. Semantic label sharing for learning with many categories. In *ECCV*, 2010.
- [34] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.
- [35] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *CoRR*, abs/1001.0736, 2010.
- [36] Andrea Frome, Yoram Singer, and Jitendra Malik. Image retrieval and classification using local distance functions. In *NIPS*, 2006.
- [37] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007.
- [38] Jacob Goldberger and Sam T. Roweis. Hierarchical clustering of a mixture model. In *NIPS*, 2004.
- [39] Raghuraman Gopalan and Jagan Sankaranarayanan. Max-margin clustering: Detecting margins from projections of points on lines. In *CVPR*, 2011.
- [40] Bharath Hariharan, Jitendra Malik, and Deva Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012.
- [41] Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. Semantic measures for the comparison of units of language, concepts or entities from text and knowledge base analysis. *CoRR*, abs/1310.1285, 2013.
- [42] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Journal of the Royal Statistical Society, Series C (JRSS)*, 28(1):100–108, 1979.
- [43] Hedi Harzallah, Frédéric Jurie, and Cordelia Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.
- [44] Minh Hoai and Andrew Zisserman. Discriminative sub-categorization. In *CVPR*, 2013.
- [45] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, 1999.
- [46] Chiao-Fang Hsu, James Caverlee, and Elham Khabiri. Hierarchical comments-based clustering. In *SAC*, 2011.
- [47] Sung Ju Hwang, Kristen Grauman, and Fei Sha. Learning a tree of metrics with disjoint visual features. In *NIPS*, 2011.
- [48] Sung Ju Hwang, Kristen Grauman, and Fei Sha. Semantic kernel forests from multiple taxonomies. In *NIPS*, 2012.

- [49] Hamid Izadinia and Mubarak Shah. Recognizing complex events using large margin joint low-level event model. In *ECCV*, 2012.
- [50] Anil Jain and Richard Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
- [51] Xin Jin and Jiawei Han. K-medoids clustering. In *Encyclopedia of Machine Learning*, pages 564–565. Springer, 2010.
- [52] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999.
- [53] Mayank Juneja, Andrea Vedaldi, C. V. Jawahar, and Andrew Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013.
- [54] Gunhee Kim and Antonio Torralba. Unsupervised detection of regions of interest using iterative link analysis. In *NIPS*, 2009.
- [55] Seyoung Kim and Eric P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, 2010.
- [56] Alexander Kläser, Marcin Marszalek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.
- [57] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(10):1568–1583, 2006.
- [58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
- [59] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, 2010.
- [60] Tarald O. Kvalseth. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man and Cybernetics (TSMC)*, 17(3):517–519, 1987.
- [61] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.
- [62] Tian Lan, Weilong Yang, Yang Wang, and Greg Mori. Image retrieval with structured object queries using latent ranking SVM. In *ECCV*, 2012.
- [63] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.
- [64] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [65] Jason D. Lee, Yuekai Sun, and Michael A. Saunders. Proximal Newton-type methods for convex optimization. In *NIPS*, 2012.
- [66] Yong Jae Lee and Kristen Grauman. Object-graphs for context-aware category discovery. In *CVPR*, 2010.

- [67] Congcong Li, Devi Parikh, and Tsuhan Chen. Automatic discovery of groups of objects for scene understanding. In *CVPR*, 2012.
- [68] Li-Jia Li, Hao Su, Eric P. Xing, and Fei-Fei Li. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010.
- [69] Quanquan Li, Jiajun Wu, and Zhuowen Tu. Harvesting mid-level visual concepts from large-scale internet images. In *CVPR*, 2013.
- [70] Yu-Feng Li, Ivor W. Tsang, James Tin-Yau Kwok, and Zhi-Hua Zhou. Tighter and convex maximum margin clustering. In *AISTATS*, 2009.
- [71] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *CVPR*, 2011.
- [72] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory (TIT)*, 28(2):129–136, 1982.
- [73] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.
- [74] Jitendra Malik, Serge Belongie, Thomas K. Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision (IJCV)*, 43(1):7–27, 2001.
- [75] Tomasz Malisiewicz and Alexei A. Efros. Recognition by association via learning per-exemplar distances. In *CVPR*, 2008.
- [76] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *ICCV*, 2011.
- [77] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [78] Marcin Marszalek, Cordelia Schmid, Hedi Harzallah, and Joost van de Weijer. Learning object representations for visual object class recognition. In *Visual Recognition Challenge Workshop at ICCV*, 2007.
- [79] Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society, Series B (JRSS)*, 70(1):53–71, 2008.
- [80] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.
- [81] Jorge Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation (MCOM)*, 35(151):773–782, 1980.
- [82] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(7):971–987, 2002.
- [83] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision (IJCV)*, 42(3):145–175, 2001.

