Hierarchical Modeling and Applications to Recognition Tasks

Thesis submitted for the degree of

"Doctor of Philosophy"

by

Alon Zweig

Submitted to the Senate of the Hebrew University

August / 2013

This work was carried out under the supervision of:
Prof. Daphna Weinshall

To Roni & Ofri.

Acknowledgements

This thesis marks the end of a wonderful period in my life.

First and foremost I would like to thank my advisor Prof. Daphna Weinshall. I am grateful to have been guided by your wisdom, friendship and support. Our discussions were priceless, your endless curiosity together with the crisp clarity into the finest of the details were intellectually stimulating and led me securely through my research path. I always enjoyed our meetings.

I would also like to thank my thesis committee Professors Lihi-Zelnik Manor and Amir Globerson for improving my work through their suggestions and comments.

I enjoyed immensely studying at the Hebrew university and specifically at the Computer Science Department where I met some of my best friends. The people are truly great. The list is too long and I am afraid I'll forget somebody, so thanks everybody for great discussions, coffee breaks and for making the travel to Jerusalem always worthwhile.

During the first years of my PhD I was part of DIRAC, an EU founded project hosting a large group of researchers. Being part of DIRAC I had the privilege to collaborate with Prof. Jörn Anemüller, Dr. Jörg-Hendrik Bach, Dr. Barbara Caputo, Dagan Eshar, Avishai Hendel, Prof. Hynek Hermansky, Dr. Roger Jie Luo, Dr. Fabian Nater, Dr. Francesco Orabona, Prof. Tomas Pajdla and many others. To all the DIRAC people, I benefited greatly from all our meetings and learnt a lot from our collaborations.

To Dr. Margarita Osadchy, I always enjoyed coming to Haifa for our deep discussions on object recognition. To Dr. Manik Varma, meeting you in Delhi was a true intellectual experience.

To all those I forgot to mention and to all those I met in conferences, summer schools, while organizing the vision seminar and other venues, thanks for being a community of interesting and friendly colleagues.

To my grandmothers, Nita and late Savta Stella, thanks for your love and always being so proud of me, it means a lot. My brother, Eytan, after all these years meeting so many smart people you are still one of the smartest people I know. I cherish the fact that we manage to stay close despite the physical distance. To my parents, thanks for always being so loving and supportive. You have always supported me in any thing I did. It is probably the intellectual yet not pretentious home I grew up in which encouraged my choice of a life of knowledge. Words cannot express how lucky I feel growing up in such a warm and loving family.

To Roni, my smart and beautiful wife, whom I met half way through my doctoral studies. I cannot imagine how this period would have been without you. Thanks for adding love and color to my life. Making me laugh each day. I love you.

Abstract

In this thesis we investigate the notion of hierarchical organization of tasks, specifically visual object class recognition tasks. As the visual recognition field advances, we are able to consider a significantly larger set of object classes simultaneously. The larger scale adds many challenges. Yet, exploring the relationship among tasks carries several potential modeling benefits, such as speeding up search or classification tasks, guide information sharing to improve the generalization properties of each of the individual tasks or better structure the class space to improve discriminative modeling.

While exploiting hierarchical structures has recently become popular in the object class recognition field, mostly these structures are considered for speed up purposes. We consider hierarchies from a different perspective aiding in recognition tasks either by the notion of information sharing or by enabling meaningful semantic labeling for unknown tasks such as novel subclass detection.

We explore different types of hierarchies and propose a unified model for the class membership and part membership hierarchies. In the context of generalization analysis we define an explicit hierarchical learning structure with provable guarantees in the multi-task setting, showing that an hierarchical information sharing approach can indeed improve generalization. We describe a general framework for novelty detection and discuss its specific application to novel subclass detection. Finally we consider the setting of sharing information while exploiting hierarchical relations even when the hierarchy is not explicitly known.

The thesis covers both theoretical ideas and practical algorithms which can scale up to very large recognition settings. Large scale object recog-

nition is a particularly motivating setting in which the number of relations among tasks is presumably large enabling a rich hierarchical model. To validate our ideas, we provide both theoretical guarantees and many experimental results testing the different applications we consider.

Contents

1

Introduction

Hierarchies are everywhere.

Consider societies where the notion of a hierarchy acts as the basis of social organization: the caste system in India, the lower-middle-upper class in western civilization, management hierarchy at the workplace, armies, schools etc. Hierarchies add a structure of order, identifying specific properties, unique to different levels in the hierarchy, such as assigning different groups of society to different levels in the hierarchy, with different assumed capabilities, responsibilities, privileges etc.

In biological/physiological research, it has been shown that hierarchies play an important role in perception. Various studies have shown different aspects of hierarchal organization of object categories in humans. A well known example of such a study of the nature of human hierarchical representation, is Rosch's Basic-Level theory [1]. A different study, by Kiani et al. [2], shows neurophysiological findings of visual object hierarchy. Recently [3], a correspondence was shown between the hierarchical structure of different scene categories and the hierarchical structure at the perceptual processing level.

In computer science, the concept of hierarchies is a fundamental concept, which is deeply rooted in many different areas. Hierarchies are used to organize models such as the notion of inheritance in code design, efficient structures such as trees for searching [4], compact representation of information such as Huffman coding [5] etc.

In the pattern recognition field, hierarchical structures have also been used widely to speed up processes such as nearest neighbor search using k-d trees [4], creating efficient classifiers capturing hierarchical correlations in the data such as CART [6], considering non-linear classifiers via several layers of processing in neural networks etc.

Hierarchies are everywhere and are applied successfully in many applications. As the fields of machine learning and visual object class recognition evolve, progressing from the task of learning a specific object class, to settings where many object

classes are considered simultaneously, new questions arise dealing with the modeling of relations among the different object classes or individual learning tasks.

In this thesis we propose modeling these relationships via the notion of a hierarchy, presenting both theoretical models of hierarchies as well as practical applications based on hierarchical structures for which we propose efficient algorithms. Our main motivation and focus is the field of visual object class recognition, though the scope of the methods we present is wider and applicable to more general fields of machine learning and pattern recognition. Specifically, we apply the notion of hierarchy to two main applications: novel subclass detection and information sharing.

The rest of the introduction is organized as follows: in 1.1 we give a general summary of hierarchal methods in visual recognition. In 1.2 and 1.3 we present the background and related work to the specific applications we consider: sharing of information and novelty detection, respectively. Finally in 1.4 we present an overview of the hierarchical methods we developed and give a short summary of each of the four chapters in this thesis whilst stating their corresponding peer-reviewed publications.

1.1 Hierarchy in Visual Recognition

The notion of semantic hierarchies in the visual recognition and machine learning communities, has attracted an increasing amount of attention over the past few years. As the field evolved from considering recognition tasks consisting of a single object class or a small set of classes, to much larger scenarios where a system is expected to deal with thousands or more categories, researchers in the field have found that different types of hierarchies can be beneficial in many scenarios. This includes: exploiting hierarchies for information sharing [7, 8, 9, 10, 11, 12] (see Section 1.2.4), constructing an efficient search space in classification tasks [13, 14, 15, 16, 17, 18, 19], guiding the training of classifiers for a given hierarchy [9, 20, 21, 22], fine grained/subordinate-level classification [23], learning distance matrices [24, 25] and considering a richer semantic output label space [26].

Building hierarchies for efficient search As the amount of information grows, with endless visual data available on the web and labeled datasets which keep grow-

ing (e.g [27] which currently contains over 20K categories), organizing the data for efficient access is becoming extremely important.

Consider the multi-class setting (classifying a sample into one of many classes): traditionally this setting is approached using a group of binary classifiers in a *1-vs-1* or *1-vs-rest* manner. Classification based on voting in the *1-vs-1* scheme is quadratic in the number of classes, as all pairs of classes need to be considered. Embedding the classifiers in a directed acyclic graph results in linear complexity in the number of classes [28]. In the *1-vs-rest* scheme, the classification is based on the max value of all classifiers and thus, it is also linear in the number of classes. As the number of classes grows, classification which is linear in the number of classes, might be too slow. To address this issue, hierarchical structures are usually considered [13, 14, 15, 16, 17, 18, 19], achieving classification runtime which is sub-linear in the number of classes. For instance, [16] speedup classification by training a classifier, based on a supervised hierarchy known in advance, the classifier is trained using a structured learning algorithm to guide the training according to the hierarchy structure.

Training classifiers based on data driven hierarchies is typically done in a two stage approach: first the hierarchy is discovered and then classifiers are trained for nodes in the hierarchy [13, 14, 15, 16, 17]. Such approaches may suffer from reduced accuracy as the discovered hierarchal structure was not optimized for the classification objective. [18, 19] propose training the classifiers and discovering the hierarchy jointly, at a single learning step, thus choosing the structure which best fits the task at hand while formalizing a clear trade-off between efficiency and accuracy.

Several different cues can be used to discover hierarchies: [13] build a hierarchy in a top-down manner based on max-cut computed over KL-divergence between feature distributions from different classes. Both [15] and [17] construct the hierarchy based on the confusion matrix calculated for the different classes. [29] proposes a recursive top-down approach splitting the classes into two at each step, based on k-means or measures such as mean distance to the origin or size of class (when focusing on efficiency rather then accuracy).

Originally suggested in [14] and recently adopted by [18] is the notion of relaxed hierarchy, where clustering decisions for a subset of confusing classes is delayed to a lower (more specific) level of the hierarchy. This flexibility eases the discovery of separable clusters of classes, which might not be the case if a confusing class cannot

be skipped. Both [14, 18] consider the more general DAG structure rather than being limited to a tree.

In addition to the discriminative supervised approaches, unsupervised generative learning approaches were also considered to construct visual hierarchies [30, 31]. Both methods adopt the hierarchical LDA model [32] to the vision domain. hLDA [32] describes a document (image) as a set of topics (visual words) chosen along a path in a tree of topics.

Part Hierarchies Hierarchies structuring the object class space (class-membership) are a natural choice when dealing with large setting, with many classes. Yet, a different type of hierarchy is also popular in the visual recognition community - the part membership hierarchy [33, 34, 35]. Such hierarchies consider specific parts which are built out of more general parts in the hierarchy. Typically these hierarchies are used to represent objects. In [33] the first layers of the hierarchy are built in an unsupervised manner capturing general low level feature characteristic, on-top of which class specific parts are trained in a supervised manner. [34] proposes representing an image as a segmentation tree and constructing the part hierarchy by finding shared segments among different images. In [35] a hierarchy of semantic parts is built in two stages: first an initial hierarchy of fragments [36] is discovered and then this hierarchy is used to find semantic equivalent fragments in all training images.

1.2 Information Sharing

Information sharing can be a very powerful tool in various domains. Consider visual object recognition where different categories typically share much in common: cars and trucks can be found on the road and both classes have wheels, whilst cows and horses have four legs and can be found in the field together, etc. Accordingly, many different information sharing approaches have been developed [7, 8, 9, 10, 11, 12, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55], exploiting the power of sharing.

Information sharing is usually considered in either the joint-learning or knowledge-transfer settings. In the joint learning scenario we consider the multi-task and multi-

class settings. *Multi-task* [38] is a setting where there are several individual tasks which are trained jointly. For example, character recognition for different writers. *Multi-class* is the case where there is only a single classification task involving several possible labels (object classes), where the task is to assign each example a single label. For *Knowledge-Transfer* [56], one typically assumes that k related models have already been learnt, and these can be used to enhance the learning of a new related task that may come with too few training examples.

Intuitively the more data and tasks there are, the more one can benefit from sharing information between them. Typically, a single level of sharing is considered [40, 41, 43, 44, 47, 48, 49, 54]. However, applying such approaches to datasets with many diverse tasks, focus the learning on shared information among **all** tasks, which might miss out on some relevant information shared between a smaller group of tasks. In many situations, the basic intuition is that it would be beneficial to be able to share a lot of information with a few related objects, while sharing less information with a larger set of less related objects. For example, we may encourage modest information sharing between a wide range of recognition tasks, such as all road vehicles and separately seek more active sharing among related objects, such as all types of trucks. Following this, intuition hierarchical approaches to information sharing have been proposed [7, 8, 9, 10, 11, 12], see discussion in Section 1.2.4.

1.2.1 Multi-Task Learning

In the multi-task learning setting several individual tasks are trained jointly. It has been shown to improve the generalization performance of the individual tasks learnt in parallel. Typically, the algorithmic and theoretical analysis of multi-task learning, deals with a two-level structure: the group of all tasks and the level of individual tasks. When designing a multi-task approach the main question is how the information is shared. Popular approaches include neural networks, Bayesian methods and manipulating a parameter's matrix structure.

Neural Networks A natural approach to multi-task learning is to consider an explicit structure in the form of a neural-network with shared nodes [38, 39]. Consider a neural-network, trained for each of the individual tasks. Each such network

consists of an input layer, hidden layers and an output node. By unifying neural-networks of the individual tasks into a single neural-network with the same input layer (assuming the tasks are defined over the same feature space), shared hidden layers and an output layer with a single node for each of the tasks, we obtain a multi-task learning approach. Training such a network, using the back-propagation algorithm for instance, will cause updates originated by each of the nodes in the output layer (each corresponding to a single task) to jointly effect the hidden layers.

Bayesian Methods Bayesian methods are typically applied to the multi-task learning problem by considering a hierarchical Bayesian approach with 2 or more layers. The bottom layers correspond to the individual aspects of each of the tasks where the top layers correspond to the commonalities among the tasks and act as a prior over the parameters of the individual tasks [42, 50]. Learning the prior is part of the training process and is thus influenced by all tasks.

Matrix structure manipulation Another popular approach to formalize the joint learning problem is to consider a matrix of parameters $\mathbf{W}$ being learnt where, each column (or row) corresponds to the parameters of a single task. Then, by manipulating the structure of this matrix, information can be shared. In [44] the traditional Frobenius-norm used to regularize $\mathbf{W}$, is replaced with the trace-norm. The trace-norm regularization of $\mathbf{W}$ is known to be equivalent to a matrix decomposition, $\mathbf{W}=\mathbf{FG}$ where $\mathbf{F}$ and $\mathbf{G}$ are regularized using the Frobenius-norm [57]. $\mathbf{F}$ corresponds to the common features and the columns of $\mathbf{G}$ correspond to the individual task classifiers in the shared features space.

A common approach to information sharing based on manipulating the matrix structure, is to enforce joint feature selection [41, 49, 54]. In [41, 49] the regularization term -$||\mathbf{W}||_{1,2} = \sum_{j=1}^{\mathbf{n}} ||\mathbf{w}_j||_2$ is used. This term induces feature sparsity, while favoring feature sharing between all tasks [41]. A different approach for joint feature selection is taken by the ShareBoost algorithm [54], where a very small set of features is found by a forward greedy selection approach, guided by a joint loss function. It is shown that the resulting predictor uses few shared features (if such a predictor exists) and that it has a small generalization error. In [47] a feature learning approach is proposed where both a shared feature space and task specific

weights are learnt. The shared feature space is learnt by using the $||\mathbf{W}||_{1,2}$ regularization term, where $\mathbf{W}$ represents the matrix of task weights (column per task), thus encouraging the feature learning process to learn a sparse representation shared among all tasks.

1.2.2 Knowledge-Transfer

In the knowledge-transfer setting we assume k related models have already been learnt and these can be used to enhance the learning of a new related task. This is a natural scenario in learning systems where we cannot assume we know or can deal with, all learning tasks in advance. When a new task appears it might appear with too few training examples. In such a scenario, we would like to be able to bootstrap the learning process given our previous learnt models. The motivation is that learning a new model should be easier, given the previously learnt models, than learning it from scratch [58].

One of the main challenges in knowledge transfer is answering the question of 'what information to transfer?'. The typical answer can be categorized roughly into 3 categories: instance transfer, feature representation transfer or parameter transfer.

Instance Transfer In the instance transfer approach, available data (e.g. training instances of pre trained tasks or auxiliary unlabeled samples), is used for the training of the novel model. An exemplar based approach is presented in [48], where a new task is trained using a sparse set of prototype samples, chosen during an earlier learning stage. A boosting approach applying instance transfer is presented in [45], where a sample set from a different distribution from that of the novel task, is used in conjunction with the novel task data by using a weighting scheme, reflecting how "good" these samples are for the novel class. In [7] we propose to share instances by adding samples of closely related tasks to the training set of the novel task.

Feature Representation Transfer The feature representation approach uses a learnt feature representation as the basis of information transfer to the novel task. In [46], kernels are learnt for the known tasks and used to represent a new kernel learnt for the novel task. A dictionary learning approach is presented in [55], where

the novel task learns the weights of atoms in an existing dictionary. The dictionary is a shared dictionary, learnt previously in a joint learning setting among all known tasks.

Parameter Transfer In many scenarios one can assume that the learning problem of the novel task is defined over the same set of parameters as the original tasks. In such cases the learning process can benefit from parameter transfer. Such as in [52] and [53], where both the original and novel tasks, are cast as SVM learning problems, where the parameters defining a hyperplane are learnt. The parameters of the novel task are constrained to be close to a weighted sum of the original tasks. In [51], shape information is transferred between object classes. The shape model is parameterized as a Gaussian distribution. An existing shape model is fully transfered to a novel task by assuming that the variations within class, behave similarly for related classes such as horse and zebra, while their mean appearance is different. Following this assumption, only the covariance parameters of the pre-trained model are transferred to the novel task via a weighted combination with the novel class covariance.

A crucial consideration when sharing information among tasks, is between which tasks to share? In the knowledge-transfer literature [56] this question is usually phrased as 'When to share?'. We would like to transfer knowledge from the most relevant tasks and avoid tasks with *negative transfer* harming the learning process. One might assume that the answer is given in advance such as in our hierarchical combination approach [7] and in the shape transfer approach of [51]. If the relevance of tasks is unknown a priori we would like to be able to infer it when the new task arrives. In model parameter based transfer, this issue is typically addressed by methods such as model weighting [52], where the pre-trained models are selected by learning their relevant weights. While in instance transfer, individual samples are weighted reflecting their relevance to the distribution of samples of the new task. For example [45] proposes a boosting approach, where at each boosting round, the weights of the auxiliary samples (unlabeled or from different tasks) is adjusted to reflect their fit to the samples from the new task.

1.2.3 Theoretical Analysis

Theoretical studies in this field focus on the reduction of the sample complexity of a single task given other related tasks. [39] analyzes the potential gain under the assumption that all tasks share a common inductive bias, reflected by a common, near-optimal hypothesis class. [59] formulates a specific notion of task relatedness, as a family of transformations of the sample generating distributions of the tasks. This model enables the separation between information which is invariant under these transformations, to information which is sensitive to these transformations, thus the potential gain in multi-task learning lies in jointly learning the invariances.

For example, [59] shows that for certain learning tasks, e.g. when the family of hypotheses is the set of rectangles in R^d, it is beneficial to decompose the problem into two subproblems, first learning task invariant aspects of the problem, and then learning the specific task aspects. Such a decomposition can be beneficial in small sample scenarios where the original learning problem cannot be learnt given the set of samples from the original task, but it can be learnt, provided that there are enough samples from the related tasks to learn the tasks invariants. In addition, the subproblem of learning only the tasks' specific aspects should be an easier learning problem, entailing a smaller hypothesis search space.

1.2.4 Hierarchical Sharing

Motivated by the potential gain in considering a diverse set of relations among tasks, where one would like to share a lot of information with a few related tasks, while sharing less information with a larger set of less related tasks, several hierarchical approaches to information sharing have been proposed. The methods can be roughly divided into two categories, assuming the hierarchy is known and assuming that it is unknown and thus trying to infer it from the data.

Known hierarchy Typically, a known hierarchy can be *"borrowed"* from different domains, such as using a semantic hierarchy when learning visual content [7, 9]. In our earlier work [7] we showed that visual categories related by a hand-labeled semantic hierarchy can share information and significantly improve the classification results of the most specific classes in the hierarchy. Zhao et al. [9] considered a large

scale categorization setting, where they use the structure of the WordNet semantic hierarchy [60], to guide the grouping of a hierarchical group lasso learning approach. Using such a semantic hierarchy has proven to be beneficial, but we cannot always assume it is available or fits the learning domain at hand.

Unknown hierarchy Previous work investigating the sharing of information when the hierarchy is unknown, attempted to find it explicitly by solving the difficult problem of clustering the tasks into sharing groups [8, 10, 11, 12]. Such clustering approaches either solve difficult problems such as the integer programing formulation of [10] which cannot scale up to large datasets or approximate the solutions by considering heuristics [8]. Two stage approaches such as the tree-guided group lasso [12], separate the learning stage from the clustering stage by first finding the grouping of the tasks and then learning, given the discovered groupings. Such approaches still deal with the hard problem of clustering whereas by separating the clustering from the learning stage it is not clear if the clustering provides the best grouping for the given learning task.

1.3 Novelty Detection

What happens when the future is not similar to the present? Machine learning algorithms are typically trained on learning tasks reflecting the current knowledge and available data. Natural organisms, on the other hand, especially developing ones, such as babies, readily identify new unanticipated stimuli and situations, and frequently generate an appropriate response.

By definition, an unexpected event is one whose probability to confront the system is low, based on the data that has been previously observed. In line with this observation, much of the computational work on novelty detection focuses on the probabilistic modeling of known classes, identifying outliers of these distributions as novel events (see e.g. [61, 62] for detailed surveys).

Previous work may estimate a spherically shaped boundary around a single class data set [63], learn a hyper-plane which separates the class data set from the rest of the feature space (one class SVM) [64], or utilize the non parametric Parzen-window density estimation approach [65]. A few methods use a multi-class discriminative

approach, as for example for the detection of novel objects in videos [66] and for the specific task of face verification [67]. To our knowledge, all novelty detection approaches, which do not rely on samples of outliers or otherwise model the outliers distribution, detect novelty by rejecting normality (i.e., novelty is detected when all classifiers of known objects fail to accept a new sample).

In the context of object class recognition, a novel event can be a sample from a novel subclass. For example, when a sample of a collie dog is presented to a system which is trained to detect only poodles and German shepherds. In such a case, we would like to be able to classify the sample as belonging to a novel object class, unknown to the system. Such a response carries much more semantics than a simple rejection, enabling the system to adapt appropriately and start learning the novel class. Existing rejection based novelty detection approaches do not naturally handle this case.

1.4 Overview and Outline

When considering a hierarchical approach, the first important question we need to answer is: 'What is the *'correct'* hierarchy?'. We address this question by considering both the setting where we can assume the hierarchy is known, (Chapter 2 and Chapter 3) as well as the setting where it is unknown prior to the learning stage (Chapter 4 and Chapter 5). To deal with the case where the hierarchy is unknown we take two different approaches: an explicit and an implicit approach, Chapter 4 and Chapter 5 respectively. In the explicit approach we discover the hierarchy from the data, whereas in the implicit approach we are able to exploit the hierarchical relations among tasks without having to find the hierarchy explicitly.

Our hierarchical methods are applied to the novel subclass detection and information sharing applications:

Novel Subclass Detection In comparison with existing work on novelty detection, an important characteristic of our approach, presented in Chapter 2, is that we look for a level of description where the novel event is sufficiently probable. Rather than simply respond to an event which is rejected by all classifiers, which often requires no special attention (as in pure noise), we construct and exploit a hierarchy

of representations. We attend to those events which are recognized (or accepted) at some more abstract levels of description in the hierarchy, while being rejected by the classifiers at the more specific (concrete) levels.

Information Sharing When considering large scale recognition tasks where many objects exist, we can no longer assume that all tasks share the same relations. We would like to be able to capture the diversity of relations among the tasks, thus exploiting the potential of shared information in the data to its maximum. Following this motivation, we take a hierarchical approach, where we seek to share a small amount of information between a large group of tasks, while sharing a large amount of information with a small groups of tasks. We developed hierarchical information sharing algorithms dealing with the joint learning (multi-task/class) and knowledge-transfer settings. Our work focuses both on a theoretical analysis seeking a better understanding of under what conditions hierarchical sharing can be beneficial, as well as designing practical algorithms which can scale up to large scale recognition settings.

Our two approaches to hierarchical sharing are presented in Chapters 3 and 5. In the first algorithm, chapter 3, we assume the hierarchy is known and is based on a cascade of learning problems, where the joint learning is applied based on an explicit grouping of tasks, defining new tasks which correspond to the hierarchical organization of tasks. We define a hierarchical multi-task learning framework and provide generalization bounds.

Our second algorithm, presented in chapter 5, encourages hierarchical sharing in an implicit manner when the hierarchy is unknown; it is efficient and is designed to scale up to large scale settings. We formalize the joint learning problem using a matrix of parameters, where each column corresponds to the set of parameters of a single task. Instead of enforcing an explicit structure over this matrix we enforce implicitly hierarchical sharing by utilizing a cascade of sparsity regularization terms. We thereby avoid the hard problem of clustering in the existing hierarchal approaches to information sharing [8, 10, 11, 12]. This method is also extended to the knowledge-transfer setting, based on the structure of shared information discovered during the joint learning phase. Two knowledge-transfer algorithms are proposed. First we consider a parameter transfer approach where the novel task is regularized to share

parameters with the original task. In the second algorithm we apply a feature representation transfer, where we learn a dimensionality reduction guided by the parameters, learnt using our regularization cascade for training the original tasks, jointly.

Both our approaches are based on a cascade of learning problems, which are inherently different. Our explicit approach in Chapter 3 is based on a cascade of loss functions while our implicit approach in Chapter 5 is based on a cascade characterized by the regularization term.

In the following we give a short description of each of the chapters in this thesis:

Chapter 2 We present our approach to novel subclass detection, where we are able to classify a sample as belonging to a novel object class, unknown during the training phase. Our method is based on a hierarchical organization of the object classes which is known during the training phase. We present the incongruence detection framework which declares novelty, by first looking for an accepting level in the hierarchy, unlike common approaches to novelty, which declare novelty based on rejection by all known models.

Publication: Daphna Weinshall, Alon Zweig, Hynek Hermansky, Stefan Kombrink, Frank W. Ohl, Jeorn Anemeuller,Jeorg-Hendrik Bach, Luc Van Gool, Fabian Nater, Tomas Pajdla, Michal Havlena and Misha Pavel. **Beyond Novelty Detection: Incongruent Events, when General and Specific Classifiers Disagree.** *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 34(10):1886-1901, Oct 2012*

Chapter 3 We propose a general framework for a full hierarchical multi-task setting. We define an explicit notion of hierarchical tasks' relatedness, where at each level we assume that some aspects of the learning problem are shared. We suggest a cascade approach, where at each level of the hierarchy, a learner learns jointly the uniquely shared aspects of the tasks by finding a single shared hypothesis. This shared hypothesis is used to bootstrap the preceding level in the hierarchy, forming a hypothesis search space. We analyze sufficient conditions for our approach to reach optimality and provide generalization guarantees, in an empirical risk minimization setting.

Publication: Alon Zweig and Daphna Weinshall. **Hierarchical Multi-Task Learning: a Cascade Approach Based on the Notion of Task Relatedness.** *Theoretically Grounded Transfer Learning workshop, held at International Conference on Machine Learning (ICML), June 2013*

Chapter 4 Unlike chapters 2 and 3 where we assume the hierarchy is known, in this chapter we present a general hierarchical model of tasks and an algorithm to infer it from data. The model captures both part-membership (conjunctive) and class-membership (disjunctive) hierarchies. We provide an algorithm for discovering hierarchical structure in data. We start by assuming a binary set representation of concepts (tasks). Each task is represented by a unique set of properties. Then, we extend the model to a statistical setting.

Chapter 5 We present a hierarchical approach for information sharing among different classification tasks, in multi-task, multi-class and knowledge-transfer settings. We propose a top-down iterative method, which begins by posing an optimization problem with an incentive for large scale sharing, among all classes. This incentive to share is gradually decreased, until there is no sharing and all tasks are considered separately. The method therefore exploits different levels of sharing within a given group of related tasks, without having to make hard decisions about the grouping of tasks.

In order to deal with large scale problems, with many tasks and many classes, we extend our batch approach to online setting and provide regret analysis of the algorithm. Based on the structure of shared information discovered in the joint learning settings, we propose two different knowledge-transfer methods for learning novel tasks. The methods are designed to work within the very challenging large scale settings. We tested our methods extensively on synthetic and real datasets, showing significant improvement over baseline and state-of-the-art methods.

Publication: Alon Zweig and Daphna Weinshall. **Hierarchical Regularization Cascade for Joint Learning.** *In Proceedings of 30th International Conference on Machine Learning (ICML), Atlanta GA, June 2013*

2

Novel Subclass Detection

Say we are an American dog expert who can identify any American dog breed instantly from a distance, such as the American Bulldog or Boston Terrier, but we never saw the Australian Shepherd. How shall we react when we encounter the Australian Shepherd for the first time? shall we run away hoping to avoid its claws? cover our nose to avoid the stink? A typical curios human dog lover, will detect the Australian shepherd as a dog, and react accordingly, such as asking the owner if the dog can be pet. In case of a novel event we would like to be able to fit our response as close as possible to the phenomena we encounter, avoiding claws or stink are not normal reactions to dogs.

In this chapter we are motivated by the task of detecting a novel subclass as such. Thus, enabling the system to react appropriately. This task is a specific type of novelty detection. For which the traditional approach to declare novelty is rejection of what is known [61, 62]. Rejection alone does not carry enough semantics and will not enable a dog expert to respond appropriately in case of an encounter with a new dog (see discussion in 1.3).

We view the subclass detection problem from the hierarchical point and formulate a two stage approach, where we first look for a semantic level which accepts and only then look for a rejection. For this we formulate the problem using the general incongruent events framework we presented in [68].

The rest of this chapter is organized as follows. In Section 2.1 we introduce the incongruent events framework. In section 2.2 we introduce the specific algorithm for the novel subclass detection task and in section 2.3 we describe an experiment showing promising results on two different data sets.

2.1 Incongruent Events Framework

We now define the concept of incongruent events as induced by a classification process that can, in general, correspond to partial order on sets of event classes. This

partial order, represented by a directed graph (DAG) that captures subset-superset relations among sets (as in algebra of sets) can also be interpreted in terms of two category-forming operations: conjunctive and disjunctive hierarchies as described in Section 2.1.1. We provide a unified definition in Section 2.1.2, and analyze its implication for the two cases.

2.1.1 Label Hierarchy and Partial Order

The set of labels (or concepts) represents the knowledge base about the stimuli domain, which is either given (by a teacher) or learnt. In cognitive systems such knowledge is hardly ever a set; often, in fact, labels are given (or can be thought of) as a hierarchy.

In general, a hierarchy can be represented by a directed graph, where each label (a set of objects) corresponds to a single node in the graph. A directed edge exists from label (specific child concept) a to (general parent concept) b, iff a corresponds to a smaller set of events or objects in the world, which is contained in the set of events or objects corresponding to label b, i.e., $a \subset b$. In this way the edges represent a partial order defined over the set of labels or concepts.

Because the graph is directed, it defines for each concept a two distinct sets of concepts (parent-child) related to it: *disjunctive concepts* which are smaller (subsets) according to the partial order, i.e. they are linked to node a by incoming edges converging on a; and *conjunctive concepts* which are larger (supersets) according to the partial order, i.e. they are linked to node a by outgoing edges diverging from a. If the DAG of partial order is a tree, only one of these sets is non trivial (larger than 1).

We consider two possible tree-like hierarchies, which correspond to two interesting intuitive cases:

Conjunctive Hierarchy. Modeling part membership, as in biological taxonomy or speech. For example, eyes, ears, and nose combine to form a head; head, legs and tail combine to form a dog (see left panel of Fig. 2.1); and sequences of phonemes constitute words and utterances. In this case, each node has a single child and possibly many parents.

Disjunctive Hierarchy. Modeling class membership, as in human categoriza-

tion – where objects can be classified at different levels of generality, from sub-ordinate categories (most specific level), to basic level (intermediate level), to super-ordinate categories (most general level). For example, a Beagle (sub-ordinate category) is also a dog (basic level category), and it is also an animal (super-ordinate category), see right panel of Fig. 2.1. In this case, each node has a single parent and possibly many children.

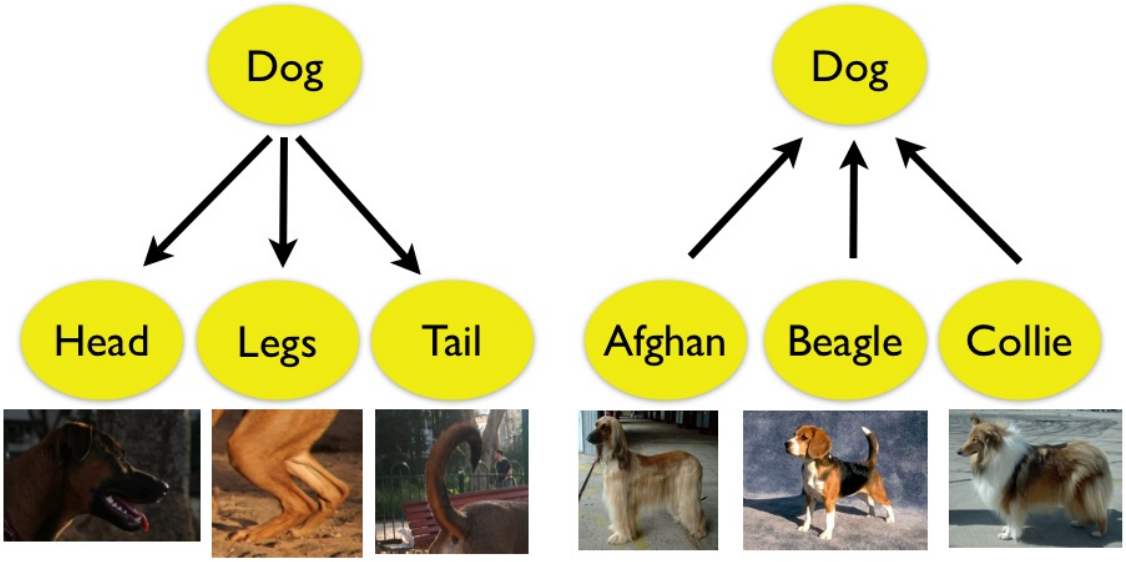


Figure 2.1: Examples. Left: *Conjunctive hierarchy,* the concept of a dog requires the conjunction of parts, including head, legs and tail. Right: *Disjunctive hierarchy,* the concept of a dog is defined as the disjunction of more specific concepts, including Afghan, Beagle and Collie.

The sets of *disjunctive* and *conjunctive* concepts induce constraints on the observed features in different ways. Accordingly, the set of objects corresponding to a given label (node) is *included* in the *intersection* of the objects in its set of *conjunctive concepts*. Thus in the example shown in the left panel of Fig. 2.1, the concept of *Dog* requires the conjunction of parts as in $DOG \subseteq LEGS \cap HEAD \cap TAIL$. To the contrary, the set of objects corresponding to a given label (node) *contains* the *union* of objects in its set of *disjunctive concepts*. In the example shown in the right panel of Fig. 2.1, the class of dogs requires the disjunction of the individual members as in $DOG \supseteq AFGHAN \cup BEAGEL \cup COLLIE$.

2.1.2 Definition of Incongruent Events

2.1.2.1 Multiple Probabilistic Models for Each Concept

For each node a, define $A^s = \{b \in G,\ b \preceq a\}$ - the set of *disjunctive concepts*, corresponding to all nodes more specific (smaller) than a in accordance with the given partial order. Similarly, define $A^g = \{b \in G,\ a \preceq b\}$ - the set of *conjunctive concepts*, corresponding to all nodes more general (larger) than a in accordance with the given partial order.

For each node a and training data $\mathcal{T}$, we hypothesize 3 probabilistic models which are derived from $\mathcal{T}$ in different ways, in order to determine whether a new data point X can be described by concept a:

- $Q_a(X)$: a probabilistic model of class a, derived from training data $\mathcal{T}$ unconstrained by the partial order relations in the graph.

- $Q_a^s(X)$: a probabilistic model of class a which is based on the probability of concepts in A^s, assuming their independence of each other. Typically, the model incorporates a simple *disjunctive* relation between concepts in A^s.

- $Q_a^g(X)$: a probabilistic model of class a which is based on the probability of concepts in A^g, assuming their independence of each other. Here the model typically incorporates a simple *conjunctive* relation between concepts in A^g.

2.1.2.2 Examples

To illustrate, consider again the simple examples shown in Fig. 2.1, where our concept of interest a is *Dog*.

In the Conjunctive hierarchy (left panel), $|A^g| = 3$ (Head, Legs, Tail) while $|A^s| = 1$. We derive two different models for the class *Dog*:

1. Q_{Dog} - obtained using training pictures of *dogs* and *not dogs* without body part labels.

2. Q_{Dog}^g - obtained using the outcome of models for Head, Legs and Tail, which have been derived from the same training set $\mathcal{T}$ with body part labels only. If

we further assume that concept a is the conjunction of its part member concepts as defined above, and assuming that these part concepts are independent of each other, we get

$$Q^g_{\text{Dog}} = \prod_{b \in A^g} Q_b = Q_{\text{Head}} \cdot Q_{\text{Legs}} \cdot Q_{\text{Tail}} \tag{2.1}$$

In the disjunctive hierarchy (right panel), $|A^s| = 3$ (Afghan, Beagle, Collie) while $|A^g| = 1$. We therefore derive two models for the class *Dog*:

1. Q_{Dog} - obtained using training pictures of *dogs* and *not dogs* without breed labels.

2. Q^s_{Dog} - obtained using the outcome of models for Afghan, Beagle and Collie, which have been derived from the same training set $\mathcal{T}$ with dog breed labels only. If we further assume that class a is the disjunction of its sub-classes as defined above, and once again assume that these sub-classes are independent of each other, we get

$$Q^s_{\text{Dog}} = \sum_{b \in A^s} Q_b = Q_{\text{Afghan}} + Q_{\text{Beagle}} + Q_{\text{Collie}}$$

2.1.2.3 Incongruent Events

In general, we expect the different models to provide roughly the same probabilistic estimate for the presence of concept a in data X. A mismatch between the predictions of the different models may indicate that something new and interesting is being observed, unpredicted by the existing knowledge of the system. In particular, we are interested in the following discrepancy:

Definition: *Observation X is* incongruent *if there exists a concept $'a'$ such that*

$$Q^g_a(X) \gg Q_a(X) \quad \text{or} \quad Q_a(X) \gg Q^s_a(X). \tag{2.2}$$

In other words, observation X is *incongruent* if a discrepancy exists between the inference of two classifiers, where the more general classifier is much more confident in the existence of the object than the more specific classifier.

Classifiers come in different forms: they may accept or reject, they may generate a (possibly probabilistic) hypothesis, or they may choose an action. For binary classifiers that either accept or reject, the definition above implies one of two mutually exclusive cases: either the classifier based on the more general descriptions from level g accepts X while the direct classier rejects it, or the direct classifier accepts X while the classifier based on the more specific descriptions from level s rejects it. In either case, the concept receives high probability at some more general level (according to the partial order), but much lower probability when relying only on some more specific level.

2.2 Algorithm

We now adopt the framework described above to the problem of novel class detection, when given a *Disjunctive Hierarchy*. We assume a rich hierarchy, with non trivial (i.e. of size larger than 1) sets of *disjunctive concepts*, see right panel of Fig. 2.1. This assumption enables the use of discriminative classifiers.

As discussed in Section 2.1.2 and specifically in the second example there, in a *disjunctive hierarchy* we have two classifiers for each label or concept: the more general classifier $Q_{concept}$, and the specific disjunctive classifier $Q^s_{concept}$. The assumed classification scenario is multi-class, where several classes are already known.

In order to identify novel classes, our algorithm detects a discrepancy between $Q_{concept}$ and $Q^s_{concept}$. The classifier $Q_{concept}$ is trained in the usual way using all the examples of the object, while the specific classifier $Q^s_{concept}$ is trained to discriminatively distinguish between the concepts in the set of *disjunctive concepts* of the object.

Algorithm 1 is formally described in the box below.

We tested the algorithm experimentally on two visual recognition datasets. We found that discriminative methods, which capture distinctions between the related known sub-classes, perform significantly better than generative methods. We demonstrate in our experiments the importance of modeling the hierarchical relations tightly. Finally, we compare the performance of the proposed approach to results obtained from novelty detection based on one-class SVM outlier detection.

Algorithm 1 Unknown Class Identification

Input :

$\mathbf{x}$ test image

C^g general level classifier

C_j specific level classifiers, $j = 1..|\text{known sub-classes}|$

$V^c_{C_i}$ average certainty of train or validation examples classified correctly as C_i

$V^w_{C_i}$ average certainty of train or validation examples classified wrongly as C_i (zero if there are none).

1. Classify $\mathbf{x}$ using C^g
2. **if** accept
   Classify $\mathbf{x}$ using all C_j classifiers and obtain a set of certainty values $V_{C_j}(\mathbf{x})$
   Let $i = \arg\max_j V_{C_j}(\mathbf{x})$
   Define $S(\mathbf{x}) = (V_{C_i}(\mathbf{x}) - V^w_{C_i})/(V^c_{C_i} - V^w_{C_i})$
   (a) **if** $S(\mathbf{x}) > 0.5$
   label $\mathbf{x}$ as belonging to a known class
   (b) **else** label $\mathbf{x}$ as belonging to a novel (unknown) class
3. **else** label $\mathbf{x}$ as a background image

Basic Object Class Classifiers To verify the generality of our approach, we tested it using two different embedded object class representation methods. For conciseness we only describe results with method [69]; the results with method [70] are comparable but slightly inferior, presumably due to the generative nature of the method and the fact that it does not use negative examples when training classifiers.

The object recognition algorithm of [69] learns a generative relational part-based object model, modeling appearance, location and scale. Location and scale are described relative to some object location and scale, as captured by a star-like Bayesian network. The model's parameters are discriminatively optimized (given

negative examples during the training phase) using an extended boosting process. Based on this model and some simplifying assumptions, the likelihood ratio test function is approximated (using the MAP interpretation of the model) by

$$F(\mathbf{x}) = \max_{C} \sum_{k=1}^{P} \max_{u \in Q(\mathbf{x})} \log\, p(u|C, \theta^k) - \nu \tag{2.3}$$

with P parts, threshold ν, C denoting the object's location and scale, and $Q(\mathbf{x})$ the set of extracted image features.

General Category Level Classifier In order to learn the general classifier $Q_{concept}$, we consider all the examples from the given sub-classes as the positive set of training examples. For negative examples we use either clutter or different unrelated objects (none of which is from the known siblings). As we shall see in Section 2.3, this general classifier demonstrates high acceptance rates when tested on the novel sub-classes.

Specific Category Level Classifier The problem of learning the specific classifier $Q^s_{concept}$ is reduced to the standard novelty detection task of deciding whether a new sample belongs to any of the known classes or not. However, the situation is somewhat unique and we take advantage of this: while there are multiple known classes, their number is bounded by the degree of the graph of partial orders (they must all be sub-classes of a single abstract object). This suggests that a discriminative approach could be effective. The algorithm is formally described in the next section.

2.2.1 Algorithm for Sub-Class Detection

The discriminative training procedure of the specific level classifier is summarized in Algorithm 2 with details subsequently provided.

Step 1, Discriminative Multi-Class Classification The specific level object model learnt for each known class is optimized to separate the class from its siblings. A new sample $\mathbf{x}$ is classified according to the most likely classification (max decision)

Algorithm 2 Train Known vs. Unknown Specific Class Classier

1. For each specific class, build a discriminative classifier with:
   positive examples: all images from the specific class
   negative examples: images from all sibling classes

2. Compute the Normalized Certainty function, and choose classification threshold for novel classes.

3. Accept iff the normalized certainty is larger than the fixed threshold.

C_i. The output of the learnt classifier (2.3) provides an estimate for the measure of classification certainty $V_{C_i}(\mathbf{x})$.

Step 2, Normalized Certainty Score Given $V_{C_i}(\mathbf{x})$, we seek a more sensitive measure of certainty as to whether or not the classified examples belong to the group of known sub-classes. To this end, we define a normalized score function, which normalizes the certainty estimate $V_{C_i}(\mathbf{x})$ relative to the certainty estimates of correct classification and wrong classification for the specific-class classifier, as measured during training or validation.

Specifically, let $V_{C_i}^c$ denote the average certainty of train or validation examples classified correctly as C_i, and let $V_{C_i}^w$ denote the average certainty of train or validation examples from all other sub-classes classified wrongly as belonging to class C_i. The normalized score $S(\mathbf{x})$ of $\mathbf{x}$ is calculated as follows:

$$S(\mathbf{x}) = \frac{(V_{C_i}(\mathbf{x}) - V_{C_i}^w)}{(V_{C_i}^c - V_{C_i}^w)} \tag{2.4}$$

If the classes can be well separated during training, that is $V_{C_i}^c >> V_{C_i}^w$ and both groups have low variance, the normalized score provides a reliable certainty measure for the multi-class classification.

Step 3, Choosing a threshold Unlike the typical learning scenario, where positive (and sometimes negative) examples are given during training, in the case of novelty detection no actual positive examples are known during training (since by

definition novel objects have never been observed before). Thus it becomes advantageous to set more conservative limits on the learnt classifiers, more so than indicated by the training set. Specifically, we set the threshold of the normalized certainty measure, which lies in the range [0..1], to 0.5.

2.3 Experiments

Datasets We used two different hierarchies in our experiments. In the first hierarchy, the general parent category level is the 'Motorbikes', see Fig. 2.2. 22 object classes, taken from [71], were added, in order to serve together with the original data set as a joint pool of object classes from which the unseen-objects are sampled. 22 object classes. In the second hierarchy, the general parent category level is the 'Face' level, while the more specific offspring levels are faces of six different individuals, see Fig. 2.3.

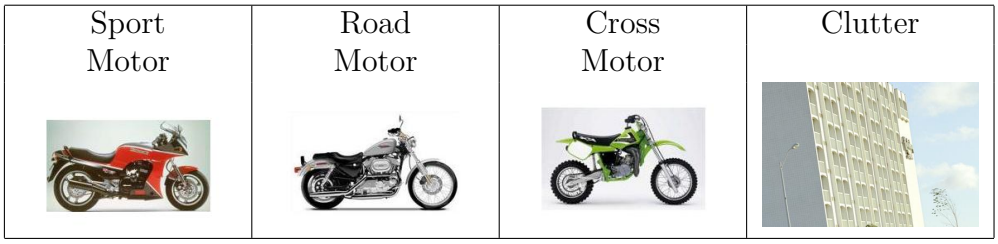


Figure 2.2: Examples from the object classes and clutter images, used to train and test the different Category level models of the 'Motorbikes' hierarchy. The more specific offspring levels are: 'Sport-Motorbikes', 'Road-Motorbikes' and 'Cross-Motorbikes'. These images are taken from the Caltech-256 [71] dataset. Clutter images are used as negative examples.

Method All experiments were repeated at least 25 times with different random sampling of test and train examples. We used 39 images for the training of each specific level class in the 'Motorbikes' hierarchy, and 15 images in the 'Faces' hierarchy. For each dataset with n classes, n conditions are simulated, leaving each of the classes out as the unknown (novel) class.

Basic Results Fig. 2.4 shows classification rates for the different types of test examples: *Known* - new examples from all known classes during the training phase;

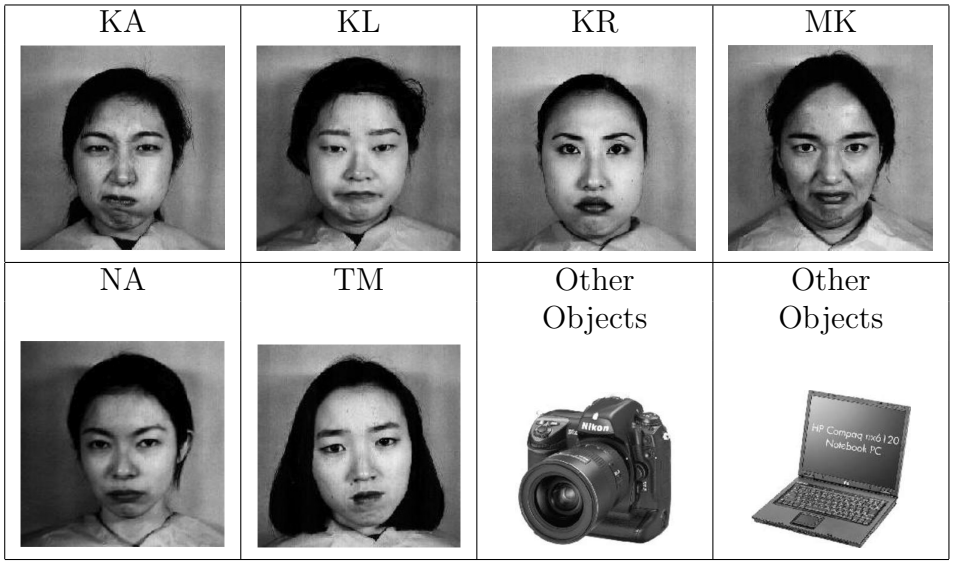


Figure 2.3: Examples from the object classes in the 'Faces' hierarchy, taken from [72]. A mixture of general object images were used as negative examples.

Unknown - examples from the unknown (novel) class which belong to the same General level as the Known classes but have been left out during training; *Background* - examples not belonging to the general level which were used as negative examples during the General level classifier training phase; and *Unseen* - examples of objects from classes not seen during the training phase, neither as positive nor as negative examples. The three possible types of classification are: *Known* - examples classified as belonging to one of the known classes; *Unknown* - examples classified as belonging to the unknown class; and *Background* - examples rejected by the General level classifier.

The results in Fig. 2.4 show the desired effects: each set of examples - Known, Unknown and Background, has the highest rate of correct classification in its own category. As desired, we also see similar recognition rates (or high acceptance rates) of the Known and Unknown classes by the general level classifier, indicating that both are regarded as similarly belonging to the same general level. Finally, examples from the Unseen set are rejected correctly by the general level classifier.

Discriminative Specific Classifiers Improve Performance We checked the importance of using a discriminative approach by comparing our approach for build-

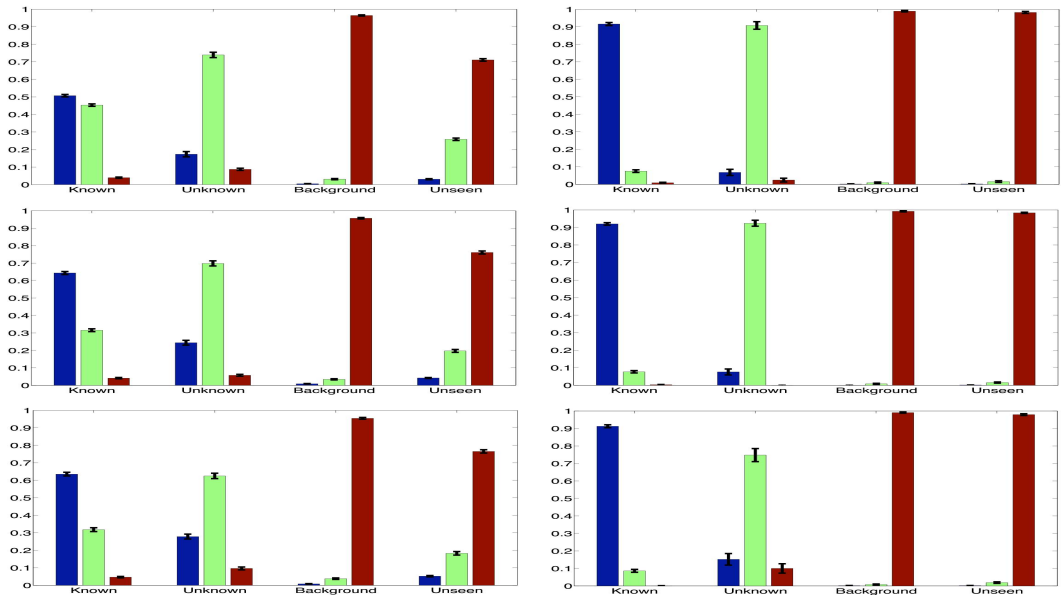


Figure 2.4: Classification ratios for 4 groups of samples: Known Classes, Unknown Class, Background and sample of unseen classes. Bars corresponding to the three possible classification rates are shown: left bar (blue) shows the known classification rate, middle bar (green) shows the unknown classification rate, and right bar (red) shows the background classification rate (rejection by the general level classifier). The left column plots correspond to the Motorbikes general level class, where the Cross (top), Sport (middle) and Road Motorbikes (bottom) classes are each left out as the unknown class. The right column plots are representative plots of the Faces general level class, where KA (top), KL (middle) and KR (bottom) are each left out as the unknown class.

ing discriminative specific-level classifiers to non-discriminative approaches. Note that the general level classifier remains the same throughout.

We varied the amount of discriminative information used when building the specific level classifiers, by choosing different sets of examples as the negative training set: 1) *1vsSiblings - Exploiting knowledge of sibling relations*; the most discriminative variant, where all train examples of the known sibling classes are used as the negative set when training each specific known class classifier. 2) *1vsBck - No knowledge of siblings relations*; a less discriminative variant, where the negative set of examples is similar to the one used when training the general level classifier.

Results are given in Fig. 2.5, showing that discriminative training with the sibling classes as negative examples significantly enhances performance.

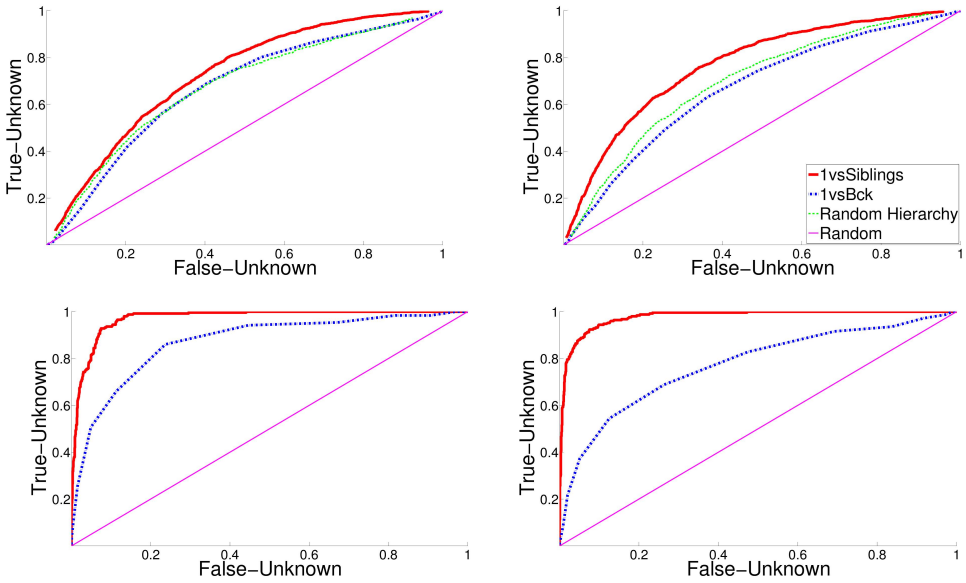


Figure 2.5: ROC curves showing True-Unknown classification rate on the vertical axis vs. False-Unknown Classification rate on the horizontal axis. We only plot examples accepted by the General level classifier. *1vsSiblings* denotes the most discriminative training protocol, where specific class object models are learnt using the known siblings as the negative set. *1vsBck* denotes the less discriminative training protocol where the set of negative examples is the same as in the training of the General level classifier. Random Hierarchy denotes the case where the hierarchy was built randomly, as described in the text. The top row plots correspond to the Motorbikes general level class, where the Cross (left) and Sport Motorbikes (right) classes are each left out as the unknown class. The bottom row plots are representative plots of the Faces general level class, where KA (left) and KL (right) are each left out as the unknown class. We only show two representative cases for each dataset, as the remaining cases look very similar.

Novel Class Detector is Specific To test the validity of our novel class detection algorithm, we checked two challenging sets if images which have the potential for false mis-classification as novel sub-class: (i) low quality images, or (ii) totally unrelated novel classes (*unseen* in Fig. 2.4). For the second case, Fig. 2.4 shows that by far most unseen examples are correctly rejected by the general level classifier.

To test the recognition of low quality images, we took images of objects from known classes and added increasing amounts of Gaussian white noise to the images. With this manipulation, background images continued to be correctly rejected by the general level classifier as in Fig. 2.4, while the fraction of known objects correctly classified decreased as we increased the noise.

In Fig. 2.6 we further examine the pattern of change in the misclassification of samples from the known class with increasing levels of noise - whether a larger fraction is misclassified as an unknown object class or as background. Specifically, we show the ratio (FU-FB)/(FU+FB), where FU denotes false classification as unknown class, and FB denotes false classification as background. The higher this ratio is, the higher is the ratio of unknown class misclassifications to background misclassifications. An increase in the false identification of low quality noisy images as the unknown class should correspond to an increase in this expression as the noise increases. In fact, in Fig. 2.6 we see the opposite - this expressions decreases with noise. Thus, at least as it concerns low quality images due to Gaussian noise, our model does not identify these images as coming from novel classes.

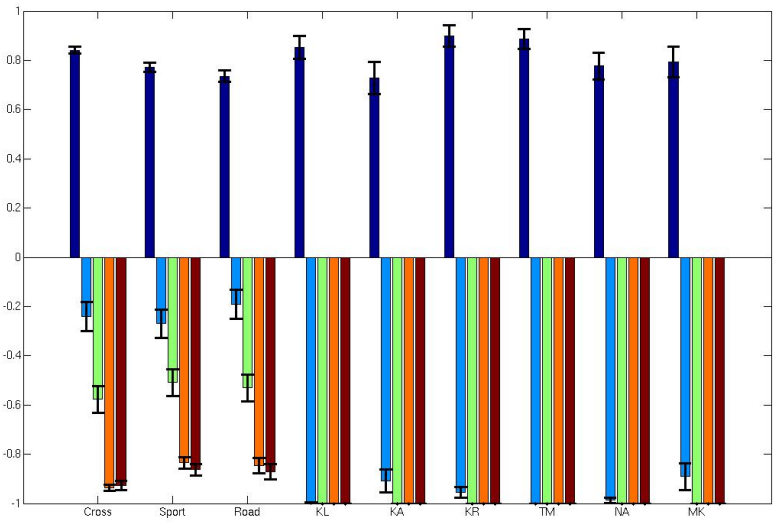


Figure 2.6: This figure shows the effect of noise on the rate of false classification of samples from known classes. Each bar shows the average over all experiments of (FU-FB)/(FU+FB). Results are shown for both the Motorbikes and Faces datasets, and each group of bars shows results for a different class left out as the unknown. In each group of bars, the bars correspond to increasing levels of noise, from the left-most bar with no noise to the fifth right-most bar with the most noise.

How Veridical Should the Hierarchy be In order to explore the significance of the hierarchy we run Algorithm 1 using different hierarchies imposed on the same set of classes, where the different hierarchies are less faithful to the actual

similarity between classes in the training set. The least reliable should be the random hierarchy, obtained by assigning classes together in a random fashion. We expect to see reduced benefit to our method as the hierarchy becomes less representative of similarity relations in the data. At the same time, if our method maintains any benefit with these sloppy hierarchies, it will testify to the robustness of the overall approach.

We therefore compared the verdical hierarchy used above to the random hierarchy, obtained by randomly putting classes together regardless of their visual similarity. As expected, our comparisons show a clear advantage to the verdical hierarchy. In order to gain insight into the causes of the decrease in performance, we analyzed separately the general and specific level classifiers. The comparison of acceptance rate by the general level classifier using the veridical hierarchy vs. random hierarchy is shown in Fig. 2.7. For examples that were accepted by the general level classifier, correct unknown classification vs. false unknown classification is shown in Fig. 2.5, for both the veridical and random hierarchy.

Results are clearly better in every aspect when using the veridical hierarchy. The performance of the learnt general level classifier is clearly better (Fig. 2.7). The distinction between known classes and the unknown class by the specific classifier is improved (Fig. 2.5). We actually see that when using a discriminative approach based on the random hierarchy, the accuracy of this distinction decreases to the level of the non discriminative approach with the veridical hierarchy. Combining both the general level classifier and the specific level classifier, clearly Algorithm 1 for the identification of unknown classes performs signnificantly better with the veridical hierarchy.

Comparison to alternative methods Novelty detection is often achieved with single class classifiers. In the experiments above we used 1vsBck classifiers as proxy to single class classifiers, and compared their performance to our approach in Fig. 2.5. In order to compare our approach to some standard novelty detection method, we chose the one class SVM [64]. Note that this is only a partial comparison, since one class SVM (like any novelty detection method which is based on rejecting the known) does not provide the distinction between novel object class and background, as we do.

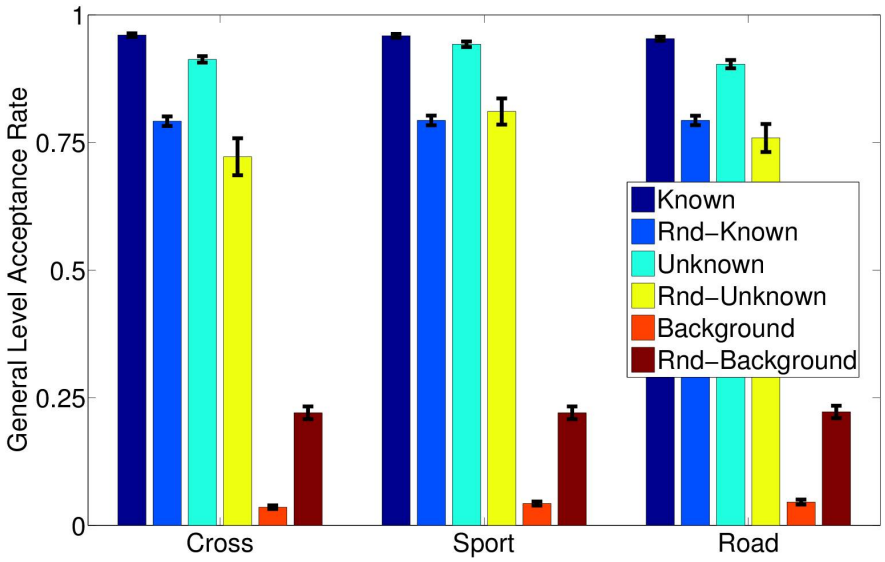


Figure 2.7: General level classifier acceptance rates, comparing the use of the verdical and random hierarchies. Six bars show, from left to right respectively: verdical hierarchy known classes ('Known'), random hierarchy known classes ('Rnd-Known'), verdical hierarchy unknown class ('Unknown'), random hierarchy unknown class ('Rnd-Unknown'), verdical hierarchy background ('Background'), random hierarchy background ('Rnd-Background'). Results are shown for the cases where the Cross-Motorbikes, Sport-Motorbikes or Road-Motorbikes are left out as the unknown class, from left to right respectively.

For technical reasons, in order to conduct this comparison we need to create a single vector representation for each instance in our dataset. To achieve this goal we followed the scheme presented in [23], describing each image by a single vector whose components are defined by the object class model of the general level class. Given this image representation we modified our algorithm and replaced the specific level classifier with a binary SVM classifier, basing the final decision on a voting scheme.

We conducted this experiment on audio-visual data collected by a single Kyocera camera with fish-eye lens and an attached microphone. In the recorded scenario, individuals walked towards the device and then read aloud identical text; we acquired 30 sequences with 17 speakers. We tested our method by choosing six speakers as members of the trusted group, while the rest were assumed unknown. The com-

parison was done separately for the audio and visual data. Fig. 2.8 shows the comparison of our discriminative approach to the one class SVM novelty detection approach using the visual data; clearly our approach achieves much better results (similar improvement was obtained when using the auditory data).

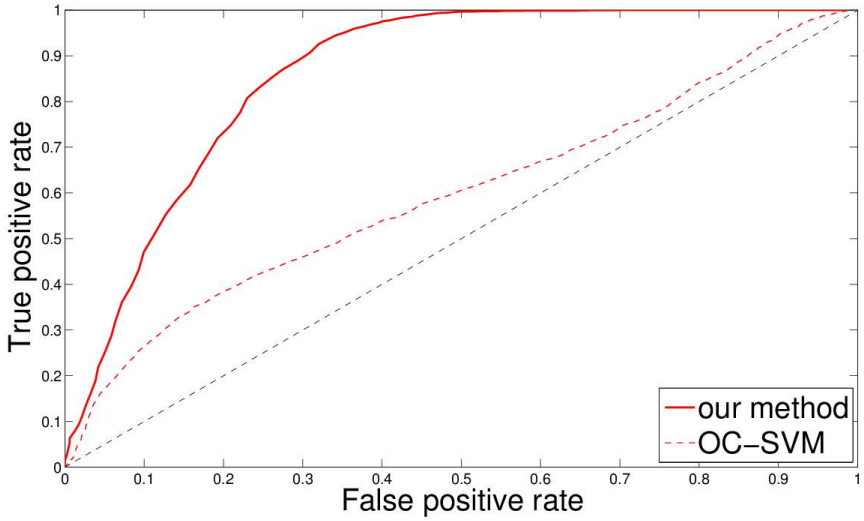


Figure 2.8: True Positive vs. False Positive rates when detecting unknown vs. trusted individuals. The unknown are regarded as positive events. Results are shown for our proposed method (solid line) and one class SVM (dashed line).

3

Hierarchical Multi-Task Learning: a Cascade Approach Based on the Notion of Task Relatedness

We propose a general framework for a full hierarchical multi-task setting. We define an explicit notion of hierarchical tasks relatedness, where at each level we assume that some aspects of the learning problem are shared. We suggest a cascade approach, where at each level of the hierarchy a learner learns jointly the uniquely shared aspects of the tasks by finding a single shared hypothesis. This shared hypothesis is used to bootstrap the preceding level in the hierarchy, forming a hypothesis search space. We analyze sufficient conditions for our approach to reach optimality, and provide generalization guarantees in an empirical risk minimization setting.

Multi-task learning is typically studied in a flat scenario, where all related tasks are similarly treated. Here we formulate a hierarchal multi-task learning setting which breaks this symmetry between the tasks, analyzing the potential gain in jointly learning shared aspects of the tasks at each level of the hierarchy. We also analyze the sample complexity of the hierarchical multi-task learning approach.

We consider a general hierarchical multi-task paradigm extending the common two level approach. We build our notion of shared information among tasks on the concept of task transformations . For this we adopt the task relatedness framework of [59] and extend it to the hierarchical setting.

The problem with considering only a single level of task relations is that the potential benefit from sharing information is derived from the size of the family of transformations which relate all the tasks. Considering a bigger and more diverse set of tasks usually means considering looser relations among those tasks, which in turn decreases the benefit obtained from having access to more tasks and additional samples. In our hierarchical setting, on the other hand, we benefit from adding tasks

even if they are related to our objective task in a rather loose manner.

To illustrate, consider the task of learning the size and location of a rectangle in R^d given a small set of samples from the target rectangle and several samples from other related rectangles (tasks). Suppose that the related tasks can be organized in a nested hierarchy, where as the level gets more abstract there are more tasks (rectangles) in it, but the number of shared dimensions gets smaller. The hierarchical ordering of tasks can be used to divide the original problem into several sub-problems, where in each level the learner learns only the shared dimensions of the tasks which are unique to that level, starting from the most abstract levels with the most tasks and least shared dimensions. The knowledge about the shared dimensions is transferred to the next level, where more dimensions are shared and fewer tasks can be used for learning. Thus we gain by dividing the original problem into easier subproblems and by adding more tasks even if this results in a looser family of transformations relating all tasks.

Motivated by the potential in hierarchically grouping tasks, where levels representing a looser notion of task relatedness correspond to many tasks and levels representing a stricter notion correspond to fewer tasks, we propose a cascade approach, where at each level of the cascade the uniquely shared aspects are learnt jointly. Compared to the original two-stage learning framework proposed for learning based on task transformations (the MT-ERM framework), our approach is more limited because at each stage of the cascade we are committed to a single hypothesis. In the original framework, the knowledge from related tasks is used to reduce the size of the search space, but no single hypothesis is singled out. However, this original approach does not give a practical way to perform shared learning of invariances. By limiting ourselves to a more restrictive setting, we are able to propose a constructive paradigm for which we can prove the optimality of the approach and derive sample complexity bounds in an Empirical Risk Minimization setting [73]. To this end we impose two assumptions - *transformation-multiplicativity* and *indifference* (see text).

The rest of this chapter is organized as follows. In Section 3.1 we review the task relatedness framework of [59] and extend it to a hierarchy of task relations. We present IMT-ERM (Iterative Multi Task-ERM), which is a hierarchical generalization of MT-ERM. We then describe and analyze a specific learning approach based

on a learning cascade, CMT-ERM (Cascade Multi Task-ERM).

The optimality analysis of a single step in the cascade is presented in Section 3.2. We describe some sufficient conditions under which the optimal shared transformation found by considering all tasks together is also optimal for a single task. In Section 3.3 we extend our analysis to the ERM setting, providing generalization error guarantees for a single stage in the hierarchy (Theorem 2). In Theorem 3 this analysis is extended recursively to the whole hierarchy. In Section 3.4 we present an experiment demonstrating the effectiveness of our approach.