- [84] Paul Over, George Awad, Jonathan Fiscus, Alan F. Smeaton, Wessel Kraaij, and Georges Quenot. TRECVID 2011 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID*, 2011.
- [85] Stephen E. Palmer. *Vision Science: Photons to Phenomenology*. MIT Press, 1999.
- [86] Megha Pandey and Svetlana Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *ICCV*, 2011.
- [87] Sobhan Naderi Parizi, John G. Oberlin, and Pedro F. Felzenszwalb. Reconfigurable models for scene recognition. In *CVPR*, 2012.
- [88] Omkar M. Parkhi, Andrea Vedaldi, C. V. Jawahar, and Andrew Zisserman. The truth about cats and dogs. In *ICCV*, 2011.
- [89] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *AAAI*, 1982.
- [90] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.
- [91] Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid, and Vittorio Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, 2012.
- [92] Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *ACM MM*, 2007.
- [93] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *CVPR*, 2009.
- [94] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *ICCV*, 2007.
- [95] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association (JASA)*, 66(336):846–850, 1971.
- [96] Richard Redner and Homer Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.
- [97] Mikel D. Rodriguez, Javed Ahmed, and Mubarak Shah. Action MACH a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008.
- [98] Bryan C. Russell, William T. Freeman, Alexei A. Efros, Josef Sivic, and Andrew Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006.
- [99] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision (IJCV)*, 77(1-3):157–173, 2008.
- [100] Sreemananath Sadanand and Jason J. Corso. Action Bank: A high-level representation of activity in video. In *CVPR*, 2012.
- [101] Fereshteh Sadeghi and Marshall F. Tappen. Latent pyramidal regions for recognizing scenes. In *ECCV*, 2012.

- [102] Mark Schmidt. *Graphical Model Structure Learning with ℓ_1 -Regularization*. PhD thesis, UBC, 2010.
- [103] Florian Schroff, C. Lawrence Zitnick, and Simon Baker. Clustering videos by location. In *BMVC*, 2009.
- [104] Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local SVM approach. In *ICPR*, 2004.
- [105] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *NIPS*, 2003.
- [106] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin Madison, 2009.
- [107] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):888–905, 2000.
- [108] Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012.
- [109] Parthipan Siva and Tao Xiang. Weakly supervised object detector learning with model drift detection. In *ICCV*, 2011.
- [110] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *Text Mining Workshop at KDD*, 2000.
- [111] Jian Sun and Jean Ponce. Learning discriminative part detectors for image classification and cosegmentation. In *ICCV*, 2013.
- [112] Tinne Tuytelaars, Christoph H. Lampert, Matthew B. Blaschko, and Wray L. Buntine. Unsupervised object discovery: A comparison. *International Journal of Computer Vision (IJCV)*, 88(2):284–302, 2010.
- [113] Arash Vahdat and Greg Mori. Handling uncertain tags in visual recognition. In *ICCV*, 2013.
- [114] Shivakumar Vaithyanathan and Byron Dom. Model-based hierarchical clustering. In *UAI*, 2000.
- [115] Hamed Valizadegan and Rong Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *NIPS*, 2006.
- [116] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.
- [117] Kevin Vervier, Pierre Mahé, Alexandre D’Aspremont, Jean-Baptiste Veyrieras, and Jean-Philippe Vert. On learning matrices with orthogonal columns or disjoint supports. In *ECML/PKDD*, 2014.
- [118] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *ICML*, 2001.
- [119] Hua Wang, Heng Huang, Farhad Kamangar, Feiping Nie, and Chris H. Q. Ding. Maximum margin multi-instance learning. In *NIPS*, 2011.