3.1 Hierarchical Multi-Task Paradigm

Let us define our hierarchical multi-task paradigm. We start by reviewing the non-hierarchical multi-task paradigm in Section 3.1.1. In Section 3.1.2 we extend this paradigm to a hierarchical one. In Section 3.1.3 we describe a cascade approach to the implementation of this paradigm, and outline some important ways by which it differs from the basic approach.

3.1.1 Task Relatedness Background

We start by reviewing the multi-task learning scenario and notion of relatedness presented in [59], following the same notations and stating the relevant definitions. In this approach to multitask learning one wishes to learn a single task, and the role of additional related tasks is only to aid the learning of this task. Formally, the multi-task learning scenario can be stated as follows: Given domain $\mathcal{X}$, n tasks $1, ..., n$ and unknown distributions P_1,...P_n over $\mathcal{X} \times \{0, 1\}$, a learner is presented with a sequence of random samples S_1,...S_n drawn from these P_i's respectively. The learner seeks a hypothesis $h : \mathcal{X} \to \{0, 1\}$ such that, for (x, b) drawn randomly from P_1, $h(x) = b$ with high probability. We focus on the extent to which the samples S_i, for $i \neq 1$ can be utilized to help find a good hypothesis for predicting the labels of task 1.

Task relatedness is defined based on a set $\mathcal{F}$ of transformations $f : \mathcal{X} \to \mathcal{X}$ (following definitions 1 and 2 in [59]). Tasks 1 and 2 are said to be $\mathcal{F}$-related if $P_1(x, b) = P_2(f(x), b)$ or $P_2(x, b) = P_1(f(x), b)$. Given a hypothesis space $\mathbb{H}$ over

domain $\mathcal{X}$, we assume $\mathcal{F}$ acts as a group over $\mathbb{H}$, namely $\mathcal{F}$ is closed under function composition and $\mathbb{H}$ is closed under transformations from $\mathcal{F}$. Two hypothesis $h_1, h_2 \in \mathbb{H}$ are said to be equivalent under $\mathcal{F}$ iff there exists $f \in \mathcal{F}$ such that $h_2 = h_1 \circ f$, and hypothesis equivalence under $\mathcal{F}$ is denoted by $\sim_{\mathcal{F}}$. $[h]_{\sim_{\mathcal{F}}} = \{h \circ f : f \in \mathcal{F}\}$ denotes the set of hypotheses which are equivalent up to transformations in $\mathcal{F}$. $\mathbb{H}/\sim_{\mathcal{F}}$ denotes the family of all equivalence classes of $\mathbb{H}$ under $\sim_{\mathcal{F}}$, namely, $\mathbb{H}/\sim_{\mathcal{F}} = \{[h]_{\sim_{\mathcal{F}}} : h \in \mathbb{H}\}$.

The learning scenario assumes that the learner gets samples $\{S_i : i \leq n\}$, where each S_i is a set of samples drawn i.i.d from P_i. The probability distributions are assumed to be pairwise $\mathcal{F}$-related. The learner knows the set of indices of the distributions, $\{1, ..., n\}$ and the family of functions $\mathcal{F}$, but does not know the data-generating distribution nor which specific function f relates any given pair of distributions. In this setting, [59] proposed to exploit the relatedness among tasks by first finding all aspects of the tasks which are invariant under $\mathcal{F}$, and then focus on learning the specific $\mathcal{F}$-sensitive elements of the target task. The potential benefit lies both in the reduction of the search space from the original $\mathbb{H}$ to a smaller $\mathcal{F}$-sensitive subspace $[h]_{\sim_{\mathcal{F}}}$, and from the bigger sample size available to learn the $\mathcal{F}$-invariant aspects of the task. However, the second potential benefit is not guaranteed and depends on the complexity of finding a single $[h]_{\sim_{\mathcal{F}}}$ in $\mathbb{H}/\sim_{\mathcal{F}}$. The complexity of finding $[h]_{\sim_{\mathcal{F}}}$ is formalized using the notion of generalized VC-dimension [39].

3.1.2 Hierarchical Task Relatedness

Next we extend the multi-task learning setting to a hierarchical multi-task learning setting. In the hierarchical setting our objective is the same as in the original one - the learning of a single task by exploiting additional related tasks. Our approach extends the original by assuming that the group of tasks $\{1, ..., n\}$, and the corresponding family of transformation functions $\mathcal{F}$, can be decomposed hierarchically. We denote by l a single level in the hierarchy, $0 \leq l \leq L$ and $\mathcal{T}_l \subseteq \{1, ..., n\}$ the group of related tasks in the l's level of the hierarchy. $\mathcal{F}_l \subset \mathcal{F}$ denotes a family of transformations for which all task in $\mathcal{T}_l$ are pairwise $\mathcal{F}_l$-related.

We assume that the set of transformations for each level $0 \leq l \leq L-1$ can be written as a concatenation of the set of domain transformations corresponding to the preceding level $\mathcal{F}_{l+1}$ and a set of domain transformations $\mathcal{G}_{l+1}$, hence $F_l =$

$\{g \circ f : g \in \mathcal{G}_{l+1}, f \in \mathcal{F}_{l+1}\}$. We call the set of transformations $\mathcal{G}_l$ the set of shared transformations among tasks in $\mathcal{T}_l$.

Definition 1 We say that $\{\mathcal{T}_l, \mathcal{F}_l, \mathcal{G}_l\}_{l=0}^{L}$ is a hierarchical decomposition of a set of $\mathcal{F}$-related tasks $\{1, ..., n\}$ iff:

1. $\mathcal{T}_0 = \{1, .., n\}$
2. $\mathcal{T}_L = \{1\}$, hence $\mathcal{T}_L$ represents the target task.
3. $\mathcal{F}_L = \{f\}$, where f is the identity transformation, hence $f(x) = x$.
4. for all $0 \leq l \leq L - 1$:
   (a) $\mathcal{T}_{l+1} \subset \mathcal{T}_l$
   (b) $\forall i, j \in \mathcal{T}_l$, there exists $f \in \mathcal{F}_l$ such that $P_i(x, b) = P_j(f(x), b)$
   (c) $\mathcal{T}_{l+1}$ shares the set of transformations $\mathcal{G}_{l+1}$
   (d) $\mathcal{F}_l = \{g \circ f : g \in \mathcal{G}_{l+1}, f \in \mathcal{F}_{l+1}\}$.
5. $\mathcal{F}_l$ and $\mathcal{G}_l$ act as a group over $\mathbb{H}$, for all $0 \leq l \leq L$.

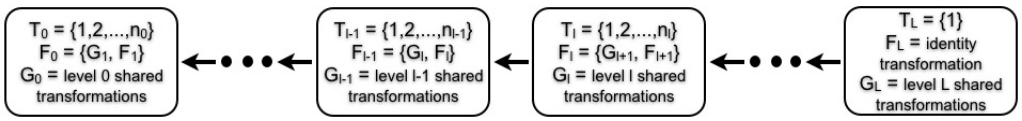


Figure 3.1: An illustration of the hierarchical decomposition $\{\mathcal{T}_l, \mathcal{F}_l, \mathcal{G}_l\}_{l=0}^{L}$. We assume an indexed set of tasks, {1,..,n_0}, where 1 is the objective task. The size of each group of tasks decreases as the level increases, thus: $n_0 > n_{l-1} > n_l > 1$, for $0 < l < L$. An arrow denotes inclusion relations, $\mathcal{T}_L \subset \mathcal{T}_l \subset \mathcal{T}_{l-1} \subset \mathcal{T}_0$ and $\mathcal{F}_L \subset \mathcal{F}_l \subset \mathcal{F}_{l-1} \subset \mathcal{F}_0$.

Fig. 3.1 provides an illustration of the hierarchical decomposition. From the definition of the hierarchical decomposition (4d) we see that the set of transformations for level l can be obtained by concatenating the set of shared transformations of levels $l + 1$ till L. This point will be crucial in understanding the benefit of the cascade hierarchical approach.

Lemma 1 $\mathcal{F}_{l+1} \subset \mathcal{F}_l$, for all $0 \leq l \leq L-1$. Since $\mathcal{G}_{l+1}$ is a group and therefore contains the identity transformation, the lemma follows from the definition in 4.d above.

Lemma 2 Given $h \in \mathbb{H}$, $[h]_{\sim_{\mathcal{F}_{l+1}}} \subset [h]_{\sim_{\mathcal{F}_l}}$, for all $0 \leq l \leq L-1$. This is an immediate consequence of Lemma 1 and the definition of $[h]_{\sim_{\mathcal{F}}}$.

3.1.3 Hierarchical Learning Paradigm, Cascade Approach

In this section we present and motivate our cascade approach for hierarchical multi-task learning. We start by presenting the IMT-ERM (Iterative MT-ERM) which generalizes the MT-ERM (Multi-Task Empirical Risk Minimization) learning paradigm defined in [59] to the hierarchical setting. This serves as a basis for our discussion, motivating our proposed cascade. The cascade method is a more constructive approach for which we can provide precise generalization guarantees; this comes at the cost of restricting the learning scope.

We follow standard notations and denote the empirical error of a hypothesis for sample S as:

$$\hat{Er}^S(h) = \frac{|\{(x,b) \in S : h(x) \neq b\}|}{|S|}.$$

The true error of a hypothesis is:

$$Er^P(h) = P(\{(x,b) \in \mathcal{X} \times \{0,1\} : h(x) \neq b\}).$$

We define the error of any hypothesis space $\mathbb{H}$ as:

$$Er^P(\mathbb{H}) = \inf_{h \in \mathbb{H}} Er^p(h).$$

For notation convenience we shall denote the i'th task in $\mathcal{T}_l$ by l_i.

3.1.3.1 Iterative MT-ERM

Definition 2 Given $\mathbb{H}$, n tasks hierarchically decomposed by $\{\mathcal{T}_l, \mathcal{F}_l, \mathcal{G}_l\}_{l=0}^{L}$ and their sequence of labeled sample sets, $S_1, .., S_n$, the IMT-ERM paradigm works as

follows:

1. $\mathbb{H}^0 = \mathbb{H}$.
2. for $l = 0..L$
   (a) Pick $h = \arg\min_{h \in \mathbb{H}^l} \inf_{h_{l_1},\ldots,h_{l_{|\mathcal{T}_l|}} \in [h]_{\sim_{\mathcal{F}_l}}} \sum_{i=1}^{|\mathcal{T}_l|} \hat{Er}^{S_{l_i}}(h_{l_i})$
   (b) $\mathbb{H}^{l+1} = [h]_{\sim_{\mathcal{F}_l}}$
3. output $h^\diamond$ the single hypothesis in $[h]_{\sim_{\mathcal{F}_L}}$ as the learner's hypothesis.

Note that the fact that $h^\diamond$ is the single hypothesis in $[h]_{\sim_{\mathcal{F}_L}}$ follows directly from the definition of $\mathcal{F}_L$ as containing only the identity transformation.

Following the definition of hierarchical decomposition one can readily see that for $L = 1$ the IMT-ERM is exactly the MT-ERM. Specifically, the first iteration corresponds to the first step of the MT-ERM - learning the aspects which are invariant under $\mathcal{F}_0$. The second iteration corresponds to the second stage of MT-ERM where only the training samples coming from the target task, the single task in $\mathcal{T}_L$, are used to find a single predictor from $[h]_{\sim_{\mathcal{F}_{L-1}}}$, which is the single hypothesis in $[h]_{\sim_{\mathcal{F}_L}}$.

The learning complexity of Step 2a, picking $[h]_{\sim_{\mathcal{F}_l}} \in \mathbb{H}^l / \sim_{\mathcal{F}_l}$, is analyzed in [59]. It uses the notion of generalized VC-dimension from [39], denoted by $d_{\mathbb{H}^l/\sim_{\mathcal{F}_l}}(n)$, where n refers to the number of tasks which is $|\mathcal{T}_l|$ in our setting.

The generalized VC-dimension determines the sample complexity; it has a lower bound of $\sup\{VCdim([h]_{\sim_{\mathcal{F}_l}}) : [h]_{\sim_{\mathcal{F}_l}} \in \mathbb{H}^l / \sim_{\mathcal{F}_l}\}$ and in the general case an upper-bound of $VCdim(\mathbb{H}^l)$. This analysis captures the interrelation between the set of transformations $\mathcal{F}_l$ and the hypothesis space $\mathbb{H}^l$. From Lemma 1 and Lemma 2 we know that both $\sup\{VCdim([h]_{\sim_{\mathcal{F}_l}}) : [h]_{\sim_{\mathcal{F}_l}} \in \mathbb{H}^l / \sim_{\mathcal{F}_l}\}$ and $VCdim(\mathbb{H}^l)$ monotonically decrease with l, which gives rise to the potential of the hierarchical approach.

The MT-ERM paradigm and its hierarchical extension IMT-ERM do not provide a constructive way of preforming Step 2a. In the following we shall consider the conditions under which Step 2a can be replaced by choosing a single hypothesis from the equivalence class derived by the shared transformation at each level: $[h]_{\sim_{\mathcal{G}_l}}$. This yields a constructive approach with an explicit search complexity of $VCdim([h]_{\sim_{\mathcal{G}_l}})$, for each level l in the hierarchy.

3.1.3.2 Cascade Approach

Following Definition 1, we see that each transformation $f \in \mathcal{F}_l$ can be expressed as a concatenation of transformations from all of the shared transformation families $\{\mathcal{G}_i\}_{i=l+1}^{L}$. We propose a learning approach where we exploit the fact that the transformations can be decomposed into shared transformations and learn together the shared transformations at each level of the cascade. In order to preform this shared learning we consider all tasks sharing a set of transformations as a single unified task. For each such unified task representing level l, we search for a single hypothesis in an equivalence class $[h]_{\sim_{\mathcal{G}_l}}$ where $\mathcal{G}_l$ is the shared family of transformations and h is the hypothesis chosen in the previous stage. The unified task is defined next.

For each level in the hierarchy $0 \leq l \leq L$, we define the task representing the level as the union of all tasks in $\mathcal{T}_l$. As before we call the i'th task in $\mathcal{T}_l$ $l_i \in \{1..n\}$. Let P_1,...P_n be the probability distributions over $\mathcal{X} \times \{0,1\}$ of tasks $\{1..n\}$ respectively. We shall now define the probability distribution over $\mathcal{X} \times \{0,1\}$, which describes the single task $\mathbf{a}_l$ representing the union of all tasks in $\mathcal{T}_l$ as the average of distribution of tasks in $\mathcal{T}_l$. This is equivalent to defining a joint probability space where eash task is given the same uniform prior. Hence:

$$P_{\mathbf{a}_l}(x,b) = \frac{1}{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} P_{l_i}(x,b) \tag{3.1}$$

Lemma 3

$$Er^{P_{\mathbf{a}_l}}(h) = \frac{1}{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h) \tag{3.2}$$

This immediately follows from the definition of $P_{\mathbf{a}_l}$ and Er^P.

We denote by S_{a_l} the union of all samples from all tasks belonging to $\mathcal{T}_l$, $0 \leq l \leq L$. For a sufficiently large i.i.d sample, $\hat{Er}^{S_{a_l}}(h)$ converges to $Er^{P_{\mathbf{a}_l}}(h)$.

Next we define the CMT-ERM paradigm (Cascade Multi-Task ERM). Here for each level $0 \leq l \leq L$ of the hierarchy we search for an optimal hypothesis for the

unified task $\mathbf{a}_l$. The algorithm iterates through two steps. The first step in iteration l defines the search space as the equivalence class $[h]_{\sim_{\mathcal{G}_l}}$ of the best hypothesis h from the previous iteration. In the second step we search this equivalence class for a single best hypothesis given task $\mathbf{a}_l$. See Fig. 3.2 for an illustration of the algorithm.

Definition 3 Given $\mathbb{H}$, n tasks hierarchically decomposed by $\{\mathcal{T}_l, \mathcal{F}_l, \mathcal{G}_l\}_{l=0}^{L}$ and the sequence of labeled sample sets corresponding to each unified task $S_{a_0}, .., S_{a_L}$, the CMT-ERM paradigm works as follows:

1. Pick $h \in \mathbb{H}$ that minimizes $\hat{Er}^{S_{a_0}}(h)$
2. for $l = 0..L$
   (a) $\mathbb{H}^l = [h]_{\sim_{\mathcal{G}_l}}$
   (b) Pick $h \in \mathbb{H}^l$ that minimizes $\hat{Er}^{S_{a_l}}(h)$
3. output $h^\diamond = h$.

Note that unlike the IMT-ERM or MT-ERM paradigms, the learning stage corresponding to the shared transformations from all tasks in Step 2b is a single well defined ERM problem. The parameter governing the sample complexity of the shared learning (learning task $\mathbf{a}_l$) can also be precisely defined - $VCdim([h]_{\sim_{\mathcal{G}_l}})$. The potential gain in such an approach lies in the increased number of samples available for the unified tasks and the decrease in search complexity as compared to the original $VCdim(\mathbb{H})$ (recall the rectangle example in the introduction, and see discussion in Section A.1.3 of the Appendix).

In Step 1 we search over all $\mathbb{H}$. This step corresponds to task $\mathbf{a}_0$, which is the union over all tasks, for which we assume a sufficient amount of samples is available. Under certain conditions the search for an optimal hypothesis in Step 1 can be avoided; for instance, we may choose a hypothesis which has minimal false-negatives irrespective of the number of false positives. See example in Fig. 3.2 and further discussion in Section 3.2.3.

The remaining analysis is divided into two parts: First, in Section 3.2 we consider the conditions under which such an approach is applicable. We ask under what conditions this approach guarantees optimality with respect to the original objective

task. Specifically, we ask when the transformation in $\mathcal{G}_l$, which is optimal for the unified task $\mathbf{a}_l$, is also optimal for a single task in level l. In other words, under what conditions $\arg\min_{g\in\mathcal{G}_l} Er^{P_{l_j}}([h\circ g]_{\sim_{\mathcal{F}_l}}) = \arg\min_{g\in\mathcal{G}_l} Er^{P_{\mathbf{a}_l}}(h\circ g) \;\; \forall l_j \in \mathcal{T}_l$.

Second, in Section 3.3 we extend our analysis to the PAC learning scenario, providing generalization error guarantees for a single iteration of the CMT-ERM. In Theorem 3 we finally provide generalization error guarantees based on the proposed CMT-ERM approach.

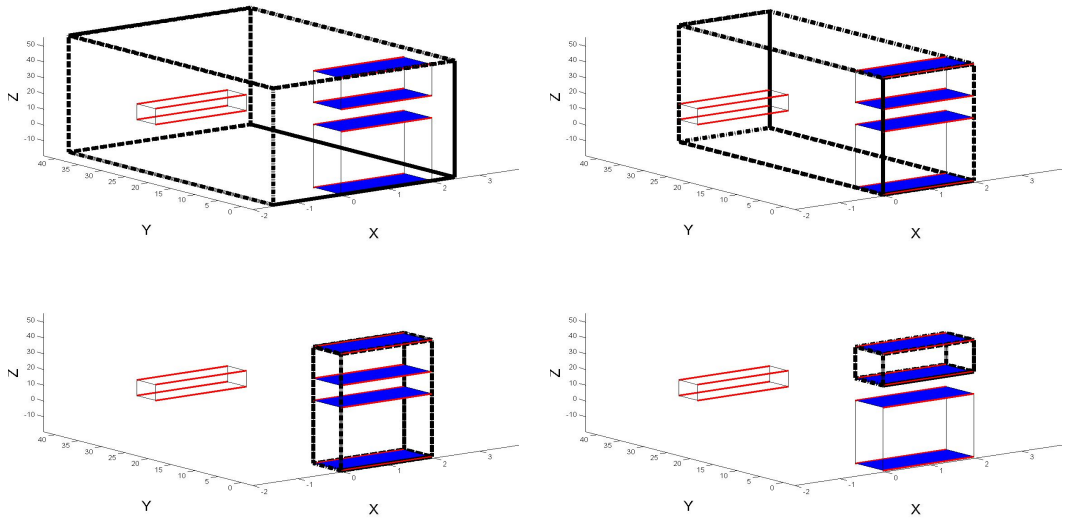


Figure 3.2: An illustration of the CMT-ERM paradigm applied to learning rectangles in R^3. Each task requires to learn a single rectangle. This is an example of a hierarchy with three levels for which G_0 is the family of translations and scaling of the X axis, G_1 is the family of translations and scaling of the Y axis and G_2 is the family of translations and scaling of the Z axis. The shared axis of all 3 tasks is marked by red, the shared axis of the two tasks at level $l = 1$ are marked by blue. For coherence we draw the rectangles and not samples from the rectangles. Each rectangle learning task is specified by a distribution P^r; we denote each rectangle by r and assume a realizable scenario - $P^r(\{(x, 0) : x \in r\}) = 0$ and $P^r(\{(x, 1) : x \notin r\}) = 0$ (only the area of the support r is drawn). We assume $P^r(\{(x, 0) : x \in \mathcal{X}\}) < P^r(\{(x, 1) : x \in \mathcal{X}\})$. Each graph corresponds to a specific step of the algorithm, starting from the top-left graph which corresponds to step 1 of the CMT-ERM paradigm. Note that this is an example where we don't need to find the optimal ERM solution but any solution covering the support of the rectangles will suffice. The top-right graph corresponds to the first iteration of step 2b, where the X axis is learnt. The bottom-left graph corresponds to the second iteration of step 2b, where the Y axis is learnt. The bottom-right graph corresponds to the third iteration of step 2b, where the Z axis is learnt from the single remaining task, the top rectangle.

3.2 Cascade Optimality

In the cascade approach we compute the final hypothesis in stages defined by the concatenation of transformations, solving at each step a simpler problem with a larger sample. In this section we analyze the conditions under which an optimal hypothesis of a single task can be found by searching through a cascade of shared transformations, where each transformation fits a single level in the hierarchical decomposition described above. We start by analyzing specific properties of the optimal transformations given our hierarchical decompositions in Section 3.2.1. In Sections 3.2.2 and 3.2.3 we present two properties of a cascade of shared transformations, which are the basis of the assumptions under which the cascade approach can reach optimality. Finally, in Section 3.2.4 we state the assumptions and prove that the cascade approach can reach optimality.

3.2.1 Hierarchical Task Relatedness Properties

We recall that from Lemma 2 in [59] we can deduce that for any task $j \in \mathcal{T}_l$

$$Er^{P_{l_j}}([h]_{\sim_{\mathcal{F}_l}}) = \inf_{h_1,\dots h_{|\mathcal{T}_l|} \in [h]_{\sim_{\mathcal{F}_l}}} \frac{1}{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h_i) \tag{3.3}$$

In the definition of the hierarchical decomposition above (Definition 1), we assumed that tasks in $\mathcal{T}_l$ share the set of transformations $\mathcal{G}_l$. In the following we show that any $g \in \mathcal{G}_l$ which is optimal for a single task $j \in \mathcal{T}_l$ in the sense that it minimizes $Er^{P_{l_j}}([h \circ g]_{\sim_{\mathcal{F}_l}})$, is optimal for all other tasks in $\mathcal{T}_l$.

Lemma 4 For each $j \in \mathcal{T}_l$, $g^* = \arg\min_{g \in \mathcal{G}_l} Er^{P_{l_j}}([h \circ g]_{\sim_{\mathcal{F}_l}}) \Leftrightarrow \forall i \in \mathcal{T}_l$ and $\forall g \in \mathcal{G}_l$ $Er^{P_{l_i}}([h \circ g^*]_{\sim_{\mathcal{F}_l}}) \leq Er^{P_{l_i}}([h \circ g]_{\sim_{\mathcal{F}_l}})$.

Proof $\Leftarrow$: Immediate; if g^* attains the minimum for all $i \in \mathcal{T}_l$, it does so also for j. $\square$

$\Rightarrow$: Assume that $g^* = \arg\min_{g \in \mathcal{G}_l} Er^{P_{l_j}}([h \circ g]_{\sim_{\mathcal{F}_l}})$. Let $f^g \in \mathcal{F}_l$ be the transformation which minimizes $Er^{P_{l_j}}([h \circ g]_{\sim_{\mathcal{F}_l}})$, so that by definition $Er^{P_{l_j}}(h \circ g \circ f^g) = Er^{P_{l_j}}([h \circ$

$g]_{\sim_{\mathcal{F}_l}}$). $\mathcal{F}_l$ is the family of transformations between tasks in $\mathcal{T}_l$, thus $\forall i \in \mathcal{T}_l \; \exists f_{ij} \in \mathcal{F}_l$, such that $Er^{P_{l_j}}(h \circ g \circ f^g) = Er^{P_{l_i}}(h \circ g \circ f^g \circ f_{ij})$, $\forall g \in \mathcal{G}_l$.

By the definition of $\mathcal{F}_l$ as a group we know that $f^g \circ f_{ij} \in \mathcal{F}_l$. Now assume that the lemma is false and therefore $Er^{P_{l_i}}(h \circ g \circ f^g \circ f_{ij}) \neq Er^{P_{l_i}}([h \circ g]_{\sim_{\mathcal{F}_l}})$. It follows that there exists $z \in \mathcal{F}_l$ such that $Er^{P_{l_i}}(h \circ g \circ z) = Er^{P_{l_i}}([h \circ g]_{\sim_{\mathcal{F}_l}})$ and $Er^{P_{l_i}}(h \circ g \circ z) < Er^{P_{l_i}}(h \circ g \circ f^g \circ f_{ij})$. Let $f_{ji} \in \mathcal{F}_l$ be the transformation from task i to task j. Thus $Er^{P_{l_j}}(h \circ g \circ z \circ f_{ji}) < Er^{P_{l_j}}(h \circ g \circ f^g)$, which contradicts the definition of f^g since $z \circ f_{ji} \in \mathcal{F}_l$.

We can therefore conclude that $Er^{P_{l_i}}(h \circ g \circ f^g \circ f_{ij}) = Er^{P_{l_i}}([h \circ g]_{\sim_{\mathcal{F}_l}})$. Since $Er^{P_{l_j}}(h \circ g^* \circ f^{g^*}) \leq Er^{P_{l_j}}(h \circ g \circ f^g) \; \forall g \in \mathcal{G}_l$, it follows that $Er^{P_{l_i}}(h \circ g^* \circ f^{g^*} \circ f_{ij}) \leq Er^{P_{l_i}}(h \circ g \circ f^g \circ f_{ij}) \; \forall i \in \mathcal{T}_l$. Thus, $\forall i \in \mathcal{T}_l$ and $\forall g \in \mathcal{G}_l \; Er^{P_{l_i}}([h \circ g^*]_{\sim_{\mathcal{F}_l}}) \leq Er^{P_{l_i}}([h \circ g]_{\sim_{\mathcal{F}_l}})$. □

Lemma 5 If $g' = \arg\min_{g \in \mathcal{G}} Er^P([h \circ g]_{\sim_{\mathcal{F}}})$ and $f' = \arg\min_{f \in \mathcal{F}} Er^P(h \circ g' \circ f)$, then $g' = \arg\min_{g \in \mathcal{G}} Er^P(h \circ g \circ f')$

Proof For each $g \in \mathcal{G}$, denote by f^g the optimal $f \in \mathcal{F}$ with respect to g, thus $f^g = \arg\min_{f \in \mathcal{F}} Er^P(h \circ g \circ f)$. From the definition of g' we know:

$$Er^P(h \circ g' \circ f') \leq Er^P(h \circ g \circ f^g), \forall g \in \mathcal{G}. \tag{3.4}$$

From the definition of f^g we know:

$$Er^P(h \circ g \circ f^g) \leq Er^P(h \circ g \circ f'), \forall g \in \mathcal{G}. \tag{3.5}$$

From (3.4) and (3.5) we get that:

$$Er^P(h \circ g' \circ f') \leq Er^P(h \circ g \circ f'), \forall g \in \mathcal{G}.$$

It follows that $g' = \arg\min_{g \in \mathcal{G}} Er^P(h \circ g \circ f')$. □

3.2.2 The property of Transformation-Multiplicativity

Definition 4 Two transformations taken from two families of transformations $g \in \mathcal{G}$ and $f \in \mathcal{F}$ are multiplicative with respect to hypothesis $h \in \mathbb{H}$ iff $h \circ g \circ f(x) =$

$h \circ g(x) \cdot h \circ f(x)$.

It is easy to see that if $g \in \mathcal{G}$ and $f \in \mathcal{F}$ are multiplicative then $\{x|h \circ g \circ f(x) = 1\} = \{x|h \circ g(x) = 1\} \cap \{x|h \circ f(x) = 1\}$. In other words, the support of the concatenated transformation is contained in the support of each of the transformations under hypothesis h.

To illustrate the multiplicative property of transformations, let the family of hypotheses $\mathbb{H}$ include all rectangles in R^2. Let $\mathcal{G}$ denote the group of scale and translation of the first dimension, and $\mathcal{F}$ denote the group of scale and translation of the second dimension. We parametrize each rectangle by $[c1, c2, s1, s2]$, where $c1$ and $c2$ denote the center of the rectangle, and $[s1, s2]$ denote the rectangle's sides (width and height). Choose $h \in \mathbb{H}$ to be the rectangle $[3, 3, 2, 2]$. Choose $g \in \mathcal{G}$ to be translation of 1 and scale of 2 of the first dimension, and $f \in \mathcal{F}$ similarly for the second dimension.

With this choice, $h \circ g$ corresponds to rectangle $[2, 3, 1, 2]$ and $h \circ f$ to $[3, 2, 2, 1]$. The intersection of their support corresponds to rectangle $[2, 2, 1, 1]$ which is the same as $h \circ g \circ f(x)$. On the other hand if we were to consider just translations but no scaling, we would get that $h \circ g$ is rectangle $[2, 3, 2, 2]$ and $h \circ f$ $[3, 2, 2, 2]$. The intersection of their support would be the rectangle parametrized by $[2.5, 2.5, 1.5, 1.5]$; it is not equal to $h \circ g \circ f(x)$, which is rectangle $[2, 2, 2, 2]$.

The *transformation-multiplicativity* property lets us write $Er^P(h \circ g \circ f)$ as $Er^P(h \circ g)$ plus a residual term, describing the gain obtained by adding the transformation from $\mathcal{F}$. We shall denote the residual term by R^{gf}, where

$$\begin{aligned} R^{gf} &= P(\{(x, b) \in \mathcal{X} \times \{0, 1\} : b = 1, h \circ g(x) = 1, h \circ f(x) = 0\}) \\ &- P(\{(x, b) \in \mathcal{X} \times \{0, 1\} : b = 0, h \circ g(x) = 1, h \circ f(x) = 0\}) \end{aligned} \quad (3.6)$$

This term measures the volume of new errors introduced when adding transformation f, while eliminating the volume of those errors of g which are corrected for by f. Under the *transformation-multiplicativity* assumption, adding a transformation f can change the overall classification value only for points in the support of $h \circ g$. In the following, for the general case (not necessarily assuming *transformation-multiplicativity*) we shall refer to this as the amount by which f 'corrects' the support of g.

Lemma 6 If transformations $g \in \mathcal{G}$ and $f \in \mathcal{F}$ are multiplicative with respect to hypothesis $h \in \mathbb{H}$ then:

$$Er^P(h \circ g \circ f) = Er^P(h \circ g) + R^{gf} \tag{3.7}$$

Proof See Appendix A.1.1.

Choosing a transformation $f \in \mathcal{F}$ which minimizes $Er^P(h \circ g \circ f)$ implies that $R^{gf} \leq 0$ as it can only decrease the overall error. This is stated formally in the following lemma.

Lemma 7 $f = \arg\min_{f' \in \mathcal{F}} Er^P(h \circ g \circ f') \Rightarrow R^{gf} \leq 0$

Proof The lemma follows from Lemma 6 and the fact that the group $\mathcal{F}$ contains the identity transformation.

Lemma 6 thus shows that under the *transformation-multiplicativity* assumption, the error $Er^P(h \circ g \circ f)$ can be decomposed into 2 terms: the error obtained by choosing $g \in \mathcal{G}$, $Er^P(h \circ g)$, and a residual term R^{gf} referring to the change in the overall classification value of points in the support of $h \circ g$ when adding a transformation $f \in \mathcal{F}$.

3.2.3 The property of Indifference

We say transformation $f \in \mathcal{F}$ is *indifferent* if for two transformations from a different family of transformations $g, g^* \in \mathcal{G}$, where g^* is optimal, the difference between the amount f 'corrects' the support of g and the amount f 'corrects' the support of g^* is bounded by the difference of the errors between g and g^*. Formally,

Definition 5 A transformation $f \in \mathcal{F}$ is said to be *indifferent* with respect to distribution P, hypothesis h and transformation $g \in \mathcal{G}$, if for $g^* = \arg\min_{g \in \mathcal{G}} Er^P(h \circ g)$ the following holds:

$$R^{g^* f} - R^{gf} \quad \leq \quad Er^P(h \circ g) - Er^P(h \circ g^*) \tag{3.8}$$

To illustrate scenarios where this property occurs, consider some family of hypothesis $\mathbb{H}$ and two groups of transformations $\mathcal{G}$ and $\mathcal{F}$. Assume that any two transformations $g \in \mathcal{G}$ and $f \in \mathcal{F}$ are statistically independent for any $h \in \mathbb{H}$, namely,

$$\begin{aligned} &P(\{(x,b) : h \circ g(x) = a, h \circ f(x) = c\}) = \\ &P(\{(x,b) : h \circ g(x) = a\}) \cdot P(\{(x,b) : h \circ f(x) = c\}) \end{aligned} \tag{3.9}$$

Based on this assumption and the definition of the residual R^{gf}, it is easy to see that we can write the residual difference as:

$$\begin{aligned} &R^{g^*f} - R^{gf} = \\ &[P(\{(x,b) : b = 1, hg(x) = 0\}) - P(\{(x,b) : b = 1, hg^*(x) = 0\})]P(\{(x,b) : b = 1, hf(x) = 0\}) + \\ &[P(\{(x,b) : b = 0, hg(x) = 1\}) - P(\{(x,b) : b = 0, hg^*(x) = 1\})]P(\{(x,b) : b = 0, hf(x) = 0\}) \end{aligned} \tag{3.10}$$

Consider the case where $P(\{(x,b) : b = 1, hg^*(x) = 0\}) = 0$ and $P(\{(x,b) : b = 1, hf(x) = 0\}) = 0$. Define $\mathbb{H}^* = \{h : P(\{(x,b) : b = 1, hg^*(x) = 0\}) = 0\}$ and $\mathcal{F}^* = \{f : P(\{(x,b) : b = 1, hf(x) = 0\}) = 0\}$. Under these definitions it is easy to see that $\forall h \in \mathbb{H}^*$, $\forall g \in \mathcal{G}$ and $\forall f \in \mathcal{F}^*$ the following holds:

$$\begin{aligned} &R^{g^*f} - R^{gf} \leq \\ &[P(\{(x,b) : b = 1, hg(x) = 0\}) - P(\{(x,b) : b = 1, hg^*(x) = 0\})] + \\ &[P(\{(x,b) : b = 0, hg(x) = 1\}) - P(\{(x,b) : b = 0, hg^*(x) = 1\})] = \\ &Er^P(h \circ g) - Er^P(h \circ g^*). \end{aligned} \tag{3.11}$$

Thus all $f \in \mathcal{F}^*$ are *indifferent* with respect to P, any $g \in \mathcal{G}$ and any $h \in \mathbb{H}^*$.

This is a simple example of *indifference*. In Appendix A.1.2 we consider several other sets of sufficient conditions for which *indifference* occurs, assuming the statistical independence of the transformations. Table A.1 in the Appendix summarizes these conditions.

3.2.4 Optimality Proof

Now we are ready to state the following two assumptions under which the optimal transformation $g^l \in \mathcal{G}_l$ for each of the tasks in $\mathcal{T}_l$ is the same as the optimal transformation $g^{\mathbf{a}_l} \in \mathcal{G}_l$ for task $\mathbf{a}_l$.

We use the following notations:

- $g^{\mathbf{a}_l} = \arg\min_{g \in \mathcal{G}_l} Er^{P_{\mathbf{a}_l}}(h \circ g)$, the optimal transformation with respect to the task $\mathbf{a}_l$, representing the union of tasks in T_l.

- $g^l = \arg\min_{g \in \mathcal{G}_l} Er^{P_{l_i}}([h \circ g]_{\sim_{\mathcal{F}_l}})$, $\forall i \in |\mathcal{T}_l|$, the optimal transformation which is shared among all of the tasks in $\mathcal{T}_l$ (it follows from Lemma 4 that g^l exists).

- $f^{l_i} = \arg\min_{f \in \mathcal{F}_l} Er^{P_{l_i}}(h \circ g^l \circ f)$, $\forall i \in |\mathcal{T}_l|$, the optimal transformation which is specific to each of the tasks in $\mathcal{T}_l$.

Assumption 1 $\forall i \in |\mathcal{T}_l|$, both $g^{\mathbf{a}_l}$, f^{l_i} and g^l , f^{l_i} are *transformation-multiplicative*.

Assumption 2 $\forall i \in |\mathcal{T}_l|$, f^{l_i} is *indifferent* with respect to distribution $P_{\mathbf{a}_l}$, the given hypothesis h and $g^l \in \mathcal{G}_l$.

Theorem 1 Under assumptions 1 and 2:

$$\arg\min_{g \in \mathcal{G}_l} Er^{P_{\mathbf{a}_l}}(h \circ g) = \arg\min_{g \in \mathcal{G}_l} Er^{P_{l_i}}([h \circ g]_{\sim_{\mathcal{F}_l}}), \forall i \in \mathcal{T}_l$$

Proof

$$\sum_{i=1}^{|\mathcal{T}_l|} R^{g^{\mathbf{a}_l} f^{l_i}} - \sum_{i=1}^{|\mathcal{T}_l|} R^{g^l f^{l_i}} \leq \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{\mathbf{a}_l}}(h \circ g^l) - \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{\mathbf{a}_l}}(h \circ g^{\mathbf{a}_l}) \Leftrightarrow \tag{3.12}$$

$$\sum_{i=1}^{|\mathcal{T}_l|} R^{g^{\mathbf{a}_l} f^{l_i}} - \sum_{i=1}^{|\mathcal{T}_l|} R^{g^l f^{l_i}} \leq \frac{1}{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^l) - \frac{1}{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l}) \Leftrightarrow \tag{3.13}$$

$$\sum_{i=1}^{|\mathcal{T}_l|} R^{g^{\mathbf{a}_l} f^{l_i}} - \sum_{i=1}^{|\mathcal{T}_l|} R^{g^l f^{l_i}} \leq \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^l) - \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l}) \Leftrightarrow \tag{3.14}$$

$$\sum_{i=1}^{|\mathcal{T}_l|} [Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l}) + R^{g^{\mathbf{a}_l} f^{l_i}}] \leq \sum_{i=1}^{|\mathcal{T}_l|} [Er^{P_{l_i}}(h \circ g^l) + R^{g^l f^{l_i}}] \Leftrightarrow \tag{3.15}$$

$$\sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l} \circ f^{l_i}) \leq \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^l \circ f^{l_i}) \Leftrightarrow \tag{3.16}$$

$$\sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l} \circ f^{l_i}) = \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^l \circ f^{l_i}) \Leftrightarrow \tag{3.17}$$

$$\sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l} \circ f^{l_i}) = \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g^l]_{\sim_{\mathcal{F}_l}}) \Leftrightarrow \tag{3.18}$$

$$\sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g^{\mathbf{a}_l}]_{\sim_{\mathcal{F}_l}}) = \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g^l]_{\sim_{\mathcal{F}_l}}) \tag{3.19}$$

In the derivation above: (3.12) follows from Assumption 2 and (3.12) $\Leftrightarrow$ (3.13) follows from the definition of the unified task (3.1). Note that in (3.12) the residual is computed based on $P_{\mathbf{a}_l}$, and in (3.13) it is based on each of the P_{l_i}. (3.15) $\Leftrightarrow$ (3.16) follows from Lemma 6. (3.16) $\Rightarrow$ (3.17) uses the optimality of g^l. (3.17) $\Leftrightarrow$ (3.18) follows from the definition of f^{l_i} and the definition of $Er^P([h]_{\sim_{\mathcal{F}}})$. To see that (3.18) $\Leftrightarrow$ (3.19), assume otherwise; from the definition of $Er^P([h]_{\sim_{\mathcal{F}}})$ we get that $\sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g^{\mathbf{a}_l}]_{\sim_{\mathcal{F}_l}}) < \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l} \circ f^{l_i})$, thus $\sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g^{\mathbf{a}_l}]_{\sim_{\mathcal{F}_l}}) < \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g^l]_{\sim_{\mathcal{F}_l}})$, which contradicts the optimality of g^l .

$g^{\mathbf{a}_l} = \arg\min_{g \in \mathcal{G}_l} Er^{P_{\mathbf{a}_l}}(h \circ g)$ also attains the minimum of the sum over the error

of all tasks in $\mathcal{T}_l$, thus:

$$g^{\mathbf{a}_l} = \arg\min_{g \in \mathcal{G}_l} \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g]_{\sim_{\mathcal{F}_l}})$$

From (3.3) above it therefore follows that:

$$g^{\mathbf{a}_l} = \arg\min_{g \in \mathcal{G}_l} Er^{P_{l_i}}([h \circ g]_{\sim_{\mathcal{F}_l}}), \forall i \in \mathcal{T}_l. \square$$

(We note in passing that other transformations may also achieve this minimum.)

3.3 Multi-Task Cascade ERM

In the previous section we showed that the optimal transformation of a task can be found by considering a group of tasks together. This implies a learning scenario where searching for the optimal transformation of a single task can be done by considering the group of tasks sharing this transformation, thus benefiting from the bigger sample size contributed from the whole group.

Our discussion until now focused on the optimal solution. Now we extend this analysis to the case where we cannot guarantee optimality of the solution. For this we need to extend our assumptions to imply that a near optimal solution for the group of tasks considered together is also near optimal for each of the tasks considered separately, thus permitting the derivation of an ERM approach.

To begin with, instead of considering the *transformation-multiplicativity* only for the optimal choice of transformations, we assume *transformation-multiplicativity* between $f^{l_i} \in \mathcal{F}_l$, the optimal transformation for task $l_i \in \mathcal{T}_l$, and any transformation $g \in \mathcal{G}_l$ which is close to the optimal transformation $g^{\mathbf{a}_l}$. Let $\mathcal{G}_l^\epsilon$ denote the set of all transformations in $\mathcal{G}_l$ which have an error bigger by at most ϵ when measured according to $P_{\mathbf{a}_l}$. Formally,

Definition 6 Given hypothesis $h \in \mathbb{H}^{l-1}$, $g \in \mathcal{G}_l^\epsilon$ iff:

$$Er^{P_{\mathbf{a}_l}}(h \circ g) \leq Er^{P_{\mathbf{a}_l}}(h \circ g^{\mathbf{a}_l}) + \epsilon$$

3. HIERARCHICAL MULTI-TASK LEARNING: A CASCADE APPROACH BASED ON THE NOTION OF TASK RELATEDNESS

Next we extend the *indifference* assumption, adding a lower bound to the difference of residuals. Thus, the improvement achieved by adding the transformation to a near optimal transformation is close to the improvement when adding a transformation to an optimal transformation.

In order to extend our cascade approach from two levels to the $L+1$ levels in the hierarchy, we extend the assumptions to the whole hierarchy:

Cascade assumptions: for $l = 0..L$ and $h \in \mathbb{H}^{l-1}$ the chosen hypothesis for Pa_{l-1}:

1. $\forall g \in \mathcal{G}_l^\epsilon$, $\forall i \in |\mathcal{T}_l|$ and $f^{l_i} = \arg\min_{f \in \mathcal{F}_l} Er^{P_{l_i}}(h \circ g \circ f)$, g and f^{l_i} are multiplicative with respect to h; therefore

$$h \circ g \circ f^{l_i}(x) = h \circ g(x) \cdot h \circ f^{l_i}(x)$$

2. For:

   - $g^{\mathbf{a}_l} = \arg\min_{g \in \mathcal{G}_l} Er^{P_{\mathbf{a}_l}}(h \circ g)$
   - $\hat{f^{l_i}} = \arg\min_{f \in \mathcal{F}_l} Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l} \circ f)$, $\forall i \in [1..|\mathcal{T}_l|]$

   $\hat{f^{l_i}}$ is *indifferent* with respect to the distribution $P_{\mathbf{a}_l}$, hypothesis h and $\forall g \in \mathcal{G}_l^\epsilon$: $|\sum_{i=1}^{|\mathcal{T}_l|} R^{g^{\mathbf{a}_l}\hat{f^{l_i}}} - \sum_{i=1}^{|\mathcal{T}_l|} R^{g\hat{f^{l_i}}}| \leq \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{\mathbf{a}_l}}(h \circ g) - \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{\mathbf{a}_l}}(h \circ g^{\mathbf{a}_l})$

(Note that we consider $\mathbb{H}^{-1}$ to be the original hypothesis space $\mathbb{H}$.)

For simplicity, in the following we shall refer to the strong form of Assumptions 1,2 stated above as *transformation-multiplicativity* and *indifference* respectively.

We can now analyze the error bound of a hypothesis composed of some original hypothesis h, a shared transformation learnt via an ERM process $g^\diamond$, and the optimal transformation for each task f^{l_i} given the hypothesis $h \circ g^\diamond$.

Theorem 2 Let $d = VCdim([h]_{\sim_{\mathcal{G}_l}})$, $h^* = \arg\min_{h \in [h]_{\sim_{\mathcal{G}_l}}} Er^{P_{\mathbf{a}_l}}(h)$ and $h^\diamond \in [h]_{\sim_{\mathcal{G}_l}}$ denote the output of a standard ERM algorithm trained on task $\mathbf{a}_l$ with sample size $|S_{\mathbf{a}_l}|$. Then for every ϵ and $\delta > 0$ and if $|S_{\mathbf{a}_l}| \geq c_0(\frac{1}{\epsilon} log\frac{1}{\delta} + \frac{d}{\epsilon} log\frac{1}{\epsilon})$, with probability greater than $(1-\delta)$

$$\forall l_i \in \mathcal{T}_l, \quad Er^{P_{l_i}}([h^\diamond]_{\sim_{\mathcal{F}_l}}) \leq Er^{P_{l_i}}([h^*]_{\sim_{\mathcal{F}_l}}) + 2\epsilon$$

Proof We note that from the above definition of $g^{\mathbf{a}_l}$ and $h^* = h \circ g^{\mathbf{a}_l}$, we may also write $h^\diamond$ as the original hypothesis h and the chosen transformation $g^\diamond \in \mathcal{G}_l$ such that $h^\diamond = h \circ g^\diamond$. We know that for a standard ERM algorithm, with probability greater than $(1-\delta)$

$$Er^{P_{\mathbf{a}_l}}(h \circ g^\diamond) \leq Er^{P_{\mathbf{a}_l}}(h \circ g^{\mathbf{a}_l}) + \epsilon \tag{3.20}$$

From Lemma 3 we can write

$$\sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^\diamond) \leq \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l}) + \epsilon$$

From the *transformation-multiplicativity* assumption and Lemma 6 we can write

$$\sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g^\diamond]_{\sim_{\mathcal{F}_l}}) - \sum_{i=1}^{|\mathcal{T}_l|} R^{g^\diamond f^{l_i}} \leq \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g^{\mathbf{a}_l}]_{\sim_{\mathcal{F}_l}}) - \sum_{i=1}^{|\mathcal{T}_l|} R^{g^{\mathbf{a}_l} \hat{f^{l_i}}} + \epsilon \tag{3.21}$$

where $\hat{f^{l_i}} \in \mathcal{F}_l$ are the optimal transformations for all $l_i \in \mathcal{T}_l$ given hypothesis $h \circ g^{\mathbf{a}_l}$, and $f^{l_i} \in \mathcal{F}_l$ are the optimal transformations for all $l_i \in \mathcal{T}_l$ given hypothesis $h \circ g^\diamond$.

From the strong form of the *indifference* assumption we know that

$$|\sum_{i=1}^{|\mathcal{T}_l|} R^{g^{\mathbf{a}_l} \hat{f^{l_i}}} - \sum_{i=1}^{|\mathcal{T}_l|} R^{g^\diamond \hat{f^{l_i}}}| \leq \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{\mathbf{a}_l}}(h \circ g^\diamond) - \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{\mathbf{a}_l}}(h \circ g^{\mathbf{a}_l}) \Leftrightarrow \tag{3.22}$$

$$|\sum_{i=1}^{|\mathcal{T}_l|} R^{g^{\mathbf{a}_l} \hat{f^{l_i}}} - \sum_{i=1}^{|\mathcal{T}_l|} R^{g^\diamond \hat{f^{l_i}}}| \leq \frac{1}{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^\diamond) - \frac{1}{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l}) \Leftrightarrow \tag{3.23}$$

$$|\sum_{i=1}^{|\mathcal{T}_l|} R^{g^{\mathbf{a}_l} \hat{f^{l_i}}} - \sum_{i=1}^{|\mathcal{T}_l|} R^{g^\diamond \hat{f^{l_i}}}| \leq \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^\diamond) - \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}(h \circ g^{\mathbf{a}_l}) \tag{3.24}$$

(3.22) $\Leftrightarrow$ (3.23) follows from the definition of the unified task (3.1). Note that in (3.22) the residual is computed based on $P_{\mathbf{a}_l}$, while in (3.23) and (3.24) it is based on each of the P_{l_i}.

Since $f^{l_i} \in \mathcal{F}_l$ are the optimal transformations for all $l_i \in \mathcal{T}_l$ given hypothesis

$h \circ g^{\diamond}$, we get that $\forall i$:

$$R^{g^{\diamond} f^{l_i}} \leq R^{g^{\diamond} \hat{f^{l_i}}} \tag{3.25}$$

From (3.20), (3.24) and (3.25) we can write

$$-\epsilon \leq \sum_{i=1}^{|\mathcal{T}_l|} R^{g^{\mathbf{a}_l} \hat{f^{l_i}}} - \sum_{i=1}^{|\mathcal{T}_l|} R^{g^{\diamond} f^{l_i}} \tag{3.26}$$

Putting together (3.21) and (3.26) gives

$$\sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g^{\diamond}]_{\sim_{\mathcal{F}_l}}) \leq \sum_{i=1}^{|\mathcal{T}_l|} Er^{P_{l_i}}([h \circ g^{\mathbf{a}_l}]_{\sim_{\mathcal{F}_l}}) + 2\epsilon \tag{3.27}$$

From the definition of $\mathcal{F}_l$ and from Lemma 2 in [59] we can conclude that

$$\forall l_i \in \mathcal{T}_l, \qquad Er^{P_{l_i}}([h \circ g^{\diamond}]_{\sim_{\mathcal{F}_l}}) \leq Er^{P_{l_i}}([h \circ g^{\mathbf{a}_l}]_{\sim_{\mathcal{F}_l}}) + 2\epsilon \tag{3.28}$$

□

Recall the CMT-ERM paradigm presented in Section 3.1.3.2. Theorem 2 deals with the error accumulated in a single stage of the cascade, in which a near optimal shared hypothesis has been found. In the following theorem we extend this analysis to all levels in the hierarchy, and conclude the overall generalization analysis of the CMT-ERM paradigm.

In the first iteration we choose a hypothesis rather than a transformation like in the remaining iterations. In order to apply the same analysis we choose hypothesis $h \in \mathbb{H}$ for P_{a_0}, and define $\mathcal{G}_{-1}$ to be the set of transformations for which $[h]_{\sim_{\mathcal{G}_{-1}}} = \mathbb{H}$. We note that as we don't actually search over $[h]_{\sim_{\mathcal{G}_{-1}}}$ but rather $\mathbb{H}$, $\mathcal{G}_{-1}$ needs to exist though we do not need to know it explicitly and we consider it just as a notation for writing each hypothesis $h' \in \mathbb{H}$ as $h' = h \circ g'$. For consistent notion we define $S_{\mathbf{a}_{-1}} = S_{\mathbf{a}_0}$, where $S_{\mathbf{a}_0}$ is the sample set for choosing $h \in \mathbb{H}$ (recall the definition of CMT-ERM). Thus in the following analysis the sample set size $|S_{\mathbf{a}_0}|$ is equal to $|S_{\mathbf{a}_{-1}}|$ and is governed by $VCdim(\mathbb{H})$. When considering many tasks this is a reasonable assumption. There are also cases when this step can be avoided see

example in Fig. 3.2.

Theorem 3 states the generalization error for CMT-ERM when applied to the hierarchically decomposed multi-task learning setting:

Theorem 3 Let $\{\mathcal{T}_l, \mathcal{F}_l, \mathcal{G}_l\}_{l=0}^{L}$ be a hierarchical decomposition of a set of $\mathcal{F}$-related tasks for which the cascade assumptions hold. Let $h^* = \arg\min_{h \in \mathbb{H}} Er^{P_1}(h)$ and $h^\diamond$ the output of the CMT-ERM algorithm. For $l = -1..L$, let $d_l = VCdim([h]_{\sim_{\mathcal{G}_l}})$, ϵ_l the objective generalization error for each step in the algorithm, $\epsilon = 2\sum_{l=-1}^{L} \epsilon_l$ and $|S_{\mathbf{a}_l}|$ the sample size available at each step of the algorithm. If $\delta > 0$ and for every $l = -1..L$, $|S_{\mathbf{a}_l}| \geq c_0(\frac{1}{\epsilon_l} log \frac{L}{\delta} + \frac{d_l}{\epsilon_l} log \frac{1}{\epsilon_l})$, with probability greater than $(1 - \delta)$

$$Er^{P_1}(h^\diamond) \leq Er^{P_1}(h^*) + \epsilon$$

Proof The proof goes by induction on L. In our setting we are concerned with at least one hierarchal level. For $L = 1$ we have three learning steps, choosing from $[h]_{\sim_{\mathcal{G}_{-1}}}$, $[h]_{\sim_{\mathcal{G}_0}}$ and $[h]_{\sim_{\mathcal{G}_1}}$. From Theorem 2 we know that given sample sets from all tasks in each level with the size of each sample set $|S_{\mathbf{a}_l}|$ defined as above, we obtain a generalization error of at most $2\epsilon_l$ when choosing an hypothesis from $[h]_{\sim_{\mathcal{G}_l}}$. Putting the three errors together we get an error of at most $2\epsilon_{-1} + 2\epsilon_0 + 2\epsilon_1$.

The induction step is straightforward given Theorem 2. Increasing the hierarchy size L by adding level l adds a generalization error of at most $2\epsilon_l$. □

3.4 Experiment

In order to demonstrate when our proposed framework achieves improved performance, we tested it in a controlled manner on a synthetic dataset we had created. For that we consider the hypothesis class of conjunction classifiers ([74, 75]). We assume an input space of boolean features. Each classifier in this hypothesis class is expressed as a conjunction of a subset of the input boolean features[1].

The set of transformations $\mathcal{F}$ are all possible feature swaps of length L, denoted by $\mathbf{i}_1...\mathbf{i}_L$ - $\mathbf{j}_1...\mathbf{j}_L$. A feature swap of length L is defined as swapping the values of

[1]More sophisticated conjunction classifiers over boolean functions can be used ([76, 77]), but this goes beyond the scope of the current presentation.

the features indexed by $\mathbf{i}_1...\mathbf{i}_L$ with the values of the features $\mathbf{j}_1...\mathbf{j}_L$. For example, a swap transformation $1,3$ - $2,4$ applied to the binary sample 0101 will result in the transformed binary sample 1010. We decompose the set of transformations $\mathcal{F}$ in the following way: for each $0 \leq l \leq L$, $\mathcal{F}_l$ corresponds to all swaps of length $L-l$ among the features that were not swapped yet. $\mathcal{G}_{l+1}$ corresponds to all swaps of length 1 among the non swapped features. It is easy to see that the sets of transformations- $\mathcal{F}$,$\mathcal{F}_l$ and $\mathcal{G}_{l+1}$ indeed follow definitions 1 and 2 in [59]. We note that each swap of length $L-l$ for $0 \leq l \leq L$ can be written as a swap of size L with l self swaps.

Same as for the example in Fig. 3.2 we consider the first hypothesis to be the hypothesis which accepts all samples. With the above notations we do this by concatenating L features valued '1' to all samples (both positive and negative) in the training set. The first hypothesis is the conjunction of these added L features.

We note that choosing L features for a conjunction classifier is clearly equivalent to search for a swap of length L given the defined conjunction over the L dummy features concatenated to the original feature representation. We also note that practically the search over the swap operations can be done on smaller search space considering only swaps between the original features and the concatenated L features.

Synthetic dataset The data defines a group of tasks related in a hierarchical manner: the features correspond to nodes in a tree-like structure, and the number of tasks sharing each feature decreases with the distance of the feature node from the tree root. The input to the algorithm includes the sets of examples and labels from all n tasks $\{\mathbf{S}^i\}_{i=1}^n$, as well as the hierarchical grouping of tasks.

More specifically, the data is created in the following manner: we define a binary tree with n leaves. Each leaf in the tree represents a single binary classification task. Each node in the tree corresponds to a single binary feature $f \in \{0,1\}$. For a single task we divide the features into two groups: task-specific and task-irrelevant. For positive examples the task-irrelevant features have equal probability of taking the value 1 or 0, whereas all the task-specific features are assigned the value 1. The negative examples have equal probability of having the value 1 or 0 for all features.

The task-specific feature set is the set of features corresponding to all nodes on the path from the leaf to the root, while all other nodes define the task-irrelevant

feature set. For each group of tasks, their shared features are those corresponding to common ancestors of the corresponding leaves.

We search for conjunctions of L features.[1]

For each leaf task, we consider the set $\{\mathcal{T}_l\}_{l=0}^{L}$ to correspond to the grouping of tasks represented by the path from the leaf to the root in the synthetic dataset. $\mathcal{T}_0$ corresponds to the root node representing all tasks. $\mathcal{T}_L$ corresponds to the leaf node representing the single objective task. The triplet $\{\mathcal{T}_l, \mathcal{F}_l, \mathcal{G}_l\}_{l=0}^{L}$ clearly obeys definition 1 being a hierarchical decomposition of a set of $\mathcal{F}$-related tasks.

We defined each set of samples $S_{a_l} \in S_{a_0}, .., S_{a_L}$ to correspond to the union of positive and negative samples from all tasks represented by the l'th node on the path down from the root to the target task L.

The general cascade algorithm (definition 3) can be re-written for this specific choice of $\mathbb{H}$ and $\mathcal{G}$ as follows:

1. Set h as the conjunction of the first L features.
2. Init Z the set of all features indices.
3. for $l = 0..L$
   (a) find optimal $h \in [h]_{\sim_{\mathcal{G}_l}}$:
      i. Swap the $l+1$ feature with a feature among all features in Z with index greater than $L+1$ that minimizes $\hat{Er}^{S_{a_l}}(h)$
      ii. Remove this feature from Z
4. output $h^{\diamond} = h$.

We now claim that for this specific experimental setting, Assumptions 1 and 2 of theorem 1 hold.

Claim 1 Assumption 1 in theorem 1 holds.

The multiplicative property is derived directly from the definition of conjunction of non overlapping sets of binary features.

[1]The assumption that all tasks have exactly L features used by their optimal classifier is valid in case we consider balanced binary trees of hight L for creating the synthetic data.

Claim 2 Assumption 2 in theorem 1 holds.

First we note that by construction the features are statistically independent given the label. Now let us consider the optimal transformation $g^* \in \mathcal{G}$. Each feature of a negative example gets its value at random uniformly. Thus any choice of feature will give the same false positive rate. On the other hand the optimal choice of feature will correctly classify all positive examples. From this we can conclude that for both types of errors the optimal choice of transformation g^* is always less than or equal to the error of any other choice of transformation. Thus together with the statistical independence property we can conclude that the indifference property holds (see Appendix for further discussion) and Assumption 2 is valid.

Results Figure 3.3 shows the performance of our approach compared to standard baselines. First we consider the original two step approach for learning conjunction classifiers, the Valiant step and Hassler step ([74, 75]) denoted by 'VHConj'. For the 'VHConj' approach each task is trained on it own. As a multi-task baseline we consider the popular approach for training jointly a set of linear classifiers, by minimizing a loss function (hinge loss) together with a group-lasso regularization term. This approach is denoted by 'LinReg-L12'.

We consider a hierarchy of tasks with 64 individual tasks. We test our approach on each task, measuring performance as a function of sample size. We show the mean classification accuracy of all tasks averaged over 5 repetitions of the experiment.

With very few samples a conjunction classifier searching for exactly L features is too restrictive as we cannot assume enough support to discover correctly *all* features. To deal with such scenarios we consider two classifiers- the original conjunction classifier using all the discovered features denoted by 'FullCascadeConj' and a more robust version which classifies as positive if $L-1$ features take the value '1', and denoted by 'PartialCascadeConj'. We note that the 'VConj' is not restricted to a pre-set number of features, thus this heuristic does not need to be applied to it.

We clearly see that the learning benefits much from our cascade approach. The cascade conjunction approach outperforms significantly both the original conjunction baseline and the multi-task learning baseline. As expected for the very small sample scenario the 'PartialCascadeConj' performs best. For larger samples 'FullCascadeConj' improves significantly. We were able to obtain the same results with-

out assuming prior knowledge of the hierarchy, using the SIPO algorithm (described in the next section) to discover it automatically.

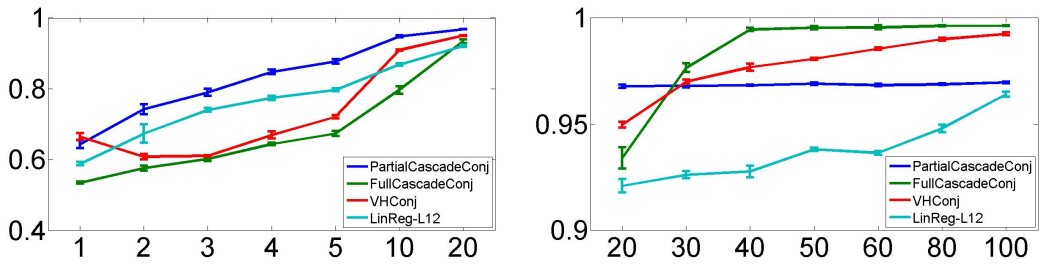


Figure 3.3: Accuracy results for the synthetic dataset experiment. 'Y-axis' corresponds to accuracy. 'X-axis' corresponds to sample size of the positive set (negative set has the same size). Left plot corresponds to a small sample scenario. Right plot corresponds to a larger sample size scenario. 'PartialCascadeConj'- denotes the classifier classifying as positive if $L-1$ features take the value '1'; 'FullCascadeConj' denotes the original conjunction classifier using all the discovered features; 'VHConj' denotes the two step approach based on Valiant's step and Hassler's step; 'LinReg-L12' corresponds to the multi-task baseline where training is done using the group-lasso regularization term.

3. HIERARCHICAL MULTI-TASK LEARNING: A CASCADE APPROACH BASED ON THE NOTION OF TASK RELATEDNESS

4

SIPO: Set Intersection Partial Order

In chapters 2 and 3 we presented scenarios where knowing a hierarchical structure over a set of predefined concepts (e.g. object categories) proves to be beneficial- the novel subclass detection and multi-task learning scenarios. Clearly it is beneficial to know a hierarchical structure of concepts. But, how can we discover it from data when it is not known in advance?

In this chapter we present a general hierarchical model of tasks and an algorithm to infer it from data. The model captures both part-membership (conjunctive) and class-membership (disjunctive) hierarchies. We provide an algorithm for discovering hierarchical structure in data. We start by assuming a binary set representation of concepts (tasks). Each task is represented by a unique set of properties. Then, we extend the model to a statistical setting.

The model captures both the part-membership and class-membership hierarchies in a single hierarchy. Given a representation of an object class as a set of properties, a grouping of several classes into a single more abstract class defines a co-occurring set of properties shared by these classes. On the other hand a set of co-occurring properties defines a grouping of one or more classes sharing these properties. The connection between the two different hierarchies is achieved by the notion of partial order. A hierarchy of classes defines a partial order among the classes, while inclusion relations between sets of properties define partial order over these sets.

In section 4.1 we present the model, a partial order graph representation unifying both the class membership and part membership hierarchies. In section 4.2 we describe the algorithm constructing the hierarchical representation based on given data. In section 4.3 we describe a statistical extension of the model- from the case where each task is represented by a set of properties to the case where each task is represented by a set of samples and each sample is represented by a set of properties. Such a representation fits the regular learning scenario, where learning a concept is

typically done from examples representing the specific concept. In 4.4 we present promising experimental results showing that the statistical SIPO algorithm can be used to discover the features of a conjunction classifier in a hierarchical setting of learning tasks.

4.1 Partial Order Representation

Our hierarchical model is a partial order model capturing both the part-membership (*conjunctive concepts*) and class-membership hierarchies (*disjunctive concepts*) 2.1.1. In the following we regard the part-membership hierarchy as a property co-occurrences hierarchy. We view the task of finding an object class hierarchy as finding commonalities among object classes, see section 4.2. Finding such commonalities can be thought of as computing the intersection of the sets of properties representing each class. The set of properties representation is a general representation which can be reduced to many of the recent object class representation approaches. This line of thought lead us to propose the following dual representation model:

Given a finite set of properties $\mathbf{P} = \{\mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{e}...\}$ and a matching set of boolean functions (predicates) $\theta = \{\psi_{\mathbf{x}} | \mathbf{x} \in \mathbf{P}\}$ each denoting whether sample $\mathbf{s}$ contains a property, thus: $\psi_{\mathbf{a}}(\mathbf{s}) = 1$ iff the property $\mathbf{a}$ is present in $\mathbf{s}$. We represent a single class by the set of properties $\mathbf{P_C}$ which each instance in this class always contains, thus $\mathbf{s} \in \mathbf{C}$ iff $\psi_{\mathbf{x}}(\mathbf{s}) = 1 \ \forall \mathbf{x} \in \mathbf{P_C}$. This can also be expressed as a boolean function which is a conjunction of all property functions, thus $\mathbf{s} \in \mathbf{C}$ iff $\psi^{\mathbf{C}}(\mathbf{s}) = 1$ where $\psi^{\mathbf{C}}(\mathbf{s}) = \bigwedge_{\mathbf{x} \in \mathbf{P_C}} \psi_{\mathbf{x}}(\mathbf{s})$; we call $\psi^{\mathbf{C}}(\mathbf{s})$ the membership function of class $\mathbf{C}$.

In the class-membership hierarchy we say that $\mathbf{B} \succeq \mathbf{A}$ iff $\forall \mathbf{s} \in \mathbf{A} \Rightarrow \mathbf{s} \in \mathbf{B}$ in the part-membership hierarchy, or equivalently we say $\mathbf{B} \succeq \mathbf{A}$ iff $\forall \mathbf{x} \in \mathbf{P_B} \Rightarrow \mathbf{x} \in \mathbf{P_A}$. Given our representation of classes as sets of properties, one can easily see that defining the specific level class representation as the union of set of properties of all its general level classes is equivalent to setting the samples belonging to a specific level class to the intersection of the set of samples belonging to all of its general level classes. This results in a partial order which is consistent with both class-membership and part-membership hierarchies.

We note that by representing a specific class $\mathbf{C}$ as the union of the set of properties of its more general classes $\mathbf{P}a_{\mathbf{C}}$ we get that $\psi^{\mathbf{C}}(\mathbf{s}) = \bigwedge_{\mathbf{p} \in \mathbf{P}a_{\mathbf{C}}} \bigwedge_{\mathbf{x} \in \mathbf{p}} \psi_{\mathbf{x}}(\mathbf{s})$, thus

the membership function of class $\mathbf{C}$ is the conjunction of the membership functions of classes $\mathbf{P}a_{\mathbf{C}}$. This fits our intuitive example of part-membership hierarchy from above - if something has a leg, tail and head (conjunctions) than it is a dog, but just knowing that there is a leg does not imply that there is a dog. Thus the conjunction of the part detections is more specific than each separate part detection. Even more, given a set of specific classes which we know share a common general class, we can represent the general class using the intersection of property sets of all its specific classes, thus we can compute the membership function. On the other hand, even if we know the identity of all samples which belong to its specific sub-classes, we cannot deduce the identity of all samples which belong to the general class, since (from our definition) the union of all the samples belonging to the specific classes is merely contained in the set of samples belonging to the general class[1]. Thus the equality in $DOG = AFGHAN \cup BEAGEL \cup COLLIE$ holds only in a fully observable world assumption where we know all categories.