- [120] Xinggang Wang, Baoyuan Wang, Xiang Bai, Wenyu liu, and Zhuowen Tu. Max-margin multiple-instance dictionary learning. In *ICML*, 2013.
- [121] Yang Wang and Liangliang Cao. Discovering latent clusters from geotagged beach images. In *MMM*, 2013.
- [122] Yang Wang and Greg Mori. Max-margin hidden conditional random fields for human action recognition. In *CVPR*, 2009.
- [123] Yang Wang and Greg Mori. A discriminative latent model of image region and object tag correspondence. In *NIPS*, 2010.
- [124] Zhengxiang Wang, Shenghua Gao, and Liang-Tien Chia. Learning class-to-image distance via large margin and ℓ_1 -norm regularization. In *ECCV*, 2012.
- [125] Zhengxiang Wang, Yiqun Hu, and Liang-Tien Chia. Image-to-class distance metric learning for image classification. In *ECCV*, 2010.
- [126] Jianxin Wu and James M. Rehg. CENTRIST: A visual descriptor for scene categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(8):1489–1501, 2011.
- [127] Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. SUN database: Exploring a large collection of scene categories. *International Journal of Computer Vision (IJCV)*, pages 1–20, 2014.
- [128] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.
- [129] Lin Xiao, Dengyong Zhou, and Mingrui Wu. Hierarchical classification via orthogonal transfer. In *ICML*, 2011.
- [130] Kai Xu, Rui Ma, Hao Zhang, Chenyang Zhu, Ariel Shamir, Daniel Cohen-Or, and Hui Huang. Organizing heterogeneous scene collections through contextual focal points. *ACM Transactions on Graphics (TOG)*, 33(4):35:1–35:12, 2014.
- [131] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *NIPS*, 2004.
- [132] Linli Xu and Dale Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *AAAI*, 2005.
- [133] Oksana Yakhnenko, Jakob Verbeek, and Cordelia Schmid. Region-based image classification with a latent SVM model. Technical report, INRIA, 2011.
- [134] Roman V. Yampolskiy. Turing test as a defining feature of AI-completeness. In *Artificial Intelligence, Evolutionary Computing and Metaheuristics - In the Footsteps of Alan Turing*, pages 3–17. Springer, 2013.
- [135] Weilong Yang and George Toderici. Discriminative tag learning on YouTube videos with latent sub-tags. In *CVPR*, 2011.

- [136] Weilong Yang, Yang Wang, Arash Vahdat, and Greg Mori. Kernel latent SVM for visual recognition. In *NIPS*, 2012.
- [137] Chun-Nam John Yu and Thorsten Joachims. Learning structural SVMs with latent variables. In *ICML*, 2009.
- [138] Kai Zhang, Ivor W. Tsang, and James T. Kwok. Maximum margin clustering made practical. In *ICML*, 2007.
- [139] Bin Zhao, Fei Wang, and Changshui Zhang. Efficient multiclass maximum margin clustering. In *ICML*, 2008.
- [140] Guang-Tong Zhou, Sung Ju Hwang, Mark Schmidt, Leonid Sigal, and Greg Mori. Hierarchical maximum-margin clustering. *CoRR*, abs/1502.01827, 2015.
- [141] Guang-Tong Zhou, Tian Lan, Arash Vahdat, and Greg Mori. Latent maximum margin clustering. In *NIPS*, 2013.
- [142] Guang-Tong Zhou, Tian Lan, Weilong Yang, and Greg Mori. Learning class-to-image distance with object matchings. In *CVPR*, 2013.
- [143] Yang Zhou, Rong Jin, and Steven C. H. Hoi. Exclusive lasso for multi-task feature selection. In *AISTATS*, 2010.
- [144] Jun Zhu, Li-Jia Li, Fei-Fei Li, and Eric P. Xing. Large margin learning of upstream scene understanding models. In *NIPS*, 2010.
- [145] Jun-Yan Zhu, Jiajun Wu, Yichen Wei, Eric I.-Chao Chang, and Zhuowen Tu. Unsupervised object class discovery via saliency-guided multiple class learning. In *CVPR*, 2012.
- [146] Xiaojin Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin Madison, 2005.