Based on the set of properties representation of a class and the definition above for specific and general levels we represent the partial order using a graph $\mathbf{G} =< \mathbf{V}, \mathbf{E} >$. Each node $\mathbf{v} \in \mathbf{V}$ represents a class and is associated with two sets: $\mathbf{R_v}$ - the property set which is the class representation, and $\mathbf{I_v}$ - all instances belonging to the class represented by node $\mathbf{v}$. Each edge $(\mathbf{v}, \mathbf{u}) \in \mathbf{E}$ indicates that v implies u, thus $\mathbf{u} \succeq \mathbf{v}$. Let $\mathbf{C}h_v$ denote the set of children of node $\mathbf{v}$, then:

1. $\mathbf{R_v} = \cup_{\mathbf{c} \in \mathbf{C}h_v} \mathbf{R_c}$
2. $\mathbf{I_v} = \cap_{\mathbf{c} \in \mathbf{C}h_v} \mathbf{I_c}$

When $\mathbf{C}h_v = \emptyset$:

1. $\mathbf{R_v} = \{\mathbf{x}\}$, where $\mathbf{x}$ is a single property from the property set.
2. $\mathbf{I_v} = \{\mathbf{s} | \psi_{\mathbf{x}}(\mathbf{s}) = 1\}$

Note, that as the representation of a node is based on the union of the representations of its children, hence for each property used to represent at least one of the classes there has to be a single node represented by this property alone. Let's denote the special set of nodes for which $\mathbf{C}h_v = \emptyset$ as $\mathbf{Layer}_1$ and the set of nodes representing all known classes we call $\mathbf{Layer}_2$. We note that $\mathbf{Layer}_1 \cup \mathbf{Layer}_2 \subseteq \mathbf{V}$.

[1] This point is the basis of the novel class detection approach presented in 2.

4.1.1 Compactness Constraints

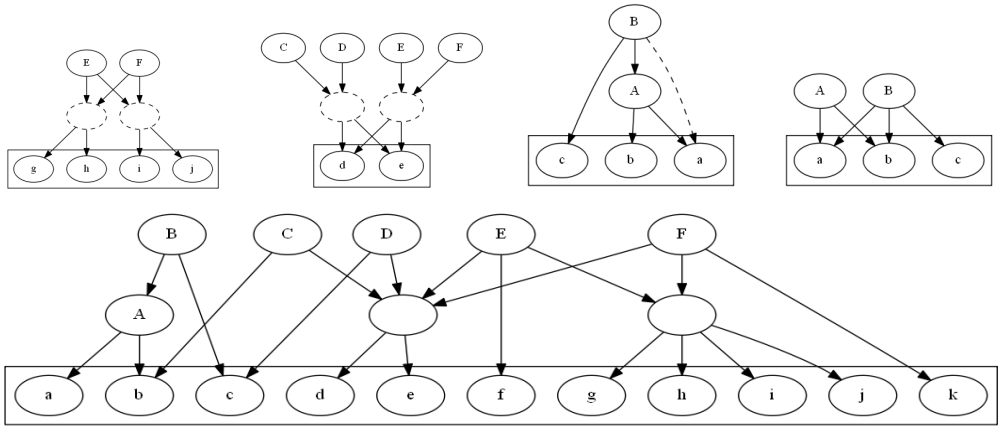


Figure 4.1: Possible graphical structures for a given set of properties: $\mathbf{P} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}, \mathbf{g}, \mathbf{h}, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$, and a set of labeled classes $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}\}$. Each class has the following sub set of properties: $\mathbf{P_A} = \{\mathbf{a}, \mathbf{b}\}$, $\mathbf{P_B} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$, $\mathbf{P_C} = \{\mathbf{b}, \mathbf{d}, \mathbf{e}\}$, $\mathbf{P_D} = \{\mathbf{c}, \mathbf{d}, \mathbf{e}\}$, $\mathbf{P_E} = \{\mathbf{d}, \mathbf{e}, \mathbf{f}, \mathbf{g}, \mathbf{h}, \mathbf{i}, \mathbf{j}\}$ and $\mathbf{P_F} = \{\mathbf{d}, \mathbf{e}, \mathbf{g}, \mathbf{h}, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$. $\mathbf{Layer}_1$ nodes are labeled with lower case. $\mathbf{Layer}_2$ nodes are labeled with upper case. Dashed lines (nodes and edges) represent constraint violations. Bottom row shows a graph representation of the data which satisfies all constraints. Top row shows possible subgraphs each has a single constraint violated- 4.2, 4.3, 4.5 or 4.4 from left to right respectively.

For a given set of known classes and properties the description so far does not describe a unique graph representation, as shown in figure 4.1. From all possible graphs representing a legal partial order, we define the following constraints which define a compact graph:

1. Data Consistency: Given a set of properties and a set of classes, we would like the graph representation to maintain the known relation and not add any new ones.

$$\forall \mathbf{u} \in \mathbf{Layer}_1 \; and \; \mathbf{v} \in \mathbf{Layer}_2 \;\; \exists path(\mathbf{v}, \mathbf{u}) \Leftrightarrow \mathbf{R_u} \subseteq \mathbf{P_v} \tag{4.1}$$

2. Vertex Minimality Constraints: from the representation point of view, there are two types of redundant vertices - those that represent the same class and those that have the same property representation. First constraint - no two

vertices may represent the same class:

$$\neg\exists \mathbf{s}, \mathbf{v} \in \mathbf{V} \; such \; that$$
$$\{\mathbf{u} : path(u, s) \subset \mathbf{E}, \mathbf{u} \in \mathbf{Layer}_2\} = \{\mathbf{u} : path(u, v) \subset \mathbf{E}, \mathbf{u} \in \mathbf{Layer}_2\} \quad (4.2)$$

Second constraint - no two vertices may have the same representation:

$$\neg\exists \mathbf{s}, \mathbf{v} \in \mathbf{V} \; such \; that$$
$$\{\mathbf{u} : path(s, u) \subset \mathbf{E}, \mathbf{u} \in \mathbf{Layer}_1\} = \{\mathbf{u} : path(v, u) \subset \mathbf{E}, \mathbf{u} \in \mathbf{Layer}_1\} \quad (4.3)$$

3. Maximal order: We would like to represent all order relations within the set of known classes.

   For example in Figure 4.1, all graphs represent a valid partial order, but the rightmost graph in the top line does not show a maximal partial order representation as there is no edge going from **B** to **A** so we cannot deduce from the graph that $\mathbf{A} \succeq \mathbf{B}$, while the bottom graph does show this relation. Thus when presented with a sample from **B** represented by $\mathbf{P_B} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ there will be two satisfied nodes **A** and **B** where neither one is more specific than the other.

$$\{\mathbf{v} : path(s, v) \in \mathbf{E}, \mathbf{v} \in \mathbf{Layer}_1\} \subset \{\mathbf{v} : path(u, v) \in \mathbf{E}, \mathbf{v} \in \mathbf{Layer}_1\} \Leftrightarrow$$
$$path(\mathbf{u}, \mathbf{s}) \subset \mathbf{E} \quad (4.4)$$

4. Edge Minimality Constraint: Let $\mathbf{v}^a$ denote the node represented by only one property **a**, and $\mathbf{v}^{ab}$, $\mathbf{v}^{ac}$ and $\mathbf{v}^{abc}$ denote classes represented by the sets of properties $\{\mathbf{a}, \mathbf{b}\}$, $\{\mathbf{a}, \mathbf{c}\}$ and $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ respectively. A possible graph would have the following edges: $(\mathbf{v}^{abc}, \mathbf{v}^{ab})$,$(\mathbf{v}^{abc}, \mathbf{v}^{ac})$,$(\mathbf{v}^{abc}, \mathbf{v}^{a})$. These edges are redundant in the sense that $(\mathbf{v}^{abc}, \mathbf{v}^{a})$ can be removed without effecting the representation of $\mathbf{v}^{abc}$ or violating the maximal order constraint.

$$\neg\exists \mathbf{e} = (\mathbf{u}, \mathbf{v}) \in \mathbf{E} \; such \; that \; \mathbf{G}^* =< \mathbf{V}, \mathbf{E} \backslash \mathbf{e} > \quad (4.5)$$
$$maintains \; the \; maximal \; order \; constraint \; and \; \mathbf{R}^*_u = \mathbf{R}_u$$

4.2 Hierarchy Discovery Algorithm

In our above discussion we dealt with representing the partial order given a set of classes. In this section we deal with two problems:

a) Not all possible labels in the partial order are given in advance.

b) How to find this partial order among a set of given classes.

How do we deal with the case where not all possible labels in the partial order are given in advance? For example an implicit description of the general level 'Motorbike' class can be given by more specific motorbikes such as 'Cross-Motorbike' and 'Sport-Motorbike', without stating explicitly that both belong to the same general level class 'Motorbike'. In such a case where a general class is not stated explicitly we'll refer to it as a *hidden general class*. In the following we will deal with discovering all the possible hidden general classes by finding commonalities among known given classes. Given any two classes we say these classes have a general class in common if the intersection of the feature sets representing the two classes is not empty. Formally, We say a class $\mathbf{C}$ is a hidden general class with respect to a given set of classes Γ iff $\exists \mathbf{A}, \mathbf{B} \in \Gamma$ such that $\mathbf{P_C} \neq \emptyset$ and $\mathbf{P_C} = \mathbf{P_A} \cap \mathbf{P_B}$.

Under this definition finding all hidden general classes requires that we find all possible intersections of representation sets between all subsets of known classes.

We propose algorithm 3 for finding all possible hidden general level classes while simultaneously building a graph representation $\mathbf{G}^{out} =< \mathbf{V}^{out}, \mathbf{E}^{out} >$ consistent with the proposed graph partial order representation. In order to do so we start with an initial graph representation of a given set of classes which is consistent with the above representation. For each property in $\mathbf{x} \in \cup_{\mathbf{c} \in \Gamma} \mathbf{P}_c$ we define a vertex $\mathbf{v}, \mathbf{R_v} = \{\mathbf{x}\}$ *and* $\mathbf{C}h_v = \emptyset$. For each class $\mathbf{C} \in \Gamma$ we define a vertex $\mathbf{u}$ with a set of outgoing edges denoted $\mathbf{O}\hat{ut}(u)$, where for each $\mathbf{x} \in \mathbf{P_C}$ there is a corresponding edge $(\mathbf{u}, \mathbf{v}) \in \mathbf{O}\hat{ut}(u)$. We denote the vertices corresponding to properties as $\mathbf{Layer}_1$ and the vertices corresponding to classes as $\mathbf{Layer}_2$. Thus the input graph $\mathbf{G}^{in} =< \mathbf{V}^{in}, \mathbf{E}^{in} >$ is defined by $\mathbf{V}^{in} = \mathbf{Layer}_1 \cup \mathbf{Layer}_2$ and $\mathbf{E}^{in} = \cup_{\mathbf{u} \in \mathbf{Layer}_2} \mathbf{O}\hat{ut}(u)$. We assume for now that in a given set of classes Γ there aren't any two class $\mathbf{A}, \mathbf{B} \in \Gamma$ where $\mathbf{P_A} \subseteq \mathbf{P_B}$ or $\mathbf{P_B} \subseteq \mathbf{P_A}$.

Now we shall describe the four basic operation carried out by the algorithm: Split, Forward-Merge, Backward-Merge and Maximal-Ordering. These four operations enable the creation of new nodes based on the set of vertices connected by outgoing edges $\mathbf{O}ut(u)$ and the set of vertices connected by incoming edges $\mathbf{I}n(u)$ of each vertex $\mathbf{u} \in \mathbf{V}$.

Split(s,in(s)): The split operation creates a new node for each pair of incoming edges in $\mathbf{in}(\mathbf{s}) \subset \mathbf{I}n(s)$ of node $\mathbf{s} \in \mathbf{V}$. Intuitively this helps us mark an intersection between the representation of two nodes as equivalent to $\mathbf{R_s}$ (the representation of node $\mathbf{s}$).

Formally:

$\forall \mathbf{u}, \mathbf{v} \in \mathbf{V}$ such that $\{\mathbf{u}, \mathbf{v}\} \in \mathbf{in}(\mathbf{s})$ do:

Create a new node $\mathbf{t}$ with

- $\mathbf{O}ut(t) = \{\mathbf{s}\}$
- $\mathbf{I}n(t) = \{\mathbf{u}, \mathbf{v}\}$

Forward-Merge(U): The Forward-Merge operation merges all nodes in a specific set of nodes $\mathbf{U}$ which share the same incoming edges. Intuitively by doing the merge operation after the split operation we find the maximum set of intersection between the representation of any two vertices. This operation is essential for maintaining the first vertex minimality constraint (eq. 4.2), verifying that no two vertices may represent the same class. Let ${\mathbf{E}_{in}}^i$ denote a set of nodes which all share the same parent nodes. Thus, $\forall \mathbf{u}, \mathbf{v} \in {\mathbf{E}_{in}}^i$ $\mathbf{I}n(u) \equiv \mathbf{I}n(v)$. Let $\mathbf{E}_{in}$ denote the group of maximal equivalence sets of nodes according to the incoming edges. Thus, $\forall \mathbf{u} \in \mathbf{U}, \mathbf{u} \in \cup_i {\mathbf{E}_{in}}^i$ and $\forall {\mathbf{E}_{in}}^i, {\mathbf{E}_{in}}^j \in \mathbf{E}_{in}, i \neq j \Leftrightarrow \forall \mathbf{u} \in {\mathbf{E}_{in}}^i$ and $\forall \mathbf{v} \in {\mathbf{E}_{in}}^j$ $\mathbf{I}n(u) \neq \mathbf{I}n(v)$.

Formally:

1. Compute the group of $\mathbf{E}_{in}$ over $\mathbf{U}$.
2. $\forall {\mathbf{E}_{in}}^i \in \mathbf{E}_{in}$ if $|{\mathbf{E}_{in}}^i| > 1$

   Create a new node $\mathbf{n}$ with

   - $\mathbf{O}ut(n) = \{\mathbf{s} | \mathbf{s} \in \mathbf{O}ut(u)\ \forall \mathbf{u} \in {\mathbf{E}_{in}}^i\}$

- $\mathbf{I}n(n) = \mathbf{I}n(u)$ where $\mathbf{u} \in \mathbf{E}_{in}{}^{i}$

As **Forward-Merge(U)** is done sequentially after the **Split(s,in(s))** operation, each node $\mathbf{u} \in \mathbf{U}$ has exactly two incoming edges- $\{(\mathbf{x}, \mathbf{u}), (\mathbf{y}, \mathbf{u})\}$, where $\mathbf{x}, \mathbf{y} \in \cup_{\mathbf{s}}\mathbf{in}(\mathbf{s})$. We can compute $\mathbf{E}_{in}$ using the algorithm described in A.2.1 after sorting the two incoming edges to each node, given some predefined order over all possible incoming edges. As input to the algorithm in A.2.1 we pass $\mathbf{U}$ as the group of sets, and $\cup_{\mathbf{s}}\mathbf{in}(\mathbf{s})$ as the group of possible elements in each set. Thus the runtime of **Foward-Merge(U)** is $O(|\cup_{\mathbf{s}} \mathbf{in}(\mathbf{s})| + |\mathbf{U}|)$.

Backward-Merge(U): The Backward-Merge operation merges all nodes in a specific set of nodes $\mathbf{U}$ which share the same outgoing edges. Intuitively by doing the Backward-Merge operation after the Forward-Merge we find the maximal set of nodes which share the same representation. This operation is essential for maintaining the second vertex minimality constraint (eq. 4.3), no two vertices may have the same representation. Let $\mathbf{E}_{out}{}^{i}$ denote a set of nodes which all share the same child nodes. Thus, $\forall \mathbf{u}, \mathbf{v} \in \mathbf{E}_{out}{}^{i}$ $\mathbf{O}ut(u) \equiv \mathbf{O}ut(v)$. Let $\mathbf{E}_{out}$ denote the group of maximal equivalence sets of nodes according to the outgoing edges. Thus, $\forall \mathbf{u} \in \mathbf{U}, \mathbf{u} \in \cup_i \mathbf{E}_{out}{}^{i}$ and $\forall \mathbf{E}_{out}{}^{i}, \mathbf{E}_{out}{}^{j} \in \mathbf{E}_{out}, i \neq j \Leftrightarrow$ $\forall \mathbf{u} \in \mathbf{E}_{out}{}^{i}$ and $\forall \mathbf{v} \in \mathbf{E}_{out}{}^{j}$ $\mathbf{O}ut(u) \neq \mathbf{O}ut(v)$.

Formally:

1. Compute the group of $\mathbf{E}_{out}$ over $\mathbf{U}$.
2. $\forall \mathbf{E}_{out}{}^{i} \in \mathbf{E}_{out}$

   Create a new node $\mathbf{n}$ with

   - $\mathbf{O}ut(n) = \mathbf{O}ut(u)$ where $\mathbf{u} \in \mathbf{E}_{out}{}^{i}$
   - $\mathbf{I}n(n) = \{\mathbf{s} | \mathbf{s} \in \mathbf{I}n(u) \, \forall \mathbf{u} \in \mathbf{E}_{out}{}^{i}\}$

For the computation of $\mathbf{E}_{out}$ we can again use the algorithm described in A.2.1, where $\mathbf{U}$ is the group of sets and $\cup_{\mathbf{s}-was-split}\mathbf{s}$ is the group of possible elements in each set (outgoing edges). As mentioned in A.2.1 for each node $\mathbf{u} \in \mathbf{U}$ the elements (outgoing edges) should be sorted. Contrary to the computation of $\mathbf{E}_{in}$ in the **Forward-Merge(U)** where each node has exactly two elements (incoming edges) and thus sorting the elements of each node can be done in

constant time, the size of the possible elements considered for each node at this stage can be at most $|\cup_{\mathbf{s}-was-split}\mathbf{s}|$. In order to overcome the need for sorting outgoing edges of each node at this stage we would like to keep the $\mathbf{O}ut(u)$ of each node, ordered according to some chosen order of the original $\mathbf{s}$ nodes. This can be accomplished easily by keeping each $\mathbf{E}_{in}{}^{i}$ during the **Forward-Merge(U)** ordered according to the order of **Split(s,in(s))**. The runtime of **Backward-Merge(U)** is $O(|\cup_{\mathbf{s}-was-split}\mathbf{s}| + \sum_{\mathbf{u}}|\mathbf{O}ut(u)|)$.

EdgeMin(U): Given a set of nodes $\mathbf{U}$ it may be the case that $\exists \mathbf{u},\mathbf{v} \in \mathbf{U}$ for which $\mathbf{R_u} \subset \mathbf{R_v}$. Under our partial order interpretation of the graph and in order to maintain the maximal order and edge minimality constraints (eq. 4.4 and 4.5) we would like to connect by an edge each node to the **most** specific node representing a more general level. When doing so we would like to delete all edges to nodes more general than the newly connected node. This can be achieved by checking for each pair of new nodes $\mathbf{u},\mathbf{v} \in \mathbf{U}$ if $\mathbf{R_u} \subset \mathbf{R_v}$ (w.l.o.g). In such a case all edges connecting $\mathbf{u}$ to nodes in the intersection of $\mathbf{I}n(u)$ and $\mathbf{I}n(v)$ should be deleted. This will ensure that $\forall \mathbf{s} \in \mathbf{I}n(u) \cap \mathbf{I}n(v)$ (for which $\mathbf{R_u} \subset \mathbf{R_v} \subset \mathbf{R_s}$) $\mathbf{s}$ will be connected to the most specific general node, $\mathbf{v}$. Hence we achieve the maximal ordering (eq. 4.4) by maintaining the connectivity between $\mathbf{s}$ to $\mathbf{v}$ and $\mathbf{u}$, and the edge minimality (e. 4.5) by deleting redundant edges- $(\mathbf{s},\mathbf{u})$.

Formally:

1. for each $\mathbf{p} \in \cup_{\mathbf{s}-was-split}\mathbf{s}$ create a list $L_{\mathbf{p}}$
2. for each $\mathbf{u}^i \in \mathbf{U}$ go over all $\mathbf{p} \in \mathbf{O}ut(u^i)$, and add i to $L_{\mathbf{p}}$.
3. for each i choose $\mathbf{p}^* = arg\,min_{\mathbf{p}\in\mathbf{O}ut(u^i)}\,|L_{\mathbf{p}}|$
   (a) for each $\mathbf{p} \in \mathbf{O}ut(u^i)$ mark all $j \neq i \in L_{\mathbf{p}}*$ which appear in $L_{\mathbf{p}}$.
      i. if j appears in all $L_{\mathbf{p}}$, conclude that $\mathbf{O}ut(u^i) \subseteq \mathbf{O}ut(u^j)$
4. for each j such that $\mathbf{O}ut(u^i) \subseteq \mathbf{O}ut(u^j)$ do $\mathbf{I}n(u^i) = \mathbf{I}n(u^i) \backslash \mathbf{I}n(u^i) \cap \mathbf{I}n(u^j)$

Note that in the following algorithm EdgeMin($\mathbf{U}$) is preformed sequentially after Backward-Merge($\mathbf{U}$), and thus for any two $\mathbf{u}^i,\mathbf{u}^j \in \mathbf{U}$ there cannot be an identity between $\mathbf{O}ut(u^i)$ and $\mathbf{O}ut(u^j)$.

Initializing all $L_{\mathbf{p}}$ takes $O(|\cup_{\mathbf{s}-was-split}\mathbf{s}|)$. Updating all $L_{\mathbf{p}}$ for all $\mathbf{u}^i \in \mathbf{U}$ takes $O(\sum_i |\mathbf{O}ut(u^i)|)$. We assume a predefined order over $\mathbf{u}^i \in \mathbf{U}$. Finding $\mathbf{p}^*$ for each i takes $O(|\mathbf{O}ut(u^i)|)$. Going over all other $\mathbf{p} \in \mathbf{O}ut(u^i)$ and checking for each $j \neq i \in L_{\mathbf{p}}*$ if it appears in $L_{\mathbf{p}}$ takes $O(\sum_{\mathbf{p}\in\mathbf{O}ut(u^i)} |L_{\mathbf{p}}|)$; this can be achieved as we assume that $L_{\mathbf{p}}*$ and $L_{\mathbf{p}}$ are ordered, so comparing both lists can be done in a single pass with runtime $O(|L_{\mathbf{p}}|)$ as $|L_{\mathbf{p}}*| < |L_{\mathbf{p}}|$. Thus, finding all containment relations among all $\mathbf{u}^i \in \mathbf{U}$ takes $O(\sum_i |\mathbf{O}ut(u^i)| + \sum_i \sum_{\mathbf{p}\in\mathbf{O}ut(u^i)} |L_{\mathbf{p}}|)$. Finding and deleting the edges in the intersection of both nodes, step 4, can take $O(|\mathbf{I}n(u^i)|+|\mathbf{I}n(u^j)|)$. Given a hash table of size $|\cup_i \mathbf{I}n(u^i)|$ (we do not assume $\mathbf{I}n(u)$ is ordered). Thus, in total we can conclude that the runtime of the EdgeMin operation is $O(\sum_i |\mathbf{O}ut(u^i)| + \sum_i \sum_{\mathbf{p}\in\mathbf{O}ut(u^i)} |L_{\mathbf{p}}| + |\mathbf{I}n(u^i)| + |\mathbf{I}n(u^j)|)$.

The Algorithm is summarized below in Alg 3.

4.2.1 Algorithm Analysis

In this section we highlight the main guarantees of algorithm 3. A detailed analysis is given in appendix A.2.2. Algorithm 3 discovers all non empty intersections and produces a graphical representation which obeys all compactness constraints. We calculate the runtime of the algorithm as a function of the size of the output- the number of all non empty intersections. We note that in case the data is not sparse in the number of non empty intersections the worst case analysis results in an exponential runtime (computing the power set of the input classes). For the analysis we denote by Γ^{in} the set of all classes in the input. Ω denotes the group of all possible non empty intersections with a unique set of features. $\hat{\mathbf{c}_i}^k \in \Omega$ denotes a group of classes with k features in their intersection. $\Delta_k \subseteq \Omega$ denotes the group of intersections with k features. We also define the following measure- $\hat{\mathbf{Dl}} = \sum_{k=1}^{n-1} \mathbf{k}\#_k$ where $\#_k = \sum_{\hat{\mathbf{c}_i}^k \in \Delta_k} |\hat{\mathbf{c}_i}^k|^2$. Theorem 1 states that all constraints are kept. Theorem 2 states that all non empty intersections are discovered. Theorem 3 presents the runtime which is governed by a quadratic term in the number of classes $|\hat{\mathbf{c}_i}^k|^2$ of each of the existing intersections in the data.

Algorithm 3 SIPO: Set Intersection Partial Order

Input :

$\mathbf{G}^{in} =< \mathbf{V}^{in}, \mathbf{E}^{in} >$ where $\mathbf{V}^{in} = \mathbf{Layer}_1 \cup \mathbf{Layer}_2$ and $\mathbf{E}^{in} = \cup_{\mathbf{u} \in \mathbf{Layer}_2} \mathbf{O}ut(u)$

Output:

$\mathbf{G}^{out} =< \mathbf{V}^{out}, \mathbf{E}^{out} >$

1. Initialization:
   - $\mathbf{V}^{out} = \mathbf{V}^{in}$,$\mathbf{E}^{out} = \mathbf{E}^{in}$,
   - $\mathbf{S} = \mathbf{Layer}_1$,$\forall \mathbf{s} \in \mathbf{S}$ $\mathbf{in}(\mathbf{s}) = \mathbf{I}n(s)$,
   - $\mathbf{FWMerg} = \emptyset$, $\mathbf{BWMerg} = \emptyset$, $\mathbf{tmpS} = \emptyset$
2. While $\exists \mathbf{s} \in \mathbf{S}$ such that $\mathbf{in}(\mathbf{s}) \neq \emptyset$ do:
   (a) $\forall \mathbf{s} \in \mathbf{S}$ do:
      i. **Split(s,in(s))**
      ii. add all nodes created by Split(**s**) to **tmpS**
      iii. $\mathbf{in}(\mathbf{s}) = \emptyset$
   (b) **Forward-Merge(tmpS)** and add all newly created nodes to **FWMerg**
   (c) **Backward-Merge(FWMerg)** and add all newly created nodes to **BWMerg**
   (d) $\mathbf{V}^{out} = \mathbf{V}^{out} \cup \mathbf{BWMerg}$
   (e) **EdgeMin(BWMerg)**
   (f) $\forall \mathbf{s} \in \mathbf{BWMerg}$ and $\forall \mathbf{u} \in \mathbf{I}n(s), \forall \mathbf{v} \in \mathbf{O}ut(s)$ do:
      i. $\mathbf{E}^{out} = \mathbf{E}^{out} \backslash (\mathbf{u}, \mathbf{v})$
      ii. $\mathbf{E}^{out} = \mathbf{E}^{out} \cup \{(\mathbf{u}, \mathbf{s}), (\mathbf{s}, \mathbf{v})\}$
      iii. $\mathbf{in}(\mathbf{v}) = \mathbf{in}(\mathbf{v}) \cup \mathbf{s}$
3. $\forall \mathbf{v} \in \mathbf{V}^{out}$ such that $|\mathbf{O}ut(v)| == 1$ do:
   (a) for $\mathbf{s} = \mathbf{O}ut(v)$ and $\forall \mathbf{u} \in \mathbf{I}n(v)$
      i. $\mathbf{O}ut(u) = \mathbf{O}ut(u) \cup \mathbf{O}ut(v)$
      ii. $\mathbf{I}n(s) = \mathbf{I}n(s) \cup \mathbf{I}n(v)$
      iii. $\mathbf{E}^{out} = \mathbf{E}^{out} \cup (\mathbf{u}, \mathbf{s})$
   (b) $\mathbf{V}^{out} = \mathbf{V}^{out} \backslash \mathbf{v}$

Theorem 1 Given $\mathbf{G}^{in}$ maintaining all constraints (1-5), the output graph $\mathbf{G}^{out}$ also maintains all constraints (1-5) .

Theorem 2 For each subset $\Gamma_{\mathbf{C}} \subset \Gamma^{in}$ such that $\cap_{\mathbf{A}\in\Gamma_{\mathbf{C}}}\mathbf{P_A} \neq \emptyset$ there exists a corresponding node in $\mathbf{v} \in \mathbf{V}^{out}$ with $\mathbf{R_v} = \cap_{\mathbf{A}\in\Gamma_{\mathbf{C}}}\mathbf{P_A}$

Theorem 3 Alg 3 has runtime $O(\hat{\mathbf{D}}\mathbf{l})$.

See appendix for detailed proofs.

4.3 Statistical SIPO Model

Until now we assumed a scenario where a class (or category) is described deterministically by a set of properties $\mathbf{P_c}$ which should appear in all instances of the class. We would now like to relax this assumption and deal with the case where a group of class properties, appear only probabilistically in each of its instances. Thus, instead of requiring $\mathbf{x}\in\mathbf{P_c} \Rightarrow \psi_{\mathbf{x}}(\mathbf{s}) = 1\ \forall \mathbf{s} \in \mathbf{C}$, we denote $\delta_{\mathbf{x}}^{\mathbf{C}} \equiv \mathbf{P}(\psi_{\mathbf{x}}(\mathbf{s}) = 1|\ \mathbf{s} \in \mathbf{C})$, the class dependent probability of property $\mathbf{x}$. We say that property $\mathbf{x}$ belongs to class $\mathbf{C}$ if its class dependent probability exceeds a certain threshold: $\mathbf{x}\in\mathbf{P_c} \Rightarrow \delta_{\mathbf{x}}^{\mathbf{C}} > \rho_{\mathbf{x}}$, in which case we shall refer to the property $\mathbf{x}$ as "typical" to class $\mathbf{C}$. In addition properties can appear in classes where they are not typical, in which case we will refer to them as noisy properties. We define a property as noisy with respect to a given class if $\delta_{\mathbf{x}}^{\mathbf{C}} \leq \rho_{\mathbf{x}}$. $\rho_{\mathbf{x}}$ is a class independent probability of a specific property $\mathbf{x}$ to appear in instances of classes where this property is not "typical".

We say that $\mathbf{x}$ is "similarly typical" between classes if the class dependent values $\delta_{\mathbf{x}}^{\mathbf{C}}$ are similar. Formally we shall denote $\mathbf{X_i}$ a group of classes for which $\mathbf{x}$ is similarly typical- $\delta_{\mathbf{x}}^{\mathbf{C}} \in [\delta_{\mathbf{xi}} - \lambda_{\mathbf{x}}, \delta_{\mathbf{xi}} + \lambda_{\mathbf{x}}], \forall \mathbf{C} \in \mathbf{X_i}$.

For example, let $\Gamma = [\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}]$ denote a group of classes, and [0.1, 0.4, 0.45, 0.78, 0.8] the corresponding class dependent probabilities $[\delta_{\mathbf{x}}^{\mathbf{A}}, \delta_{\mathbf{x}}^{\mathbf{B}}, \delta_{\mathbf{x}}^{\mathbf{C}}, \delta_{\mathbf{x}}^{\mathbf{D}}, \delta_{\mathbf{x}}^{\mathbf{E}}]$ of property $\mathbf{x}$, where $\rho_{\mathbf{x}} = 0.2$ and $\lambda_{\mathbf{x}} = 0.1$. We will say that $\mathbf{x}$ is "typical" to classes $\mathbf{B}, \mathbf{C}, \mathbf{D}$ and $\mathbf{E}$ while $\mathbf{B}$ is "similarly typical" to $\mathbf{C}$ with respect to $\mathbf{x}$ and also $\mathbf{D}$ is "similarly typical" to $\mathbf{E}$ with respect to $\mathbf{x}$.

We now extend the notions of "typical" and "similarly typical" properties to a

"typical group" of co-occurring properties, where $\mathbf{x}$ now denotes a group of properties, $\mathbf{P_c}$ a group of groups of properties and $\delta_{\mathbf{x}}^{\mathbf{C}} \equiv \mathbf{P}(\bigwedge_{\mathbf{x}' \in \mathbf{x}} \psi_{\mathbf{x}'}(\mathbf{s}) = 1 | \mathbf{s} \in \mathbf{C})$.

In order to apply the graphical model and graph construction algorithm to this statistical scenario we restrict the probability model using the following assumptions:

1. Noisy properties of a class are statistically independent.

2. If a group of properties is "typical" to a class than each individual property is also "typical".

3. A group of properties is said to be "typical" to a class if $\delta_{\mathbf{x}}^{\mathbf{C}} \geq \prod_{\mathbf{x}' \in \mathbf{x}} \delta_{\mathbf{x}'}^{\mathbf{c}}$ where $\mathbf{x}$ denotes the group and $\mathbf{x}'$ an individual property in the group.

Assumption 2 is a strong restriction as we might want to allow two or more properties to be "typical" only when they occur together but not separately, in such a case we can define a new property as the conjunction of these property; the new property will be typical, but the old properties and their conjunction won't.

Statistical Intersection In the deterministic case, we presented Algorithm 3 for finding all possible general classes. This algorithm is based on finding commonalities among classes by computing the intersection between the property sets of each class. The notion of intersection between the sets $\mathbf{P_c}$ can be extended to the statistical scenario by regarding any "similarly typical" group of properties between two or more classes as a partial intersection between the group of classes. In the deterministic case the intersection of a given group of classes contains all properties shared by all classes in the group. In the statistical case it is more complicated: two groups of properties $\mathbf{x}$ and $\mathbf{y}$, which are both "similarly typical" for a group of classes Γ, may not be "similarly typical" together with respect to the same group of classes, hence $\mathbf{t} = \mathbf{x} \cup \mathbf{y}$ is not necessarily "similarly typical" in Γ. In such a case we will say that the intersection between a given group of classes is a set of sets of "similarly typical" properties, as opposed to the deterministic scenario where the intersection is a single group. Thus, we will say that the statistical intersection **SI** of Γ is $\{\mathbf{x}, \mathbf{y}\}$, denoted $\mathbf{SI}(\Gamma) = \{\mathbf{x}, \mathbf{y}\}$.

Given the set Υ of all "similarly typical" groups with respect to a group of classes Γ, we define the statistical intersection of Γ as a set of "similarly typical" groups of properties with the following two conditions:

1. $\forall \mathbf{x} \in \Upsilon \ \exists \mathbf{y} \in \mathbf{SI}(\Gamma)$ such that $\mathbf{x} \subseteq \mathbf{y}$.

2. $\forall \mathbf{y} \in \mathbf{SI}(\Gamma) \ \nexists \mathbf{x} \in \Upsilon$ such that $\mathbf{y} \subset \mathbf{x}$.

It can be said that **SI** is the maximal set of "similarly typical" groups with respect to any group of classes.

4.3.1 Statistical Algorithm

We shall now extend Algorithm 3 to deal with the statistical scenario and build a graph discovering all possible classes based on the statistical intersection while organizing them according to the partial order representation. In the statistical scenario we assume (as is common) that each known class is described using a set of i.i.d samples from the class, each having its own set of properties. We estimate $\delta_{\mathbf{x}}^{\mathbf{C}}$ from the empirical probability of a group of properties $\mathbf{x}$ appearing in class $\mathbf{C}$. For now we assume that for each single property $\mathbf{x}$ the following parameters are given- $\rho_{\mathbf{x}}$, $\delta_{\mathbf{xi}}$ and $\lambda_{\mathbf{x}}$. We note that $\lambda_{\mathbf{x}}$ depends on our confidence in the value of $\delta_{\mathbf{x}}^{\mathbf{C}}$ of all classes. As $\delta_{\mathbf{x}}^{\mathbf{C}}$ is the parameter of a Bernoulli random variable estimated by the average of an empirical set sampled i.i.d, our confidence in this value grows with the number of examples given for each class. This can guide the setting of $\lambda_{\mathbf{x}}$- the more samples we have the smaller we will set $\lambda_{\mathbf{x}}$.

In order to compute the statistical intersection we are interested in the "similarly typical" groups. Thus if two classes have property $\mathbf{x}$ as a "typical" property which is not "similarly typical" between the classes we would ignore this in the statistical intersections computations. This leads to the following preprocessing of the data, where we divide all single properties into the potential "similarly typical" groups, representing each such group as a node in $\mathbf{Layer}_1$. We note that a single property can belong to several "similarly typical" groups in case the similarity intervals of these groups overlap. Each such group is represented as a property of its own. When the "similarly typical" groups overlap, we regard each intersection including properties representing different similarity groups of the same original property as several

intersections, each containing only one possible similarity group of this property, For example, suppose property $\mathbf{y}$ has 3 potential similarity groups defined by the following intervals: [0.4,0.8],[0.5,0.9] and [0.6,1] and the values $\delta_{\mathbf{x}}^{\mathbf{A}}$, $\delta_{\mathbf{x}}^{\mathbf{B}}$ and $\delta_{\mathbf{x}}^{\mathbf{C}}$ are 0.6,0.6,0.8 respectively. We will say that $\mathbf{y}$ appears in the intersection of classes $\mathbf{A}$,$\mathbf{B}$ and $\mathbf{C}$ with respect to the third similarity group interval and in the intersection of $\mathbf{A}$ and $\mathbf{B}$ with respect to the first and second similarity group intervals. This will be represented in the graph by three possible nodes $\mathbf{y}_1$,$\mathbf{y}_2$ and $\mathbf{y}_3$, where $\mathbf{y}_2$ and $\mathbf{y}_1$ are descendants of a node representing classes $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{y}_3$ is a descendant of a node representing classes $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$.

Given the preprocessing creation of nodes representing the similarity groups of each property, the original Split operation in Algorithm 3 remains the same and the main modification is done in the Forward-Merge operation. Two points need to be considered when computing the statistical intersection:

1. It is not straight-forward that for $\mathbf{x}$ and $\mathbf{y}$ which are similarly typical the group $\mathbf{x} \cup \mathbf{y}$ is also similarly typical. This follows from the third assumption on the probability model.

2. The statistical intersection is represented by several nodes, where each single node represents a "similarly typical" group of properties.

We address this point by looking at the output of the Forward-Merge as defined above, and splitting each newly created node into a set of nodes representing the statistical intersection between the parents of the new node. We define the following node split operation:

Foward-Merge-Split(n): The output of the Forward-Merge (Algorithm 3) groups all properties shared between two classes, say $\mathbf{A}$ and $\mathbf{B}$, together into a single node $\mathbf{n}$. The properties are represented by the outgoing edges and the classes by the incoming edges. As noted above from the fact that $\mathbf{x}$ and $\mathbf{y}$ are similarly typical we cannot infer that the group $\mathbf{t} = \mathbf{x} \cup \mathbf{y}$ is also such. This can be verified by computing $\delta_{\mathbf{t}}^{\mathbf{A}}$ and $\delta_{\mathbf{t}}^{\mathbf{B}}$ and checking if $\mathbf{t}$ is "similarly typical" in accordance with assumption 3 on the probability distribution. $\delta_{\mathbf{t}}^{\mathbf{A}}$ can be computed by counting the number of samples from the class which have all properties in $\mathbf{t}$.

This is represented by splitting the single node **n** into a set of nodes, assigning a node to each maximal set of properties such that no properties can be added to it without violating the "similarly typical" relation. The single node representing the intersection between classes **A** and **B** is replaced by the set of nodes representing the statistical intersection. We propose the following iterative algorithm:

1. Init **SI**= All single properties descending from the outgoing edges.

2. for each $\mathbf{s} \in \mathbf{SI}$

- find all $\mathbf{x} \in \mathbf{SI}$ for which $\mathbf{t} = \mathbf{s} \cup \mathbf{x}$ a "similarly typical" group with respect to classes **A** and **B**.
- If there is at least one **t** "similarly typical" group
  - Remove **s** and **x** from **SI**
  - Insert all **t** into the end of **SI**

3. for each $\mathbf{s} \in \mathbf{SI}$

- For each property in **s**: if there exists another property representing the same original property but a different similarity group, split this node into two nodes, and keep track.

The complexity of checking if a group of properties is "similarly typical" is governed by the number of samples for each class. In case there are many samples this can be an expensive operation. On the other hand, when there are many samples the values of $\lambda_{\mathbf{x}}$ can be low as we are confident in our measure of $\delta_{\mathbf{t}}^{\mathbf{C}}$, with the underlining sparsity assumption of intersection among classes this means that only true (or confidant) intersections will be computed which should be relatively rare.

In step 3 of the Forward-Merge-Split we keep track of all nodes which were split due to a duplicate representation of the same property, which might occur in case the similarity intervals overlap. If these nodes were not merged separately during the Backward-Merge, we will unite them again.

4.4 Experiment

Recall the experiment described in 3.4, where we assumed we know the true hierarchy of tasks. We now repeat this experiment without assuming the hierarchy is known.

We test both the task hierarchy structure and the part membership hierarchy discovered by the statistical SIPO algorithm. We used the synthetic data and repeated the exact same experiment described in Section 3.4. For each task we used its positive samples as input to the SIPO algorithm. The output of SIPO represents a hierarchy of tasks, where for each internal node n we can define a unified task based on all tasks it represents (the tasks which are represented by the ancestor nodes of node n in $\mathbf{Layer}_2$).

In order to apply the same learning algorithm used in Section 3.4, a cascade of unified learning problems needs to be defined for each task. We define this cascade for each task based on the longest path in the SIPO graph, starting from the node representing the task to a node in $\mathbf{Layer}_1$. Each node in this cascade corresponds to a single unified task. We denote this approach by 'FullSIPOERMConj'.

While the statistical SIPO algorithm discovers the task hierarchy structure, it also discovers the part membership hierarchy. From this perspective, we can use SIPO directly as a feature selection algorithm for the conjunction classifier. Instead of using the discovered cascade to guide the cascade of learning steps defined in Section 3.4, we can simply construct the conjunction classifier feature set by selecting all features represented by the nodes in the SIPO graph corresponding to the cascade. We denote this approach by 'SIPOConj'.

Fig. 4.2 shows the results. First we note that clearly the statistical SIPO algorithm is able to discover the relevant hierarchical structure: 'FullCascadeConj' and 'FullSIPOERMConj' achieve the same results. Second we note that for the small sample scenario, inferring the relevant features directly from the SIPO structure outperforms significantly the risk minimization approach. All approaches outperform the baseline conjunction classifier 'VHConj'.

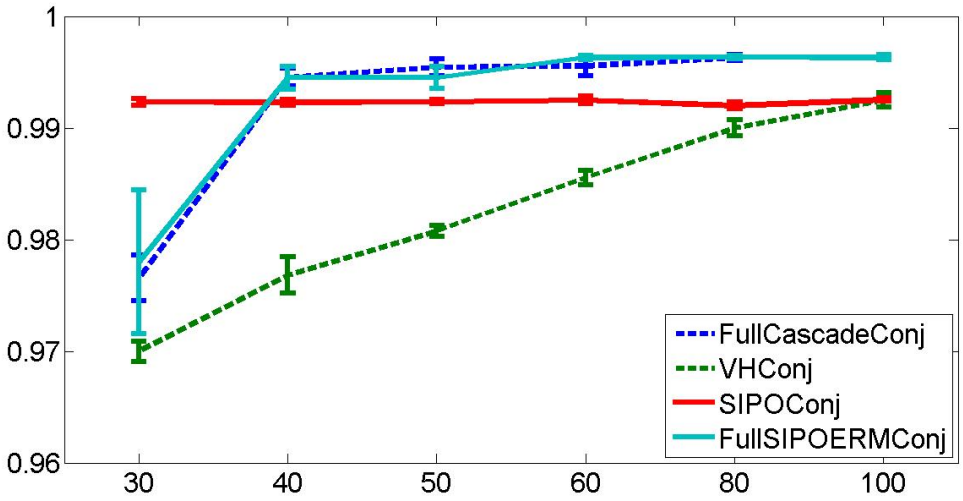


Figure 4.2: Accuracy results for the synthetic dataset experiment described in 3.4. The SIPO based approaches are compared to the baseline and the the original conjunction classifier described in 3.4, see Figure 3.3 for the results of the original experiment. 'Y-axis' corresponds to accuracy. 'X-axis' corresponds to sample size of the positive set (negative set has the same size). 'FullCascadeConj' denotes the original conjunction classifier described in 3.4; 'VHConj' denotes the two step approach based on Valiant's step and Hassler's step; 'FullSIPOERMConj' denotes training the original conjunction classifier using the cascade of training tasks discovered by SIPO; 'SIPOConj' denotes the conjunction classifier which is defined by selecting all features represented by the nodes in the SIPO graph corresponding to the discovered cascade.

5

Hierarchical Regularization Cascade

We present a hierarchical approach for information sharing among different classification tasks, in multi-task, multi-class and knowledge-transfer settings. We propose a top-down iterative method, which begins by posing an optimization problem with an incentive for large scale sharing among all classes. This incentive to share is gradually decreased, until there is no sharing and all tasks are considered separately. The method therefore exploits different levels of sharing within a given group of related tasks, without having to make hard decisions about the grouping of tasks.

In order to deal with large scale problems, with many tasks and many classes, we extend our batch approach to online setting and provide regret analysis of the algorithm. Based on the structure of shared information discovered in the joint learning settings, we propose two different knowledge-transfer methods for learning novel tasks. The methods are designed to work within the very challenging large scale settings. We tested our methods extensively on synthetic and real datasets, showing significant improvement over baseline and state-of-the-art methods.

More specifically, in the joint learning scenario, we propose a top-down iterative feature selection approach: It starts with a high level where sharing features among all tasks is induced. It then gradually decreases the incentive to share in successive levels, until there is no sharing at all and all tasks are considered separately in the last level. As a result, by decreasing the level of incentive to share, we achieve sharing between different subsets of tasks. The final classifier is a linear combination of diverse classifiers, where diversity is achieved by varying the regularization term.

The diversity of regularization we exploit is based on two commonly used regularization functions: the l_1 norm [78] which induces feature sparsity, and the l_1/l_2 norm analyzed in [41] which induces feature sparsity while favoring feature sharing between all tasks. Recently the sparse group lasso [79] algorithm has been introduced, a linear combination of the lasso and group-lasso [80] algorithms, which can yield sparse solutions in a selected group of variables, or in other words, it can discover smaller groups than the original group constraint (l_1/l_2).

We also developed two knowledge transfer approaches which are based on the shared information discovered by the regularization cascade in a joint learning scenario. In the knowledge transfer scenario we assume that some of the common information shared by the pre-trained tasks and the novel task can and has been found in the joint learning of the pre-trained tasks. When learning the novel class, the challenge is to discover which elements of shared information found in the joint learning of the pre-trained models is relevant, and possibly beneficial, to the training of the new task.

We propose two fundamentally different ways to accomplish knowledge transfer. In the first approach we use the pre-learnt structures to impose hierarchical dimensionality reduction on the data, which makes possible the learning from small sample of the new task. This approach is implemented in a batch algorithm. The second approach is an online scheme which maintains the original feature space and regularization structure used during the multi-task learning stage. In this second approach the pre-learnt structures are used to boot-strap the online learning of the new task.

Chapter Outline The main contribution of this chapter is to develop an implicit hierarchical regularization approach for information sharing, inducing shared information discovery in joint learning scenarios (see Section 5.1). Another important contribution is the extension to the online setting where we are able to consider a lot more learning tasks simultaneously, thus benefiting from the many different levels of sharing in the data (see Section 5.2 for algorithm description and regret analysis). In Section 5.3 we describe extensive experiments for the joint learning scenario on both synthetic and five popular real datasets. The results show that our algorithm performs better than baseline methods chosen for comparison, and state of the art methods described in [10, 12]. It scales well to large data sets, achieving significantly better results even when compared to the case where an explicit hierarchy is known in advance [9]. In Section 5.4 we present our batch and online knowledge-transfer approaches. In section 5.5 we describe the experiments for the knowledge-transfer scenario showing promising results both on synthetic and real datasets of medium and large size.

5.1 Hierarchical Regularization Cascade for Multi Task Learning

We now describe our algorithm, which learns while sharing examples between tasks. We focus only on classification tasks, though our approach can be easily generalized to regression tasks.

Notations Let $\mathbf{k}$ denote the number of tasks or classes. In the multi-task setting we assume that each task is binary, where $\mathbf{x} \in \mathcal{R}^{\mathbf{n}}$ is a datapoint and $y \in \{-1, 1\}$ its label. Each task comes with its own sample set $\mathbf{S}^i = \{(\mathbf{x}_s, \mathbf{y}_s)\}_{s=1}^{\mathbf{m}_i}$, where $\mathbf{m}_i$ is the sample size and $i \in \{1...\mathbf{k}\}$. In the multi-class setting we assume a single sample set $\mathbf{S} = \{(\mathbf{x}_s, \mathbf{y}_s)\}_{s=1}^{\mathbf{m}}$, where $\mathbf{y}_s \in \{1..k\}$. Henceforth, when we refer to $\mathbf{k}$ classes or tasks, we shall use the term tasks to refer to both without loss of generality.

Let $\mathbf{n}$ denote the number of features, matrix $\mathbf{W} \in \mathcal{R}^{\mathbf{n}\times\mathbf{k}}$ the matrix of feature weights being learnt jointly for the $\mathbf{k}$ tasks, and $\mathbf{w}^i$ the i'th column of $\mathbf{W}$. Let $\mathbf{b} \in \mathcal{R}^{\mathbf{k}}$ denote the vector of threshold parameters, where $\mathbf{b}^i$ is the threshold parameter corresponding to task i. $||\mathbf{W}||_1$ denotes the l_1 norm of $\mathbf{W}$ and $||\mathbf{W}||_{1,2}$ denotes its l_1/l_2 norm - $||\mathbf{W}||_{1,2} = \sum_{j=1}^{\mathbf{n}} ||\mathbf{w}_j||_2$, where $\mathbf{w}_j$ is the j'th row of matrix $\mathbf{W}$ and $||\mathbf{w}_j||_2$ its l_2 norm.

The classifiers we use for the i'th task are linear classifiers of the form $\mathbf{f}^i(\mathbf{x}) = \mathbf{w}^i * x + \mathbf{b}^i$. Binary task classification is obtained by taking the sign of $\mathbf{f}^i(\mathbf{x})$, while multi-class classification retrieves the class with maximal value of $\mathbf{f}^i(\mathbf{x})$.

To simplify the presentation we henceforth omit the explicit reference to the bias term $\mathbf{b}$; in this notation $\mathbf{b}$ is concatenated to matrix $\mathbf{W}$ as the last row, and each datapoint x has 1 added as its last element. Whenever the regularization of $\mathbf{W}$ is discussed, it is assumed that the last row of $\mathbf{W}$ is not affected. The classifiers now take the form $\mathbf{f}^i(\mathbf{x}) = \mathbf{w}^i * x$.

To measure loss we use the following multitask loss function:

$$\mathbf{L}(\{\mathbf{S}^i\}_{i=1}^{\mathbf{k}}, \mathbf{W}) = \sum_{i=1}^{k} \sum_{s\in\mathbf{S}^i} \max(0, 1 - \mathbf{y}_s * \mathbf{f}^i(\mathbf{x}_s)) \tag{5.1}$$

Without joint regularization this is just the sum of the hinge loss of $\mathbf{k}$ individual

tasks. The multi-class loss function is defined as in [81]:

$$\mathbf{L}(\mathbf{S}, \mathbf{W}) = \sum_{s \in \mathbf{S}} \max(0, 1 + \max_{\mathbf{y}_s = i, j \neq i} (\mathbf{f}^j(\mathbf{x}_s) - \mathbf{f}^i(\mathbf{x}_s))) \tag{5.2}$$

For brevity we will refer to both functions as $\mathbf{L}(\mathbf{W})$. Note that the choice of the hinge loss is not essential, and any other smooth convex loss function can be used (see [82]).

5.1.1 Hierarchical Regularization

We construct a hierarchy of regularization functions in order to generate a diverse set of classifiers that can be combined to achieve better classification. The construction of the regularization cascade is guided by the desire to achieve different levels of information sharing among tasks. Specifically, at the highest level in the hierarchy we encourage classifiers to share information among all tasks by using regularization based on the l_1/l_2 norm. At the bottom of the hierarchy we induce sparse regularization of the classifiers with no sharing by using the l_1 norm. Intermediate levels capture decreasing levels of sharing (going from top to bottom), by using for regularization a linear combination of the l_1 and l_1/l_2 norms. We denote the regularization term of level l by ψ^l:

$$\psi^l(\mathbf{W}) = \phi^l((1 - \lambda^l)||\mathbf{W}||_{1,2} + \lambda^l||\mathbf{W}||_1) \tag{5.3}$$

where λ^l is the mixing coefficient and ϕ^l is the regularization coefficient. The regularization coefficient of the last row of $\mathbf{W}^l$ corresponding to bias $\mathbf{b}$ is 0.

For each individual task (column of $\mathbf{W}^l$) we learn L classifiers, where each classifier is regularized differently. Choosing the L mixing terms $\lambda^l \in [0..1]$ diversely results in inducing L different levels of sharing, with maximal sharing at $\lambda^l = 0$ and no incentive to share at $\lambda^l = 1$.[1]

[1] Note that while each regularization term ψ^l induces the sparsity of $\mathbf{W}^l$, the output classifier $\sum_{l=1}^{L} \mathbf{W}^l$ may not be sparse.

5.1.2 Cascade Algorithm

Learning all diversely regularized classifiers jointly involves a large number of parameters which increases multiplicatively with the number of levels L. A large number of parameters could harm the generalization properties of any algorithm which attempts to solve the optimization problem directly. We therefore propose an iterative method presented in Algorithm 4, where each level is optimized separately using the optimal value from higher levels in the hierarchy.

Specifically, we denote by L the preset number of levels in our algorithm. In each level only a single set of parameters $\mathbf{W}^l$ is being learnt, with regularization uniquely defined by λ^l. We start by inducing maximal sharing with $\lambda^1 = 0$. As the algorithm proceeds λ^l monotonically increases, inducing decreased amount of sharing between tasks as compared to previous steps. In the last level we set $\lambda^L = 1$, to induce sparse regularization with no incentive to share.

Thus starting from $l = 1$ up to $l = L$, the algorithm for *sparse group learning cascade* solves

$$\mathbf{W}^l = \underset{\mathbf{W}}{\operatorname{argmin}}\ \mathbf{L}(\mathbf{W} + \mathbf{W}^{l-1}) + \psi^l(\mathbf{W}) \tag{5.4}$$

The learnt parameters are aggregated through the learning cascade, where each step l of the algorithm receives as input the learnt parameters up to that point- $\mathbf{W}^{l-1}$. Thus the combination of input parameters learnt earlier together with a decrease in incentive to share is intended to guide the learning to focus on more task/class specific information as compared to previous steps.

Note also that this sort of parameter passing between levels works only in conjunction with the regularization; without regularization, the solution of each step is not affected by the solution from previous steps. In our experiments we set $\lambda^l = \frac{l-1}{L-1}$ for all $l \in \{1..L\}$, while the set of parameters $\{\phi^l\}_{l=1}^L$ is chosen using cross-validation.

Algorithm 4 Regularization cascade

Input : L , $\{\lambda^l\}_{l=1}^L$, $\{\phi^l\}_{l=1}^L$
Output : $\mathbf{W}$

1. $\mathbf{W}^1 = \underset{\mathbf{W}}{\operatorname{argmin}}\ \mathbf{L}(\mathbf{W}) + \phi^1||\mathbf{W}||_{1,2}$
2. for $l = 2$ to L
   (a) $\mathbf{W} = \underset{\mathbf{W}}{\operatorname{argmin}}\ \mathbf{L}(\mathbf{W} + \mathbf{W}^{l-1}) + \phi^l((1-\lambda^l)||\mathbf{W}||_{1,2} + \lambda^l||\mathbf{W}||_1)$
   (b) $\mathbf{W}^l = \mathbf{W}^{l-1} + \mathbf{W}$
3. $\mathbf{W} = \mathbf{W}^L$

5.1.3 Batch Optimization

At each step of the cascade we have a single unconstrained convex optimization problem, where we minimize over a smooth convex loss function[1] summed with a non-smooth regularization term (5.4). This type of optimization problems has been studied extensively in the optimization community in recent years [82, 83]. We used two popular optimization methods [83, 84], which converge to the single global optimum with rate of convergence $O(\frac{1}{T^2})$ for [83]. Both are iterative procedures which solve at iteration t the following sub-problem:

$$\min_{\Theta^t}(\Theta^t - \Theta^{t-1})\nabla\mathbf{L}(\Theta^t) + \frac{\alpha^t}{2}||\Theta^t - \Theta^{t-1}||_2^2 + \phi\psi(W^t)$$

where $\psi(W^t) = ((1-\lambda)||\mathbf{W}^t||_{1,2} + \lambda||\mathbf{W}^t||_1)$ and α^t is a constant factor corresponding to the step size of iteration t. This sub-problem has a closed form solution presented in [85], which yields an efficient implementation for solving a single iteration of Algorithm 4. The complexity of the algorithm is L times the complexity of solving (5.4).

[1] In our experiments we used the hinge loss which is non-differentiable at 'x=1', at which point we used the sub-gradient '0'.

5.2 Online Algorithm

When the number of training examples is very large, it quickly becomes computationally prohibitive to solve (5.4), the main step of Algorithm 4. We therefore developed an online algorithm which solves this optimization problem by considering one example at a time - the set of parameters $\mathbf{W}^l$ is updated each time a new mini-sample appears containing a single example from each task.[1]

In order to solve (5.4) we adopt the efficient dual-averaging method proposed by [86], which is a first-order method for solving stochastic and online regularized learning problems. Specifically we build on the work of [87], who presented a closed form-solution for the case of sparse group lasso needed for our specific implementation of the dual-averaging approach. The update performed at each time step by the dual averaging method can be written as:

$$\mathbf{W}_t = \underset{\mathbf{W}}{\operatorname{argmin}}\ \bar{\mathbf{U}}\mathbf{W} + \psi^l(\mathbf{W}) + \frac{\gamma}{\sqrt{t}}h(\mathbf{W}) \tag{5.5}$$

where $\mathbf{U}$ denotes the subgradient of $\mathbf{L_t}(\mathbf{W}+\mathbf{W}^{l-1})$ with respect to $\mathbf{W}$, $\bar{\mathbf{U}}$ the average subgradient up to time $\mathbf{t}$, $h(\mathbf{W}) = \frac{1}{2}||\mathbf{W}||_2^2$ an auxiliary strongly convex function, and γ a constant which determines the convergence properties of the algorithm.

Algorithm 5 describes our online algorithm. It follows from the analysis in [86] that the run-time and memory complexity of the online algorithm based on update (5.5) is linear in the dimensionality of the parameter-set, which in our setting is nk. Note that in each time step a new example from each task is processed through the entire cascade before moving on to the next example.

5.2.1 Regret Analysis

In online algorithms, regret measures the difference between the accumulated loss over the sequence of examples produced by the online learning algorithm, as compared to the loss with a single set of parameters used for all examples and op-

[1] In an online to batch setting it suffices to solve (5.4) by iterating over all examples within each level before proceeding to the next level. This efficiently solves large scale problems, but it is not a truly online method which can deal with a setting where examples appear sequentially and are not known in advance.

Algorithm 5 Online regularization cascade

Input : L, γ, $\{\phi^l\}_{l=1}^L$, $\{\lambda^l\}_{l=1}^L$
Initialization: $\hat{\mathbf{W}}_0^l = 0$, $\bar{\mathbf{U}}_0^l = 0\ \forall l \in \{1..L\}$, $\mathbf{W}_t^0 = 0\ \forall t$

1. for t = 1,2,3,... do
   (a) for $l = 1$ to L
      i. Given $\mathbf{L_t}(\mathbf{W} + \mathbf{W}^{l-1})$, compute a subgradient $\mathbf{U}_t^l \in \partial \mathbf{L_t}(\mathbf{W} + \mathbf{W}^{l-1})$
      ii. $\bar{\mathbf{U}}_t^l = \frac{t-1}{t}\bar{\mathbf{U}}_{t-1}^l + \frac{1}{t}\mathbf{U}_t^l$
      iii. $\hat{\mathbf{W}}_t^l = \underset{\mathbf{W}}{\operatorname{argmin}}\ \bar{\mathbf{U}}_t^l \mathbf{W} + \psi^l(\mathbf{W}) + \frac{\gamma}{\sqrt{t}} h(\mathbf{W})$
      iv. $\mathbf{W}_t^l = \mathbf{W}_t^{l-1} + \hat{\mathbf{W}}_t^l$
   (b) $\mathbf{W}_t = \mathbf{W}_t^L$

timally chosen in hindsight. For T iterations of the algorithm, we can write the regret as $\mathbf{R}_T(\mathbf{W}_*) = \sum_{t=1}^T (\mathbf{L}_t(\mathbf{W}_t) + \psi(\mathbf{W}_t) - \mathbf{L}_t(\mathbf{W}_*) - \psi(\mathbf{W}_*))$, where $\mathbf{W}_* = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{t=1}^T (\mathbf{L}_t(\mathbf{W}) + \psi(\mathbf{W}))$.

At each time step $\mathbf{t}$, Algorithm 5 chooses for each level $l > 1$ of the cascade the set of parameters $\hat{\mathbf{W}}_t^l$, to be added to the set of parameters $\mathbf{W}_t^{l-1}$ calculated in the previous level of the cascade. Thus the loss in level l at time t $\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}) = \mathbf{L_t}(\hat{\mathbf{W}} + \mathbf{W}_t^{l-1})$ depends on both the example at time t and the estimate $\mathbf{W}_t^{l-1}$ obtained in previous learning stages of the cascade. We define the following regret function that compares the performance of the algorithm at level l to the best choice of parameters for all levels up to level l:

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) = \sum_{t=1}^T (\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_t^l) + \psi^l(\hat{\mathbf{W}}_t^l)) - \sum_{t=1}^T (\mathbf{L}_{\mathbf{t},\mathbf{W}_*^{l-1}}(\hat{\mathbf{W}}_*^l) + \psi^l(\hat{\mathbf{W}}_*^l)) \tag{5.6}$$

where $\hat{\mathbf{W}}_*^l = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{t=1}^T (\mathbf{L}_{t,\mathbf{W}_*^{l-1}}(\mathbf{W}) + \psi^l(\mathbf{W}))$, $\mathbf{W}_*^0 = 0$ and $\mathbf{W}_*^l = \sum_{k=1}^l \hat{\mathbf{W}}_*^k$.

We note that the key difference between the usual definition of regret above and the definition in (5.6) is that in the usual regret definition we consider the same loss function for the learnt and optimal set of parameters. In (5.6), on the other

hand, we consider two different loss functions - $\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}$ and $\mathbf{L}_{t,\mathbf{W}_*^{l-1}}$, each involving a different set of parameters derived from previous levels in the cascade.

We now state the main result, which provides an upper bound on the regret (5.6).

Theorem 1. *Suppose there exist* $\mathbf{G}$ *and* $\mathbf{D}$ *such that* $\forall \mathbf{t}, l \ ||\mathbf{U}_t^l|| < \mathbf{G}$ *and* $h(\mathbf{W}) < \mathbf{D}^2$*, and suppose that* $\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \geq -C\sqrt{T}$*; then*

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \leq A\sqrt{T} + B(T+1)^{\frac{3}{4}} \tag{5.7}$$

where $A = (\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})$*,* $B = \frac{4}{3}(l-1)\mathbf{G}\sqrt{\frac{2M}{\sigma}A}$*,* $C = -(M-1)(\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})$ *for some constant* $M > 1$*, and* σ *denotes the convexity parameter of* ψ*.*

Proof. We assume in the statement of the theorem that

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \geq -(M-1)(\gamma\mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})\sqrt{T} \tag{5.8}$$

for some constant $M > 1$. This assumption is justified if the accumulated cost obtained by the online algorithm is larger than the cost obtained by the optimal batch algorithm for most of the training examples. We argue that if this assumption is violated then the online algorithm is revealed to be a very good performer, and it therefore makes little sense to judge its performance by the deviation from another algorithm (the "optimal" batch algorithm) whose performance is worse for a significant fraction of the training sample.

Using the definition of $\mathbf{L}_{t,\mathbf{W}_*^{l-1}}$ and $\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}$, we rewrite the regret (5.6) with a single loss function

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) = \sum_{t=1}^{T}(\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_t^l) + \psi^l(\hat{\mathbf{W}}_t^l)) - [\sum_{t=1}^{T}(\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_*^l + \mathbf{W}_*^{l-1} - \mathbf{W}_t^{l-1}) + \psi^l(\hat{\mathbf{W}}_*^l))]$$

From the convexity of $\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}$ it follows that

$$\sum_{t=1}^{T}(\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_t^l) - \mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}(\hat{\mathbf{W}}_*^l + \mathbf{W}_*^{l-1} - \mathbf{W}_t^{l-1})) \leq \sum_{t=1}^{T} < U_t^l, \hat{\mathbf{W}}_t^l - \hat{\mathbf{W}}_*^l - \mathbf{W}_*^{l-1} + \mathbf{W}_t^{l-1} >$$

Under the conditions of the theorem, it is shown in (corollary 2a [86]) that

$$\sum_{t=1}^{T} < U_t, \mathbf{W}_t - \mathbf{W}_* > +\psi(\mathbf{W}_t) - \psi(\mathbf{W}_*)) \leq \Delta_T$$

where $\Delta_T = (\gamma \mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})\sqrt{T}$. Using this result and the sublinearity of the inner product, we get

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \leq \Delta_T + \sum_{t=1}^{T} < U_t^l, \mathbf{W}_t^{l-1} - \mathbf{W}_*^{l-1} > \tag{5.9}$$

From the definition of $\mathbf{W}_t^{l-1} = \sum_{k=1}^{l-1} \hat{\mathbf{W}}_t^k$ and $\mathbf{W}_*^{l-1} = \sum_{k=1}^{l-1} \hat{\mathbf{W}}_*^k$ and the Cauchy-Schwarz inequality we get

$$\begin{aligned}\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) &\leq \Delta_T + \sum_{k=1}^{l-1}\sum_{t=1}^{T} < U_t^l, \hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k > \\ &\leq \Delta_T + \sum_{k=1}^{l-1}\sum_{t=1}^{T} ||U_t^l||||\hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k|| \end{aligned} \tag{5.10}$$

We use the bound on $||\hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k||$ from (theorem 1b [86]) and assumption (5.8) to obtain

$$\begin{aligned}\sum_{t=1}^{T} ||\hat{\mathbf{W}}_t^k - \hat{\mathbf{W}}_*^k|| &\leq \sum_{t=1}^{T} \sqrt{\frac{2M\Delta_t}{\sigma t + \gamma\sqrt{t}}} \\ &\leq Q\sum_{t=1}^{T} t^{-\frac{1}{4}} \leq Q\frac{4}{3}(T+1)^{\frac{3}{4}}\end{aligned}$$

where $Q = \sqrt{\frac{2M}{\sigma}(\gamma \mathbf{D}^2 + \frac{\mathbf{G}^2}{\gamma})}$. Inserting this last inequality into (5.10), and since $\forall \mathbf{t}, l \ ||\mathbf{U}_t^l|| < \mathbf{G}$, we obtain

$$\mathbf{R}_T^l(\hat{\mathbf{W}}_*^l) \leq \Delta_T + (l-1)\mathbf{G}Q\frac{4}{3}(T+1)^{\frac{3}{4}} \tag{5.11}$$

from which (5.7) immediately follows. □

5.3 Joint Learning Experiments

Comparison Methods. We compare our algorithms to three baseline methods, representing three common optimization approaches: 'NoReg' - where learning is done simply by minimizing the loss function without regularization. 'L12' - a common approach to multi-task learning where in addition to minimizing the loss function we also regularize for group sparseness (enforcing feature sharing) using the l_1/l_2 norm. 'L1'- a very common regularization approach where the loss function is regularized in order to induce sparsity by using the l_1 norm. All methods are optimized using the same algorithms described above, where for the non-hierarchical methods we set $L = 1$, for 'NoReg' we set $\phi = 0$, for 'L12' we set $\lambda = 0$, and for 'L1' we set $\lambda = 1$. The parameter ϕ for 'L12' and 'L1' is also chosen using cross validation.

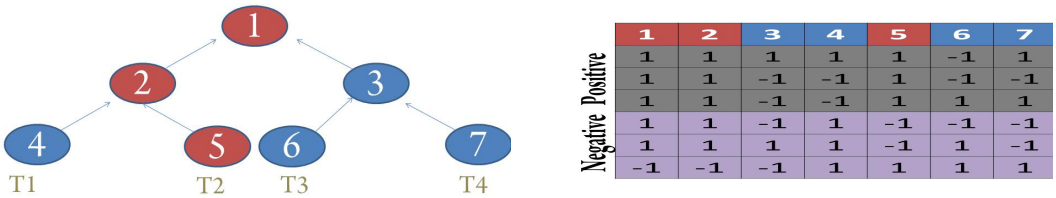
We also use for comparison three recent approaches which exploit relatedness at multiple levels. (i) The single stage approach of [10] which simultaneously finds the grouping of tasks and learns the tasks. (ii) The tree-guided algorithm of [12] which can be viewed as a two stage approach, where the tasks are learnt after a hierarchical grouping of tasks is either discovered or provided. We applied the tree-guided algorithm in three conditions: when the true hierarchy is known, denoted 'TGGL-Opt'; when it is discovered by agglomerative clustering (as suggested in [12]), denoted 'TGGL-Cluster'; or when randomly chosen (random permutation of the leafs of a binary tree), denoted 'TGGL-Rand'. (iii) The method described in [9] which is based on the work of [12] and extends it to a large scale setting where a hierarchy of classes is assumed to be known.

5.3.1 Synthetic Data

In order to understand when our proposed method is likely to achieve improved performance, we tested it in a controlled manner on a synthetic dataset we had created. We use the same data creation process as in the previous chapters 3 and 4. This synthetic dataset defines a group of tasks related in a hierarchical manner: the features correspond to nodes in a tree-like structure, and the number of tasks sharing each feature decreases with the distance of the feature node from the tree root. We tested to see if our approach is able to discover (implicitly) and exploit the hidden structure thus defined. The inputs to the algorithm are the sets of examples and labels from all k tasks $\{\mathbf{S}^i\}_{i=1}^k$, without any knowledge of the underlying structure.

More specifically, the data is created in the following manner: we define a binary tree with k leaves. Each leaf in the tree represents a single binary classification task. Each node in the tree corresponds to a single binary feature $f \in \{-1, 1\}$. For a single task we divide the features into two groups: task-specific and task-irrelevant. Task-irrelevant features have equal probability of having the value 1 or -1 for all examples in the task. Task-specific features are assigned the value 1 for all positive examples of the task. All negative examples of the task must have at least one feature from the task-specific feature set with a value of -1.

The task-specific feature set is the set of features corresponding to all nodes on the path from the leaf to the root, while all other nodes define the task-irrelevant feature set. For each group of tasks, their shared features are those corresponding to common ancestors of the corresponding leaves. Illustration is given in Fig. 5.1.



1	2	3	4	5	6	7
1	1	1	1	1	-1	1
1	1	-1	-1	1	-1	-1
1	1	-1	-1	1	1	1
1	1	-1	1	-1	-1	-1
1	1	1	1	-1	1	-1
-1	-1	-1	1	1	1	1

Figure 5.1: Synthetic data illustration. Left graph shows a tree of features corresponding to four tasks. The task specific features of task 2, 'T2', are highlighted in red. The task-irrelevant features are marked blue. The table on the right shows a sample example for task 'T2' given the task tree to the left. Each row denotes a sample. Each column denotes a feature. 'Positive' and 'Negative' denote positive and negative examples.

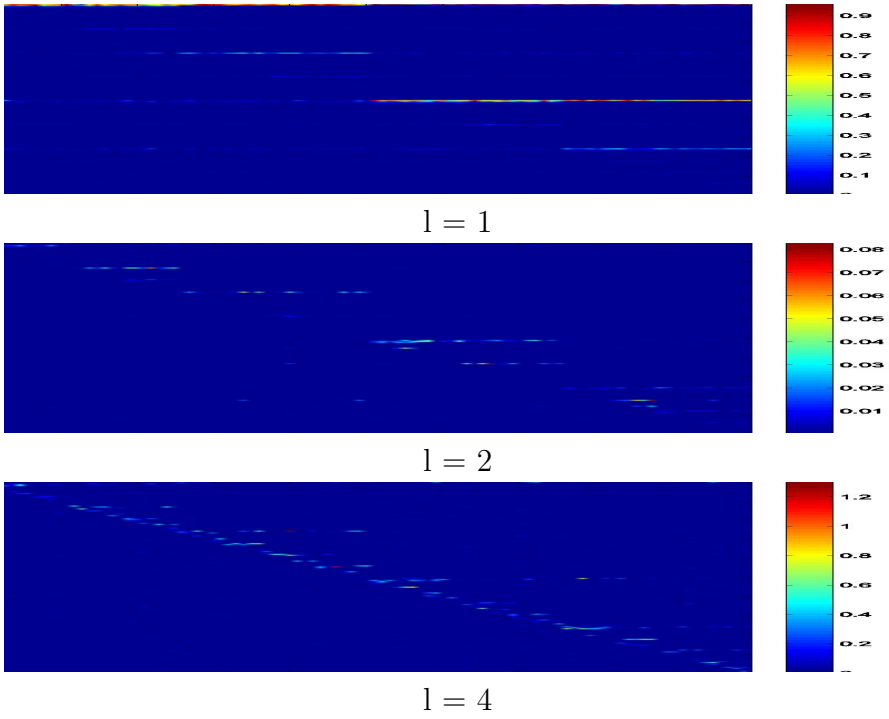


Figure 5.2: Synthetic experiment learnt parameters $\mathbf{W}^l$. Each plot corresponds to a single matrix learnt at stage l of the cascade. The rows correspond to features and each column corresponds to a single task.

Structure discovery: Feature weights learnt at different learning steps of the cascade for an experiment with 100 synthetic tasks, 199 features and 20 positive and negative examples per task, are shown in Fig 5.2. As can be seen, the first learning stages, $l = 1$ and $l = 2$ capture shared information, while the last stage $l = 4$ captures task specific features. The hierarchical shared structure of features is discovered in the sense that higher levels in the cascade share the same set of features, and as the cascade progresses the chosen features are shared by fewer tasks. The pattern of learnt feature weights fits the engineered pattern of feature generation.

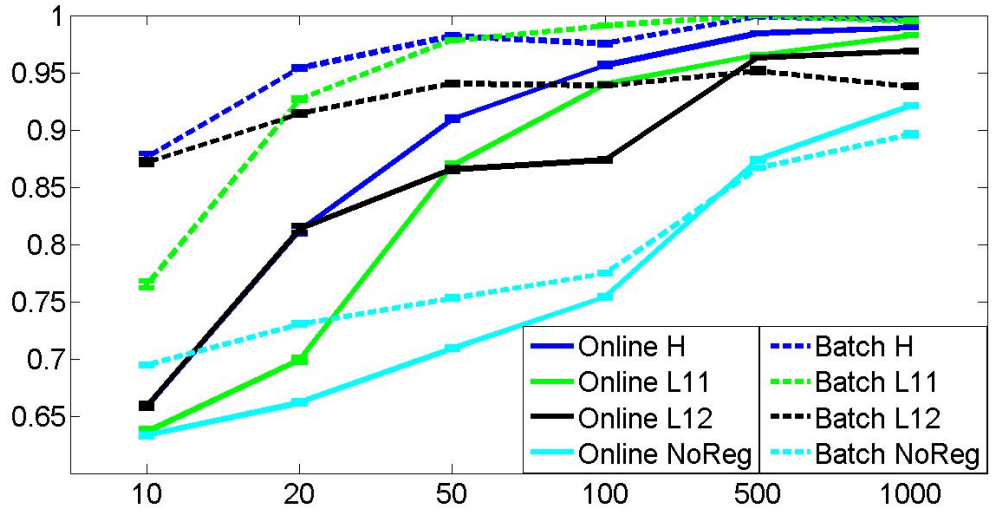
Classification performance: We start by showing in Fig. 5.3a that our hierarchical algorithm achieves the highest accuracy results, in both batch and online settings. For the smallest sample size the 'L12' baseline achieves similar performance, while for the largest sample size the 'L1' baseline closes the gap indicating that given enough examples, sharing of information between classes becomes less important.

We also see that the online Algorithm 5 converges to the performance of the batch algorithm after seeing enough examples.

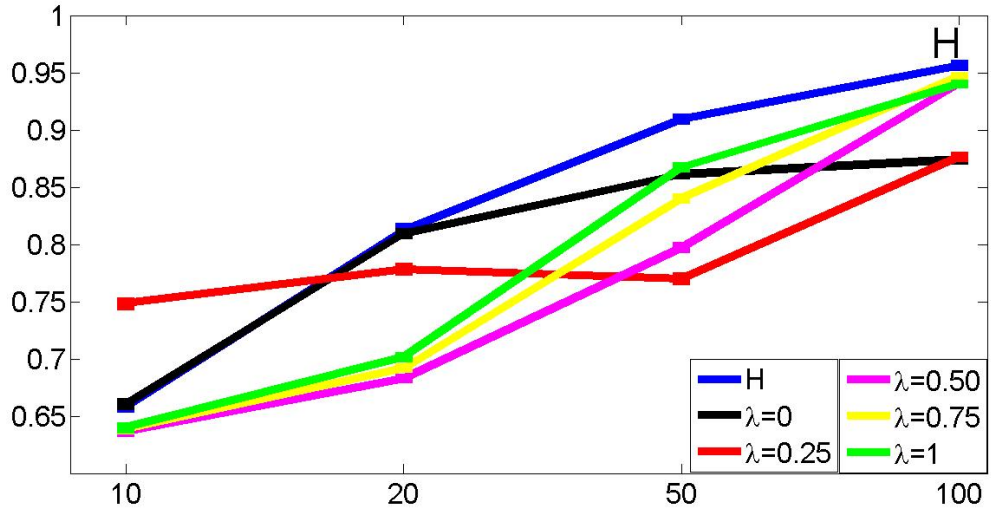
With a small sample size, it is common to present the data to an online algorithm multiple times; in our case this achieves similar qualitative performance to Fig. 5.3a (results are omitted). In Fig. 5.3b we show a comparison of the cascade to a group of single level regularization schemes, using the same set of λ values we used in the cascade. Clearly no single-level regularization achieves as good performance as the hierarchical method.

The advantage of the hierarchical method is not due simply to the fact that it employs a combination of classifiers, but rather that it clearly benefits from sharing information. Specifically, when the cascade was applied to each task separately, it achieved only 93.21% accuracy as compared to 95.4% accuracy when applied to all the 100 tasks jointly.

To evaluate our implicit approach we compared it to the Tree-Guided Group Lasso [12] (TGGL) where the hierarchy is assumed to be known - either provided by a supervisor (the *true* hierarchy), clustered at pre-processing, or randomly chosen. Fig. 5.4 shows results for 100 tasks and 20 positive examples each. We challenged the discovery of task relatedness structure by adding to the original feature representation a varying number (500-4000) of irrelevant features, where each irrelevant feature takes the value '-1' or '1' randomly. Our method performs much better than TGGL with random hierarchy or clustering-based hierarchy. Interestingly, this advantage is maintained even when TGGL gets to use the true hierarchy, with up to 1500 irrelevant features, possibly due to other beneficial features of the cascade.



(a)



(b)

Figure 5.3: Synthetic data results, where accuracy results correspond to 10 repetitions of the experiment. Above the 'Y'-axis measures the average accuracy over all tasks, and the 'X'-axis the sample size. 'H' denotes our hierarchical Algorithm with 5 levels, 'L11' ,'L12' and 'NoReg' - the different baseline methods. (a) Results for both the batch and online algorithms with a single presentation of the data. (b) A comparison of our cascade approach 'H' to variants corresponding to intermediate levels in the cascade, defined by the value λ.

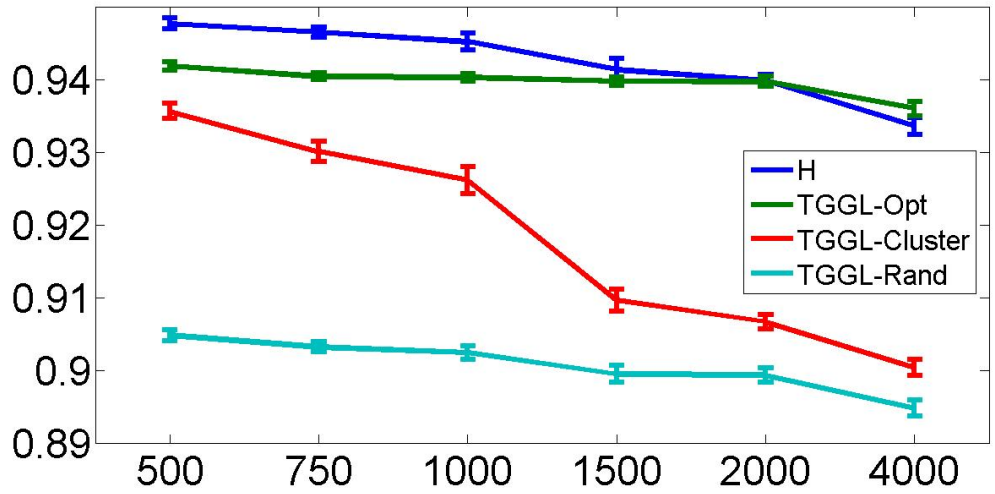


Figure 5.4: Synthetic data results. The 'Y'-axis measures the average accuracy over all tasks, where accuracy results correspond to 10 repetitions of the experiment. Performance as a function of the number of random features ('X'-axis). 'H' denotes our hierarchical Algorithm with 5 levels. We show comparison to the tree-guided group lasso algorithm based on the true hierarchy 'TGGL-Opt', clustered hierarchy 'TGGL-Cluster' and random hierarchy 'TGGL-Rand'.

Parameter Robustness Fig. 5.5 shows the robustness of the hierarchical approach with respect to the regularization parameter ϕ. For all three regularizing approaches we varied the value of ϕ in the range [0.01-2] with 0.01 jumps. For the hierarchal approach we set $\phi^1 = \phi$ and $\phi^l = \frac{\phi^{l-1}}{2}$ for all $l \in [2..L]$.

We also found the method to be quite robust to the parameter L determining the number of levels in the cascade. Varying the value of L between 3 to 7 on the synthetic data with 100 tasks gave close results in the range 94.5% to 95.5%.

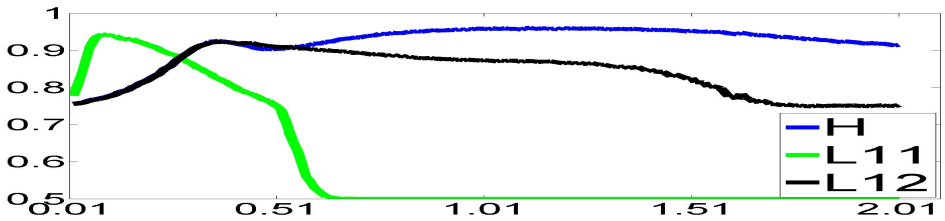


Figure 5.5: Performance as a function of the regularization parameter ϕ. Synthetic data with 100 examples per task. The 'Y'-axis corresponds the average accuracy of all tasks on 10 repetitions of the experiment. The 'X'-axis corresponds to the value of ϕ. Note that the max values of each method are: 96.02, 94.23 and 92.30 for 'H', 'L11' and 'L12' respectively.

Table 5.1: Varying the number of levels L

L = 2	L = 3	L = 4	L = 5	L = 6	L = 7	L = 8
92.42	95.47	95.73	94.74	95.21	94.48	93.52

Adding Tasks We examined the effect of the number of tasks in the multitask setting. Ideally adding more tasks should never reduce performance, while in most cases leading to improved performance. We tested two scenarios - adding tasks which are similarly related to the existing group of tasks Fig. 5.6a, and adding tasks which are loosely related to all other tasks but strongly related among themselves Fig. 5.6b.

With additional tasks of the same degree of relatedness, we increase the amount of information available for sharing. As expected, we see in Fig. 5.6a that performance improves with increasing number of tasks, both for the hierarchical algorithm

and for 'L12Reg'. 'L1Reg' is not intended for information sharing, and therefore it is not affected by increasing the number of tasks. When adding loosely related tasks, we see in Fig. 5.6b that the performance of the hierarchical algorithm increases as we add more tasks; for the 'L12Reg' method, on the other hand, we see a significant drop in performance. This is because in this case the overall relatedness among all tasks decreases as additional tasks are added; the 'L12Reg' method still tries to share information among all tasks, and its performance therefore decreases.

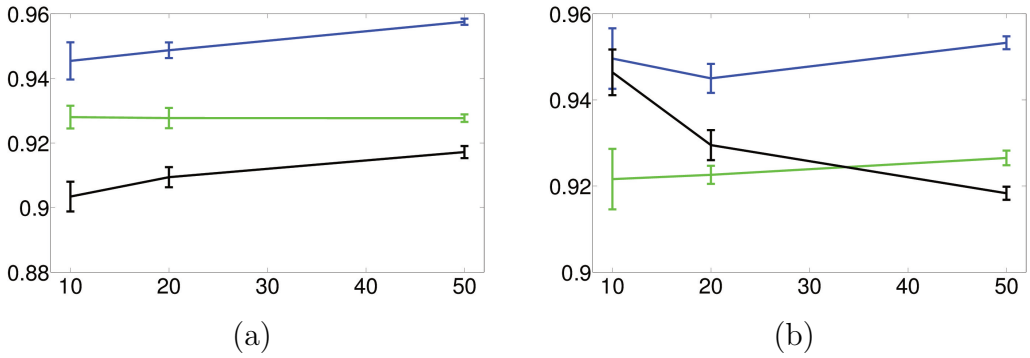


Figure 5.6: Adding tasks results. Plots correspond to the average 1-vs-rest accuracy as a function the number of tasks, when (a) adding similarly related tasks and (b) adding loosely related tasks. Blue denotes our hierarchical Algorithm 4, green the 'L1Reg' baseline and black the 'L12Reg' baseline method.

Data Rotation Next, we wish to isolate the two factors of sparsity and shared information. The synthetic data was constructed so that there is an increased level of shared information between classes as a function of the distance between their respective leaves in the defining tree of features. The shared features are also sparse. In order to maintain the shared information and eliminate sparseness, we rotate the vector of features; when the rotation is applied to more features, the amount of sparseness decreases respectively.

Table 5.2 shows the comparative results both for the case when all features are rotated and when only half are rotated (in which case the features being rotated are chosen randomly). As expected, the regularization-free method - 'NoReg' - is not effected by any of this. The performance of 'L1Reg', which assumes sparsity, drops as expected, reaching baseline with full rotation, presumably because during

Table 5.2: Performance comparison for the different methods applied to the Synthetic data. 'T 100 S 20' denotes the multi-task setting with 100 tasks and 20 samples each, 'half rotated' - the same setting as 'T 100 S 20' with a random rotation of half of the features, and 'full rotation' - a random rotation of all the features.

	T 100 S 20	half rotation	full rotation
H	95.40 ± 0.17	90.37 ± 0.61	78.49 ± 0.16
L1Reg	92.54 ± 0.17	86.70 ± 0.59	73.01 ± 0.09
L12Reg	91.49 ± 0.2	85.56 ± 0.62	78.49 ± 0.16
NoReg	72.88 ± 0.19	72.81 ± 0.12	73.03 ± 0.10

cross-validation a very low value for the regularization parameter is chosen. The two methods which exploit shared information, our hierarchical algorithm and the 'L12Reg' baseline method, perform better than baseline even with no sparseness (full rotation), showing the advantage of being able to share information.

5.3.2 Real Data

Small Scale [10] describe a multi-task experimental setting using two digit recognition datasets MNIST [88] and USPS [89], which are small datasets with only 10 classes/digits. For comparison with [10], we ran our method on these datasets using the same representations, and fixing $L = 3$ for both datasets. Table 5.3 shows all results, demonstrating clear advantage to our method. The results of our basic baseline methods 'NoReg' and 'L1' achieve similar or worse results,[1] comparable to the single task baseline approach presented in [10]. Thus, the advantage of the cascade 'H' does not stem from the different optimization procedures, but rather reflects the different approach to sharing.

Medium Scale We tested our batch approach on four medium sized data sets: Cifar100 [90], Caltech101, Caltech256 [91] and MIT-Indoor Scene dataset [92] with 100, 102, 257 and 67 categories in each dataset respectively. We tested both the

[1] 'NoReg' - 9.5% ± 0.2 and 17% ± 0.5 for USPS and MNIST respectively; 'L1' - 8.8% ± 0.5 and 16% ± 0.8 for USPS and MNIST respectively.

Table 5.3: Error rates on digit datasets

	USPS	MNIST
H	6.8% ± 0.2	13.4% ± 0.5
Kang et al.	8.4% ± 0.3	15.2% ± 0.3

multi-class and multi-task settings[1]. For the multi-task setting we consider the 1-vs-All tasks. For Cifar-100 we fixed $L = 5$ for the number of levels in the cascade, and for the larger datasets of Caltech101/256 and Indoor-Scene we used $L = 4$.

We also investigated a variety of features: for the Cifar-100 we used the global Gist [93] representation embedded in an approximation of the RBF feature space using random projections as suggested by [94], resulting in a 768 feature vector. For the Caltech101/256 we used the output of the first stage of Gehler et al's kernel combination approach [95] (which achieves state of the art on Caltech101/256) as the set of features. For the MIT-Indoor Scene dataset we used *Object Bank* features [96], which achieves state-of-the-art results on this and other datasets.

We tested the Cifar-100 dataset in the multi-task and multi-class settings. We used 410 images from each class as the training set, 30 images as a validation set, and 60 images as the test set. For the multi-task setting we considered the 100 1-vs-rest classification tasks. For each binary task, we used all images in the train set of each class as the positive set, and 5 examples from each class in the 'rest' set of classes as the negative set. The experiment was repeated 10 times using different random splits of the data. Results are shown in Table 5.4, showing similar performance for the cascade and the competing Tree-Guided Group Lasso method.

In our experiments with the MIT-Indoor Scene dataset we used 20, 50 and 80 images per scene category as a training set, and 80, 50 and 20 images per category as test set respectively. We repeated the experiment 10 times for random splits of the data, including the single predefined data split provided by [92] as a banchmark. Loss was measured using the multi-class hinge loss (5.2). Fig. 5.7-left shows the classification results of the cascade approach, which significantly outperformed the

[1] In the binary tasks we measured accuracy by the standard measure $\frac{\#True-Positive}{2*\#Positive} + \frac{\#True-Negative}{2*\#Negative}$, and report the average accuracy for several tasks. In the multi-class setting we report the average recall.

Table 5.4: Cifar-100: accuracy results. 'Baselines' denotes - 'L1', 'L12' and 'NoReg' which showed similar performance.

	multi-class	1-vs-rest
H	21.93 ± 0.38	79.91 ± 0.22
Baselines	18.23 ± 0.28	76.98 ± 0.17
TGGL-Cluster	-	79.97 ± 0.10
TGGL-Rand	-	80.25 ± 0.10

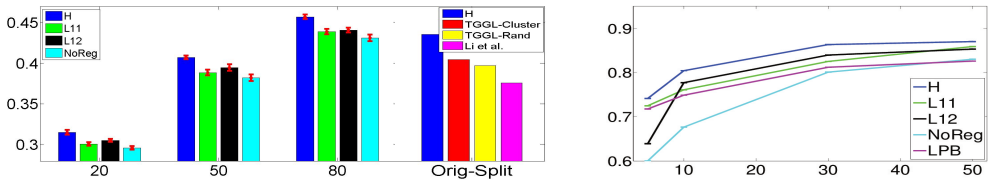


Figure 5.7: Real data results. 'Y'-axis measures the average accuracy over all tasks. Left, Multiclass accuracy results on the MIT-Indoor-Scene dataset, for 4 experimental conditions: 20, 50, and 80 images used for training respectively, and 'OrigSplit' - the single predefined split of [92]. Right, Multi-task 1-vs-rest results for the Caltehc256 dataset, where 'LPB' denotes our implementation of the binary version of the approach presented in [95] (see text). The 'X'-axis varies with sample size.

baseline methods and the previously reported state of the art result of 37.6% [96], achieving 45.9% using the exact same feature representation. We also significantly outperformed 'TGGL-Cluster' and 'TGGL-Rand'.

With Caltech101 and Caltech256 we used the data provided by [95] for comparisons in both their original multi-class scenario and a new multi-task scenario. We tested our approach using the exact same experimental setting of [95] given the scripts and data provided by the authors. In the original multi-class setting addressed in [95] our results compare to their state-of-the-art results both for 30 training images (78.1%) in the Caltech101 and for 50 images (50%) in the Caltech256.

In the multi-task scenario we trained a single 1-vs-rest classifier for each class. In addition to our regular baseline comparisons we implemented a variant of the ν-LPB method, which was used in [95] as the basis to their multi-class approach.

Fig. 5.7-right shows results for the multi-task setting on the Caltech256. First

we note that our algorithm outperforms all other methods including ν-LPB. We also note that given this dataset all regularization approaches exceed the NoReg baseline, indicating that this data is sparse and benefits from information sharing. (Results for the Caltech101 are similar and have therefore been omitted.)

Large Scale We demonstrate the ability of our online method to scale up to large datasets with many labels and many examples per each label by testing it on the ILSVRC(2010) challenge [97]. This is a large scale visual object recognition challenge, with 1000 categories and 668-3047 examples per category. With so many categories the usual l_1/l_2 regularization is expected to be too crude, identifying only a few shared features among such a big group of diverse classes. On the other hand, we expect our hierarchical method to capture varying levels of useful information to share.

Since the representation is not under investigation here, we only compare our method to other methods which used a similar representation, preferring when possible to use representation scripts provided by the authors. With the ILSVRC(2010) challenge we compared our method to [9], which is also a hierarchical scheme based on the tree guided group lasso approach of [12]. We compared our method to the hierarchical scheme of [9] (using their exact same feature representation). Rather than compute the hierarchy from the data, their method takes advantage of a known semantic word-net hierarchy, which is used to define a hierarchical group-lasso regularization and calculate a similarity matrix augmented into the loss function. The comparison was done on the single split of the data provided by the challenge [97] .

Table 5.5: ILSVRC(2010): Classification accuracy of the best of N decisions, Top N.

	Top 1	Top 2	Top 3	Top 4	Top 5
Alg 5	0.285	0.361	0.403	0.434	0.456
Zhao et al	0.221	0.302	0.366	0.411	0.435

Table 5.5 shows our performance as compared to that of [9]. We show accuracy rates when considering the 1-5 top classified labels. In all settings we achieve significantly better performance using the exact same image representation and much less labeled information.

In Fig 5.8 we show the error rates for the Top-1 scenario, considering a single classification. We show results when training using all the data Fig 5.8-left, and when using only 100 examples per each task Fig 5.8-right. The results are shown as a function of the number of repetitions of the algorithm over all the data.

At convergence we see an improvement of 1.67% in accuracy when using the cascade with 7 levels, 28.96% compared to 27.29%. [9] obtained an improvement of 1.8% in accuracy when comparing their approach to their own baseline, 22.1% vs. 20.3%. We obtained a similar rate of improvement using much less information (not knowing the hierarchy) for a higher range of accuracies.

We note that our baseline approaches converge after 20 repetitions when using all the data, (for clarity we show only up to 15 repetitions in the left plot of Fig 5.8). This effectively means the same runtime, as the cascade runtime is linear in the number of levels where each level has the same complexity of the baseline approaches. On the other hand the online cascade algorithm 5 can be trivially parallelized where as the repetitions over the data for a single baseline cannot. Thus, in a parallel setting the gain in runtime would be linear in the number of levels of the cascade. A trivial parallelization can be implemented by running each level of the online cascade on a time stamp shifted by l thus the first level of the cascade will see at time t the t sample while level l will see sample $t - l$.

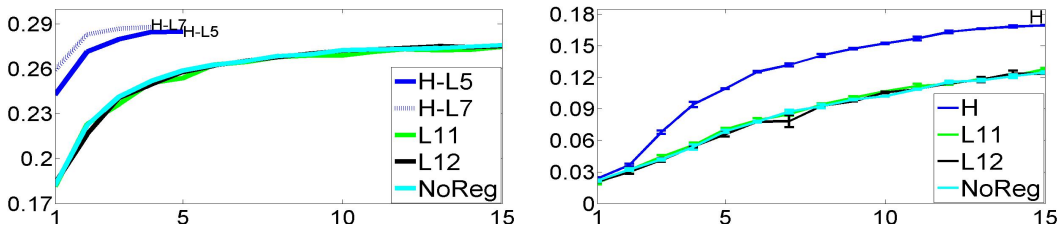


Figure 5.8: Real data, showing performance of Top-1 classification on the ILSVRC(2010) challenge [97] using all examples (left plot) or only 100 examples per each category (right plot). Here the 'X'-axis corresponds to repetitions over the training data. In the left plot 'H-L5' and 'H-L7' denote our hierarchical algorithm with 5 and 7 levels respectively. In the right plot 'H' corresponds to 5 levels in our hierarchical algorithm. The error bars in correspond to the standard error given 3 different choices of 100 Examples.

Discussion For small and medium scale datasets we see that our cascade approach and the batch Algorithm 4 outperform the baseline methods significantly. For the large scale dataset the online Algorithm 5 significantly outperforms all other baseline methods. It is interesting to note that even when the alternative baseline methods perform poorly, implying that the regularization functions are not beneficial on their own, combining them as we do in our hierarchical approach improves performance. This can be explained by the fact that a linear combination of classifiers is known to improve performance if the classifiers are accurate and diverse.

When comparing to the recent related work of [12] denoted TGGL, we see that our implicit approach performs significantly better with the synthetic and MIT-Indoor scene datasets, while on the Cifar dataset we obtain similar results . With the synthetic dataset we saw that as the clustering of the task hierarchy becomes more challenging, TGGL with clustering degrades quickly while the performance of our method degrades more gracefully. We expect this to be the case in many real world problems where the underlining hierarchy is not known in advance. Furthermore, we note that with the small scale digit datasets our approach outperformed significantly the reported results of [10]. Finally, we note that our approach can be used in a pure online setting while these two alternative methods cannot.

We also compared with a third method [9] using the ILSVRC(2010) challenge [97]; this is a challenging large scale visual categorization task, whose size - both the number of categories and the number of examples per category, provides the challenges particularly suitable for our approach. The online algorithm makes it possible to scale up to such a big dataset, while the hierarchical sharing is important with possibly many relevant levels of sharing between the tasks. Particularly encouraging is the improvement in performance when compared to the aforementioned work where an explicit hierarchy was provided to the algorithm.

5.4 Knowledge Transfer

In this section we discuss methods to transfer information to novel classes, where information transfer is based on the cascade of matrices $\{\mathbf{W}^l\}_{l=1}^L$ built during the training of pre-trained models. The method is illustrated in Fig. 5.9.

We describe two distinct knowledge-transfer methods. The first is a batch al-

gorithm described in Section 5.4.1, which is based on dimensionality reduction of pre-trained models. This method is particularly useful when the data lies in a high dimensional feature space, and the number of examples from the novel task is too small to learn effectively in such a space. The second is an online method described in Section 5.4.2, which maintains the same regularization structure and parameters of the online multi-task setting. When the dataset is large and an online method is called for, this approach is particularly useful for bootstrapping the online learning of novel classes, achieving higher performance at the early stages of the learning as compared to a non transfer approach.

5.4.1 Batch Method

In order to overcome the problem of small sample, we use the learnt models $\{\mathbf{W}^l\}_{l=1}^{L}$ to perform dimensionality reduction via projection; now the new task is represented in L sub-spaces that capture the structure of the shared information between the previous k tasks (see illustration in Fig. 5.9).

The method is described below in Algorithm 6. At each level l we project the new data-points onto the subspace spanned by the top z left singular-vectors of $\mathbf{W}^l$. The projection matrix $\mathbf{P}^l$ is defined by the first z columns of the orthonormal matrix $\mathbf{U}^l$, where $svd(\mathbf{W}^l) = \mathbf{U}^l\Sigma\mathbf{V}^l$. Thus the modeling of the new task involves $z * L$ parameters, which are learned in a cascade where each learning problem deals with only z parameters. Note that in the l'th level we project the data onto the unique subspace characteristic of level l. In order to pass the parameters to the next level we project back to the original feature space, which is accomplished by $(\mathbf{P}^l)^t$ in Step 2d.

5.4.2 Online Method

We assume that a related set of tasks has been learnt using the online Algorithm 5, capturing various levels of shared information. Then, in order to initialize the online learning of a new single task or group of tasks, we use their learnt matrix of parameters together with the regularization structure underlying the learning cascade.

The online knowledge-transfer algorithm is described below in Algorithm 7; it succeeds $\mathbf{t}_{old}$ iterations of Algorithm 5. The input to the algorithm is the same set

Algorithm 6 Knowledge-Transfer with shared features projections

Input :

L number of levels

$\{\mathbf{P}^l\}_{l=1}^L$ Set of projections matrices learnt from the k pre-trained tasks

Output :

$\mathbf{W}$

1. $\mathbf{W}^0 = 0$
2. for $l = 1$ to L
   (a) Projection:
      i. $\hat{\mathbf{x}} = \mathbf{P}^{l^t} * \mathbf{x}$, $\forall i \in [1..\mathbf{k}]$ and $\forall \mathbf{x} \in \mathbf{S}^i$
      ii. $\hat{\mathbf{w}}^{l-1} = \mathbf{P}^{l^t} * \mathbf{w}^{l-1}$
   (b) $\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}}\ \mathbf{L}(\{\hat{\mathbf{S}}^i\}_{i=1}^{\mathbf{k}}, \mathbf{W} + \hat{\mathbf{W}}^{l-1})$
   (c) Backprojection: $\mathbf{w} = \mathbf{P}^l * \hat{\mathbf{w}}$
   (d) $\mathbf{w}^l = \mathbf{w}^{l-1} + \mathbf{w}$
3. $\mathbf{w} = \mathbf{w}^L$

of parameters used for Algorithm 5 and its intermediate calculations - the cascade $\{\mathbf{W}_{old}^l\}_{l=1}^L$ and the set of final average subgradients $\{\bar{\mathbf{U}}_{old}^l\}_{l=1}^L$. These are used to approximate future subgradients of the already learnt tasks, since Algorithm 7 receives no additional data-points for these tasks. The parameters of Algorithm 5 are used because cross-validation for parameter estimation is not possible with small sample.

Below we denote the vector corresponding to the mean value of each feature as $mean(\mathbf{W}_{old}^l)$. We denote the concatenation of columns by $\circ$. In order to account for the difference between the old time step $\mathbf{t}_{old}$ to the new time step $\mathbf{t}$ we consider $h(\mathbf{W})$ to be the squared l_2 norm applied to each column separately. We calculate the inner product of the resulting vector with $\frac{\gamma}{\sqrt{\mathbf{t}_{old} \circ \mathbf{t}}}$ in step 1(a).(iv); $\sqrt{\mathbf{t}_{old} \circ \mathbf{t}}$ denotes a vector derived by the concatenation of $\mathbf{k}$ times $\mathbf{t}_{old}$ with $\mathbf{t}$.

Algorithm 7 Online Knowledge-Transfer learning cascade

Input :

L, $\{\lambda^l\}_{l=1}^L$, $\{\phi^l\}_{l=1}^L$, γ set of parameters as in Algorithm 5

$\{\bar{\mathbf{U}}_{old}^l\}_{l=1}^L$ the average subgradient of the last iteration of Algorithm 5

$\{\mathbf{W}_{old}^l\}_{l=1}^L$ the set of parameters learnt by Algorithm 5

$\mathbf{t}_{old}$ number of temporal iterations of Algorithm 5

Initialization:

$\hat{\mathbf{W}}_0^l = \mathbf{W}_{old}^l \circ mean(\mathbf{W}_{old}^l),\ \bar{\mathbf{U}}_0^l = \bar{\mathbf{U}}_{old}^l \circ 0\ \forall l \in \{1..L\}$

$\mathbf{W}_t^0 = 0\ \forall t$

1. for t = 1,2,3,... do
   (a) for $l = 1$ to L
      i. Given the function $\mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}$, compute a subgradient $\mathbf{U}_{t,new}^l \in \partial \mathbf{L}_{\mathbf{t},\mathbf{W}_t^{l-1}}$
      ii. $\bar{\mathbf{U}}_{t,new}^l = \frac{t-1}{t}\bar{\mathbf{U}}_{t-1,new}^l + \frac{1}{t}\mathbf{U}_{t,new}^l$
      iii. $\bar{\mathbf{U}}_t^l = \bar{\mathbf{U}}_{old}^l \circ \bar{\mathbf{U}}_{t,new}^l$
      iv. $\hat{\mathbf{W}}_t^l = \underset{\mathbf{W}}{\operatorname{argmin}}\ \bar{\mathbf{U}}_t^l \mathbf{W} + \psi^l(\mathbf{W}) + < \frac{\gamma}{\sqrt{\mathbf{t}_{old} \circ \mathbf{t}}}, h(\mathbf{W}) >$
      v. $\mathbf{W}_t^l = \mathbf{W}_{old}^l \circ (\mathbf{W}_{t,new}^{l-1} + \hat{\mathbf{W}}_{t,new}^l)$
   (b) $\mathbf{W}_t = \mathbf{W}_t^L$

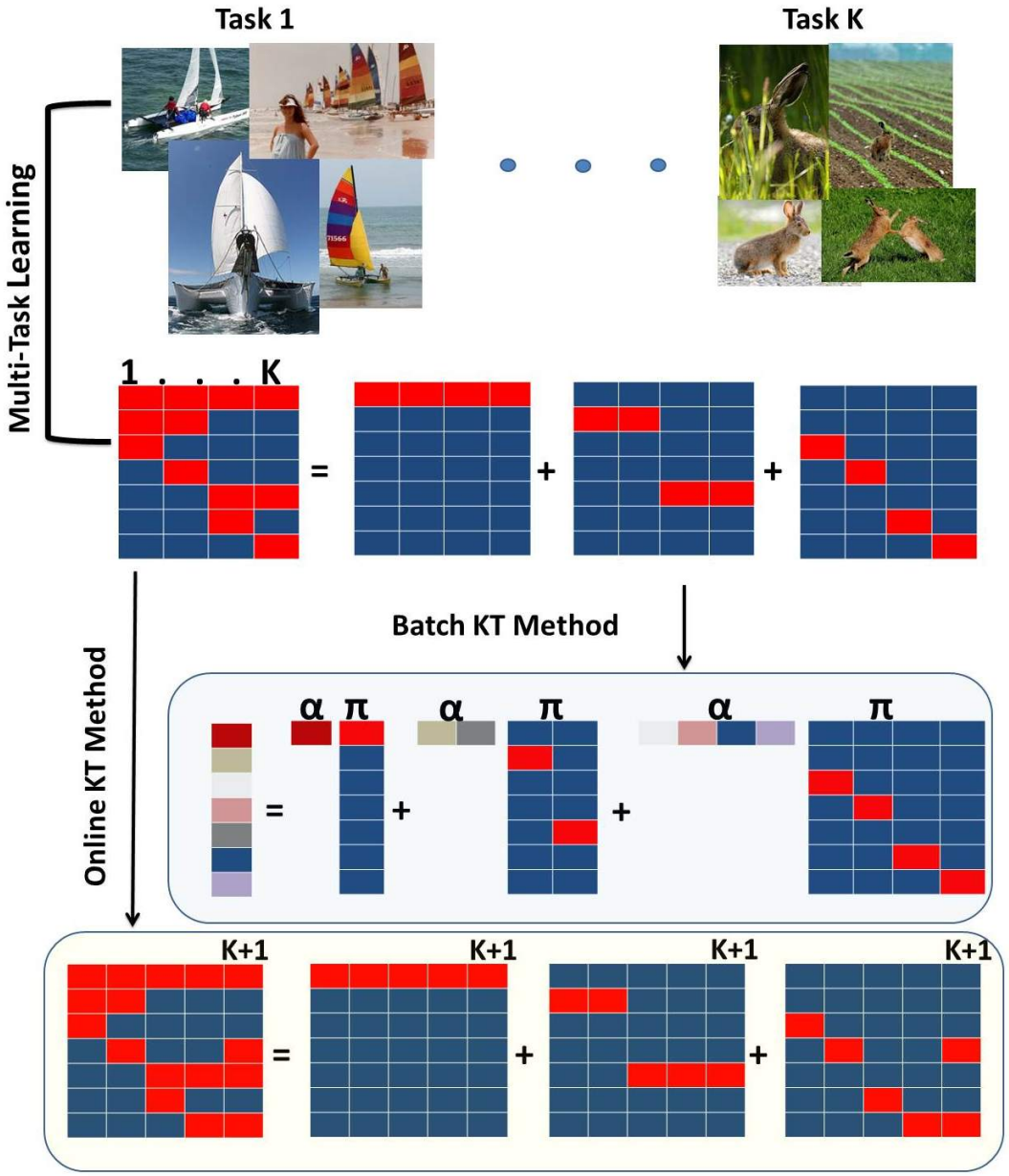


Figure 5.9: Illustration of the knowledge transfer approach, explained from top to bottom. First we pre-train jointly K recognition tasks. The output of the multi-task ('MTL') training stage can be visualized as a sum of matrices each corresponding to a different level of shared information among tasks. This output is passed on to the knowledge-transfer ('KT') phase where task '$K+1$' is learnt given a small set of examples. We consider two methods for transferring the hierarchical structure of shared information. The first method is a batch algorithm, which is based on dimensionality reduction to the sub-spaces of common information. 'α' denotes the linear classifier coefficients of the new task in each reduced sub-space, while 'π' denotes the dimensionality reduction projection. The second method is an online algorithm with knowledge transfer happening in the original feature space, where the pre-trained classifiers are used to bootstrap the classifier of the novel task. The color blue corresponds to value '0' and red to '1'.

5.5 Knowledge-Transfer Experiments

In this section we evaluate our algorithms for knowledge transfer in small sample scenarios. We start by comparing the different methods on controlled synthetic data in Section 5.5.1. We then test the performance of our method using several real datasets employing different image representation schemes in two different settings: *medium size*, with several tens of classes and a dimensionality of 1000 features as image representation in Section 5.5.2; and *large size*, with hundreds of classes and an image representation of 21000 features in Section 5.5.3. We also compared the performance in a '1-vs-Rest' setting and in a setting with a common negative class (as in clutter).

The methods used for comparison are the following:

Batch methods

- *KT-Batch-H*: corresponds to Algorithm 6 where knowledge-transfer is based on the projection matrices extracted from the batch cascade learning.
- *KT-Batch-NoReg*: here knowledge transfer corresponds to Algorithm 6 with $L = 1$ and $\phi = 0$, where information is transferred from the previously learnt models which were learnt in a single level with no regularization and no incentive to share information.

Online methods

- *KT-On-H*: corresponds to Algorithm 7 where we transfer information given the full cascade of regularization functions.
- *KT-On-L12*: corresponds to Algorithm 7 with $L = 1$ and $\lambda = 0$, where a single level of information transfer is based on models trained to share information between all tasks equally.

Baseline methods from Section 5.3, without Knowledge Transfer:

- *NoKT-Batch*: corresponds to the multi-task batch algorithm with $L = 1$ and $\phi = 0$.
- *NoKT-On-NoReg*: corresponds to the multi-task online Algorithm 5 with $L = 1$ and $\phi = 0$.

5.5.1 Synthetic Data

In order to understand when our proposed method is likely to achieve improved performance in knowledge transfer, we tested it in a controlled manner on the synthetic dataset described in Section 5.3.1.

To test Algorithms 6 and 7 we trained 99 tasks using the multi-task batch and online algorithms with only 99 tasks, keeping the remaining task aside as the unknown novel task. Each known task was trained with 50 examples, with 10 repetitions over the data for the online Algorithm 5. After this multi-task pre-processing had finished, we trained the left out task using Algorithms 6 and 7 with either 1-10, 20 or 50 examples (with 100 repetitions in the online Algorithm 7). This was done 100 times, leaving out in turn each of the original tasks. In the batch knowledge-transfer we chose the rank of each projection matrix to keep 99.9% of the variance in the data. In the hierarchical knowledge-transfer this resulted in approximately 10 dimensions at the top level of the cascade, and 90 dimensions at the lowest level of the cascade.

As can be seen in Fig. 5.10a, our Knowledge-Transfer methods based on shared multi-task models achieve the best performance as compared to the alternative methods. The online knowledge-transfer method achieves the best results on this dataset. Note that with very small samples, the method which attempts to share information with all tasks equally - *KT-On-L12* - achieves the best performance. As the sample increases to 50, Algorithm 7 is able to perform better by exploiting the different levels of sharing given by the hierarchical approach *KT-On-H*. For both online Knowledge-Transfer options we see a significant improvement in performance as compared to the online with no knowledge transfer approach, *NoKT-On-NoReg*.

Looking at the batch method we see that Knowledge-Transfer based on sharing information between the original models, *KT-Batch-H*, outperforms significantly the knowledge-transfer based on no sharing of information in the original model training, *KT-Batch-NoReg*. The *KT-Batch-NoReg* actually performs no better than the no knowledge-transfer approach *NoKT-Batch*.

It is also interesting to note that the difference in average performance between the novel task to the pre-trained tasks is less than 0.5% for 50 training examples when using the hierarchical knowledge-transfer. This indicates that in this experiment

our hierarchical knowledge-transfer method reaches the potential of sharing as in the multi-task method, which outperforms all other methods on this synthetic data.

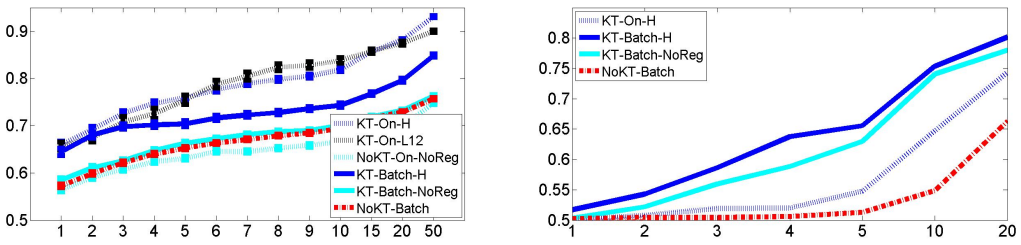


Figure 5.10: Results: in all plots the 'Y'-axis corresponds to the average accuracy over all tasks, and the 'X'-axis to the sample size. (a) Results for Synthetic data experiment. (b) results for the large size imagenet experiment.

5.5.2 Medium Size

We tested our method with real data, starting with a moderate problem size and only 31 classes. For these experiments we used a subset of the ILSVRC(2010) challenge [97], which is an image dataset organized according to the WordNet hierarchy. From this huge dataset we chose 31 classes (synsets)[1] for the set of pre-trained known classes with many training examples. This group of classes was chosen heuristically to contain varying levels of relatedness among classes, grouping together various terrestrial, aerial and sea vehicles, buildings, sea animals etc. For the novel classes with small sample we considered 30 randomly chosen classes from the remaining 969 classes in the dataset. The set of features used to describe images in this data set is based on the Sift features quantized into a codebook of 1000 words, which was tf-idf normalized.

We considered binary learning tasks where each chosen class, either pre-trained or novel, is contrasted with a set of images (negative examples) chosen from a group of different classes. The negative set was constructed in two ways: In the *1-vs-Rest*

[1] quail, partridge, hare, Angora rabbit, wood rabbit, indri, Madagascar cat, orangutan, chimpanzee, gorilla, fire engine, garbage truck, pickup truck, trailer truck, police wagon, recreational vehicle, half track, snowmobile, tractor, tricycle, fiddler crab, king crab, silver salmon, rainbow trout, striper, airliner, warplane, lifeboat, catamaran, boathouse and church building.

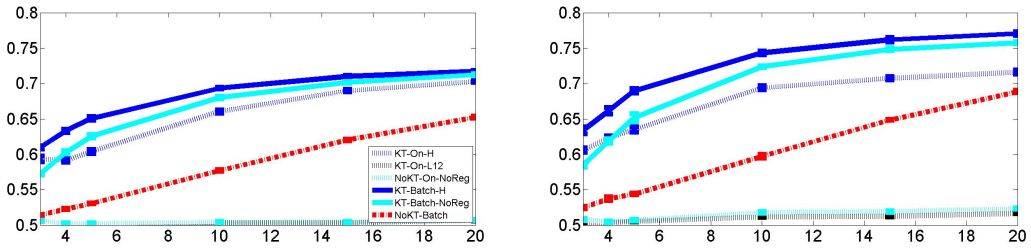


Figure 5.11: Mid-size experiment accuracy results, with (a) common negative set, and (b) 1-vs-Rest. In all plots the 'Y'-axis corresponds to the average accuracy over all tasks, and the 'X'-axis to the sample size.

condition the negative set of classes, the *Rest*, was defined as the group of 31 classes from which knowledge is transferred. In the second condition the negative set included 31 different classes sampled randomly from the original dataset excluding the small sample classes and the set of pre-trained classes. In this second condition all classes, both pre-trained and novel, had the exact same set of negative examples. This condition resembles previous experiments with knowledge-transfer [7, 52], where all tasks share the same negative set.

In both conditions the pre-trained models were trained using 480 positive examples and 480 negative examples. For the positive set of the small sample classes, we considered a sample size in the range 1-20. For the negative set we used all examples from each negative group (480 examples per class in the 1-vs-Rest condition and 480 in total in the second condition). Examples were weighted according to sample size. For the pre-trained models we used a validation set of 60 examples per class; we used 100 examples per each class as its test set.

We considered each of the novel 30 classes separately. The experiment was repeated 8 times with different random splits of the data, for both the pre-trained and novel tasks. In the batch knowledge transfer methods we set the projection rank to maintain 99.9% of the variance in the original models learnt.

Results for the condition with shared negative set are shown in Fig. 5.11a. Results for the 1-vs-Rest condition are shown in Fig. 5.11b. We see that knowledge-transfer methods achieve improved performance as compared to alternative methods. In both conditions the best performer is the hierarchical batch approach

for Knowledge-transfer, *KT-Batch-H*. The poor performance of the *KT-Online-L12* can be explained by the fact that the regularization coefficient ϕ chosen by cross-validation during the multi-task pre-learning phase of the pre-trained models was chosen to be very low, indicating that a single level of sharing is not sufficient for this data.

5.5.3 Large Size

As the ultimate knowledge transfer challenge, we tested our method with real data and large problem size. Thus we used all 1000 classes from the ILSVRC(2010) challenge [97]. Each image was represented by a vector of 21000 dimensions, following the representation scheme used by [9]. 900 classes were chosen randomly as pre-trained classes, while the remaining 100 classes were used as the novel classes with small sample. We considered *1-vs-Rest* tasks as explained above. Pre-trained tasks were trained using 500 examples from the positive class and 500 examples chosen randomly from the remaining 899 classes. We used the online Algorithm 5 with 2 repetitions over the training data to train pre-trained tasks.

During test, the set of negative examples in the *1-vs-Rest* condition was chosen randomly from all of the dataset, total of 999 classes. We used the labeled test set provided by [97]. As small sample we considered 1-20 examples per class. Due to the original large image representation, in the batch knowledge-transfer methods we fixed the projection rank to maintain only 80% of the variance in the original models learnt.

We note that once the pre-trained models are computed, each step of training with the projected batch approach is faster than each step of the online approach as the online Algorithm 7 needs at each step to consider all the pre-trained parameters in order to compute the regularization value, while the batch Algorithm 6 considers these parameters only once during the projection phase. Using a big image representation as we do the online methods becomes computationally expensive if repeating the experiment for each of the novel small samples classes separately.

Results are shown in Fig. 5.10b. Clearly all methods inducing information sharing outperformed significantly the batch and online learning with no sharing. The *NoKT-on-NoReg* method performed poorly similarly to *NoKT-batch* and was omit-

ted for brevity. *KT-on-L12* also performed poorly due to the very low regularization parameter ϕ automatically chosen.

5.6 Summary

We presented a hierarchical approach for information sharing both in joint learning and knowledge-transfer scenarios. Our methods are based on a cascade of regularized minimization problems designed to induce implicit hierarchical sharing of information. Particular care was put into the design of efficient methods which enable the scaling up to large settings, considering a wide range of hierarchical relations among task.

For the joint learning scenario we described two efficient batch and online learning algorithms implementing the cascade. For the online algorithm we provided a regret bound from which it follows that the average regret of our learning method converges. The method was tested both on synthetic data and seven real datasets, showing significant advantage over baseline methods, and similar or improved performance as compared to alternative state of the art methods.

Using the output of the joint learning phase we are able to transform information from pre-trained tasks to novel tasks. We designed two different methods to transfer knowledge from a set of known tasks to new tasks. Knowledge transfer is typically needed when the sample available for the training of the new tasks is too small. Our methods were designed to be useful under conditions where either the object representation is very large (the knowledge-transfer batch method), or with streaming data which can be best managed with an online approach (the knowledge-transfer online method). Basing our methods on pre-trained models learnt jointly we are able to transfer varying levels of shared information according to the implicit hierarchy captured during the joint training phase. We conducted a number of experiments, testing different settings of knowledge-transfer on real and synthetic datasets. Unambiguously our proposed knowledge transfer methods achieved the highest performance in all the experiments. To our knowledge we demonstrated the largest visual object recognition knowledge-transfer experiment of this type.

6

Epilogue

In this thesis I presented hierarchical models of recognition tasks, considering both the case when a hierarchy is known and the case when it is unknown. In the unknown case, I proposed two different approaches; a formal model together with a structure discovery algorithm and an implicit approach for exploiting the hierarchal structure while avoiding the hard problem of its discovery.

I started by considering the task of novel subclass detection (chapter 2) based on a known hierarchy. First, considering an accepting level enables our approach to separate novel subclass detection from poor signal detection, a distinction which is challenging for the common approaches to novelty detection, which are based only on a rejection cue in a non hierarchical architecture. I also show the importance of knowing the veridical hierarchy.

In chapter 3, I presented a general hierarchical multi-task learning framework, extending the commonly used two-level hierarchy in multi-task settings, to a multi-level hierarchy. A full hierarchical view of multi-task learning enables tasks of varying degrees of relatedness to contribute in the learning process. I consider a top-down cascade learning approach, starting from the most abstract level, where at each level of the hierarchy a single optimal hypothesis for the unified task (unifying all tasks at that level) is chosen. This hypothesis is then used to define the inductive bias for the next level in the hierarchy. The complexity of each learning stage is determined by the size of the hypothesis equivalence class, defined by applying the shared transformations in each level to the chosen hypothesis of the previous level. I state sufficient conditions for the optimality of our approach and provide generalization guarantees. Finally I described an experimental setting demonstrating the potential in the proposed approach.

Following chapters 2 and 3, where I assume the hierarchy is known, I present in chapter 4, a general hierarchical model of classes and an algorithm to infer it from data. This is a unified model for both the class-membership and part-membership hierarchies. I provide run time guarantees based on the complexity of the rela-

tionships within a given set of classes and prove that the output of the algorithm indeed obeys the defined constraints of the model. I also present the usage of such a hierarchy for the case of classification based on conjunction of features.

In chapter 5, I presented an implicit hierarchical approach to information sharing based on a cascade of regularized minimization problems. The implicit approach which avoids the hard problem of explicitly discovering the hierarchy together with the efficient algorithm enabled us to scale up to large scenarios in both the multi-task and knowledge-transfer settings. Our method can also be applied to an online setting where I assume a continuous flow of data and prove that the average regret of the online learning algorithm converges. An extensive set of experiments was conducted, considering a large set of datasets on which I showed very promising results, compared to the natural baselines and existing state of the art methods.

The general use of hierarchies in the context of recognition tasks and the specific contributions in this thesis serves two purposes. Improving the original recognition tasks and enriching the knowledge base of recognition systems by considering an ontology. In the context of improving recognition tasks, the main contribution of this thesis lies in the methods suggested for information sharing, where I contribute to the existing hierarchical methods by providing a novel theoretical framework, together with generalization guarantees and by presenting efficient algorithms dealing with the unknown hierarchy scenario, scaling up to much larger settings with promising recognition results.

The second type of contribution, is in the semantic organization of knowledge. With the advance in recognition tasks, such as specific object as well as object class recognition, the field today is advancing back to its historical origins, where a perceptual system is considered as part of a whole AI system, which can harness its ontological knowledge towards inference of its' perceptual environment. For instance, the presented novel subclass detection algorithm is a sheer instance of such inference, where the known hierarchical organization of classes enables a semantic inference of unknown signals. To this end, I believe the SIPO model is important in its contribution to the problem of explicitly structuring the knowledge base.

The structure discovery algorithm based on the SIPO model is efficient in the sense that its complexity is dependent on the size of the output. However considering groupings of concepts, the size of the output could be exponential. My hypothesis

is that for an ontology to be effective it has to consider sparse relations among the concepts it contains. Assuming that true relations among concepts are sparse, the output size would not be exponential and the proposed SIPO algorithm would deal with the data efficiently. For large scale scenarios considering the magnitude of concepts a modern system should deal with and the noisy nature of signals supporting these concepts (which could challenge the discovery of the true sparse relations), I believe it would be beneficial to focus future research on scaling up the structure discovery algorithm by relaxing the requirement that all intersections are found in the SIPO model.

Finally, considering advancing both recognition tasks and the modeling of their relations is essentially a classical chicken and egg problem, as I have shown, the relations can help the tasks while clearly having better models of the tasks could assist the modeling of their relations. A future direction which I view as particularly interesting, is trying to infer the hierarchical structure of tasks from the task specific parameters learnt by the implicit information sharing method I present, thereby initiating a bootstrapping learning approach, starting from an implicit hierarchical representation and ending up with an explicit hierarchical model.

References

[1] E. Rosch. **Basic Objects in Natural Categories.** *Cognitive Psychology*, **8**(3):382–439, 1976. 1

[2] R. Kiani, H. Esteky, K. Mirpour, and K. Tanaka. **Object Category Structure in Response Patterns of Neuronal Population in Monkey Inferior Temporal Cortex**. *Journal of Neurophysiology*, **97**(6):4296, 2007. 1

[3] Ilan Kadar and Ohad Ben-Shahar. **A perceptual paradigm and psychophysical evidence for hierarchy in scene gist processing**. *Journal of vision*, **12**(13), 2012. 1

[4] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. 1

[5] David A Huffman. **A method for the construction of minimum-redundancy codes**. *Proceedings of the IRE*, **40**(9):1098–1101, 1952. 1

[6] L. Brieman, J.H. Friedman, R.A. Olshen, and C.J. Stone. **Classification and regression trees**. *Wadsworth Inc*, **67**, 1984. 1

[7] A. Zweig and D. Weinshall. **Exploiting Object Hierarchy: Combining Models from Different Category Levels**. *Proc. ICCV*, 2007. 2, 4, 5, 7, 8, 9, 110

[8] A. Torralba, K.P. Murphy, and W.T. Freeman. **Sharing visual features for multiclass and multiview object detection**. *T-PAMI, IEEE*, **29**(5):854–869, 2007. 2, 4, 5, 10, 12

[9] B. Zhao, L. Fei-Fei, and E.P. Xing. **Large-Scale Category Structure Aware Image Categorization**. *Proc. NIPS*, 2011. 2, 4, 5, 9, 80, 89, 100, 101, 102, 111

[10] Z. Kang, K. Grauman, and F. Sha. **Learning with Whom to Share in Multi-task Feature Learning**. *NIPS*, 2011. 2, 4, 5, 10, 12, 80, 89, 97, 102

[11] P. Jawanpuria and J.S. Nath. **A Convex Feature Learning Formulation for Latent Task Structure Discovery**. *ICML*, 2012. 2, 4, 5, 10, 12

[12] S. Kim and E.P. Xing. **Tree-Guided Group Lasso for Multi-Task Regression with Structured Sparsity**. *ICML*, 2010. 2, 4, 5, 10, 12, 80, 89, 92, 100, 102

REFERENCES

[13] Yangchi Chen, Melba M Crawford, and Joydeep Ghosh. **Integrating support vector machines in a hierarchical output space decomposition framework**. In *Geoscience and Remote Sensing Symposium, 2004. IGARSS'04. Proceedings. 2004 IEEE International*, **2**, pages 949–952. IEEE, 2004. 2, 3

[14] Marcin Marszałek and Cordelia Schmid. **Constructing Category Hierarchies for Visual Recognition**. In *European Conference on Computer Vision*, **IV** of *LNCS*, pages 479–491. Springer, oct 2008. 2, 3, 4

[15] Gregory Griffin and Pietro Perona. **Learning and using taxonomies for fast visual categorization**. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2, 3

[16] Alexander Binder, Motoaki Kawanabe, and Ulf Brefeld. **Efficient classification of images with taxonomies**. In *Computer Vision–ACCV 2009*, pages 351–362. Springer, 2010. 2, 3

[17] S. Bengio, J. Weston, and D. Grangier. **Label embedding trees for large multi-class tasks**. *NIPS*, 2010. 2, 3

[18] T. Gao and D. Koller. **Discriminative learning of relaxed hierarchy for large-scale visual recognition**. *ICCV*, 2011. 2, 3, 4

[19] Jia Deng, Sanjeev Satheesh, Alexander C Berg, and Li Fei-Fei. **Fast and balanced: Efficient label tree learning for large scale object recognition**. *Advances in Neural Information Processing Systems*, **24**:567–575, 2011. 2, 3

[20] Ofer Dekel, Joseph Keshet, and Yoram Singer. **Large margin hierarchical classification**. In *Proceedings of the twenty-first international conference on Machine learning*, page 27. ACM, 2004. 2

[21] O. Dekel. **Distribution-Calibrated Hierarchical Classification**. 2

[22] D. Zhou, L. Xiao, and M. Wu. **Hierarchical classification via orthogonal transfer**. In *ICML*, 2011. 2

[23] A. Bar-Hillel and D. Weinshall. **Subordinate class recognition using relational object models**. *Proc. NIPS*, **19**, 2006. 2, 30

[24] S.J. Hwang, K. Grauman, and F. Sha. **Learning a Tree of Metrics with Disjoint Visual Features**. *NIPS*, 2011. 2

[25] N. Verma, D. Mahajan, S. Sellamanickam, and V. Nair. **Learning hierarchical similarity metrics**. In *CVPR*. IEEE, 2012. 2

[26] Jia Deng, Alexander C Berg, Kai Li, and Li Fei-Fei. **What does classifying more than 10,000 image categories tell us?** In *Computer Vision–ECCV 2010*, pages 71–84. Springer, 2010. 2

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. **ImageNet: A Large-Scale Hierarchical Image Database**. In *CVPR09*, 2009. 3

[28] John C Platt, Nello Cristianini, and John Shawe-Taylor. **Large margin DAGs for multiclass classification**. *Advances in neural information processing systems*, **12**(3):547–553, 2000. 3

[29] Volkan Vural and Jennifer G Dy. **A hierarchical method for multi-class support vector machines**. In *Proceedings of the twenty-first international conference on Machine learning*, page 105. ACM, 2004. 3

[30] E. Bart, I. Porteous, P. Perona, and M. Welling. **Unsupervised learning of visual taxonomies**. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8, 2008. 4

[31] J. Sivic, B.C. Russell, A. Zisserman, W.T. Freeman, and A.A. Efros. **Unsupervised discovery of visual object class hierarchies**. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8, 2008. 4

[32] T Griffiths, M Jordan, and J Tenenbaum. **Hierarchical topic models and the nested Chinese restaurant process**. *Advances in neural information processing systems*, **16**:106–114, 2004. 4

[33] S. Fidler and A. Leonardis. **Towards scalable representations of object categories: Learning a hierarchy of parts**. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, pages 1–8, 2007. 4

[34] Narendra Ahuja and Sinisa Todorovic. **Learning the taxonomy and models of categories present in arbitrary images**. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007. 4

[35] B. Epshtein and S. Ullman. **Semantic hierarchies for recognizing objects and parts**. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, pages 1–8, 2007. 4

[36] Boris Epshtein and S Uliman. **Feature hierarchies for object classification**. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, **1**, pages 220–227. IEEE, 2005. 4

[37] Sebastian Thrun and L. Pratt. *Learning To Learn*. Kluwer Academic Publishers, November 1997. 4

[38] R. Caruana. **Multitask learning**. *Machine Learning*, **28**(1):41–75, 1997. 4, 5

[39] J. Baxter. **A model of inductive bias learning**. *J. Artif. Intell. Res. (JAIR)*, **12**:149–198, 2000. 4, 5, 9, 36, 39

[40] M. Fink, S. Shalev-Shwartz, Y. Singer, and S. Ullman. **Online multiclass learning by interclass hypothesis sharing**. In *Proceedings of the 23rd international conference on Machine learning*, pages 313–320. ACM, 2006. 4, 5

[41] G. Obozinski, B. Taskar, and M. Jordan. **Joint covariate selection for grouped classification**. *Department of Statistics, U. of California, Berkeley, Tech. Rep*, **743**, 2007. 4, 5, 6, 79

[42] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. **Multi-task learning for classification with Dirichlet process priors**. *The Journal of Machine Learning Research*, **8**:35–63, 2007. 4, 6

[43] Y. Zhang and D.Y. Yeung. **A Convex Formulation for Learning Task Relationships in Multi-Task Learning**. In *UAI*, 2010. 4, 5

[44] Y. Amit, M. Fink, N. Srebro, and S. Ullman. **Uncovering shared structures in multiclass classification**. In *Proceedings of the 24th international conference on Machine learning*, pages 17–24. ACM New York, NY, USA, 2007. 4, 5, 6

[45] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. **Boosting for transfer learning**. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007. 4, 7, 8

[46] Ulrich Rückert and Stefan Kramer. **Kernel-based inductive transfer**. In *Machine Learning and Knowledge Discovery in Databases*, pages 220–233. Springer, 2008. 4, 7

[47] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. **Convex multi-task feature learning**. *Machine Learning*, **73**(3):243–272, 2008. 4, 5, 6

[48] A. Quattoni, M. Collins, and T. Darrell. **Transfer learning for image classification with sparse prototype representations**. In *CVPR*, 2008. 4, 5, 7

[49] J. Duchi and Y. Singer. **Boosting with structural sparsity**. In *ICML*, pages 297–304. ACM, 2009. 4, 5, 6

[50] H. Daume III. **Bayesian multitask learning with latent hierarchies**. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 135–142. AUAI Press, 2009. 4, 6

[51] Michael Stark, Michael Goesele, and Bernt Schiele. **A shape-based object class model for knowledge transfer**. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 373–380. IEEE, 2009. 4, 8

[52] T. Tommasi, F. Orabona, and B. Caputo. **Safety in numbers: Learning categories from few examples with multi model knowledge transfer**. In *CVPR*, 2010. 4, 8, 110

[53] Yusuf Aytar and Andrew Zisserman. **Tabula rasa: Model transfer for object category detection**. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2252–2259. IEEE, 2011. 4, 8

[54] S. Shalev-Shwartz, Y. Wexler, and A. Shashua. **ShareBoost: Efficient Multiclass Learning with Feature Sharing**. *Proc. NIPS*, 2011. 4, 5, 6

[55] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. **Sparse coding for multitask and transfer learning**. In *Proceedings of the 30th international conference on Machine learning*, 2013. 4, 7

[56] S.J. Pan and Q. Yang. **A survey on transfer learning**. *Knowledge and Data Engineering, IEEE Transactions on*, **22**(10):1345–1359, 2010. 5, 8

[57] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. **Maximum-margin matrix factorization**. In *Advances in neural information processing systems*, pages 1329–1336, 2004. 6

[58] Sebastian Thrun. **Is learning the n-th thing any easier than learning the first?** *Advances in neural information processing systems*, pages 640–646, 1996. 7

[59] S. Ben-David and R.S. Borbely. **A notion of task relatedness yielding provable multiple-task learning guarantees**. *Machine Learning*, **73**(3):273–287, 2008. 9, 33, 34, 35, 36, 38, 39, 44, 54, 56

[60] George A. Miller. **WordNet: A Lexical Database for English**. *Communications of the ACM*, **38**:39–41, 1995. 10

[61] M. Markou and S. Singh. **Novelty detection: a review-part 1: statistical approaches**. *Signal Processing*, **83**(12):2499 – 2521, 2003. 10, 15

[62] M. Markou and S. Singh. **Novelty detection: a review-part 2: neural network based approaches**. *Signal Processing*, **83**(12):2481–2497, 2003. 10, 15

[63] D.M.J. Tax and R.P.W. Duin. **Support Vector Data Description**. *Machine Learning*, **54**(1):45–66, 2004. 10

[64] B. Scholkopf, R.C. Williamson, A.J. Smola, J. Shawe-Taylor, and J. Platt. **Support vector method for novelty detection**. *NIPS*, 2000. 10, 29

[65] D.Y. Yeung and C. Chow. **Parzen-window network intrusion detectors**. *ICPR*, 2002. 10

[66] CP Diehl and II JB. **Real-time object classification and novelty detection forcollaborative video surveillance**. *IJCNN*, 2002. 11

[67] S. Chopra, R. Hadsell, and Y. LeCun. **Learning a similarity metric discriminatively, with application to face verification**. *Proc. of Computer Vision and Pattern Recognition Conference*, 2005. 11

[68] Daphna Weinshall, Alon Zweig, Hynek Hermansky, Stefan Kombrink, Frank W Ohl, J Bach, Luc Van Gool, Fabian Nater, Tomas Pajdla, Michal Havlena, et al. **Beyond novelty detection: Incongruent events,**

when general and specific classifiers disagree. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **34**(10):1886–1901, 2012. 15

[69] A. Bar-Hillel, T. Hertz, and D. Weinshall. **Efficient learning of relational object class models**. *Proc. ICCV*, 2005. 21

[70] B. Leibe, A. Leonardis, and B. Schiele. **Robust object detection with interleaved categorization and segmentation**. *IJCV*, **77**(1):259–289, 2008. 21

[71] G. Griffin, A. Holub, and P. Perona. **Caltech-256 Object Category Dataset**. Technical Report UCB/CSD-04-1366, California Institute of Technology, 2007. 24

[72] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. **Coding facial expressions with gabor wavelets**. *Proc. ICAFGR*, pages 200–205, 1998. 25

[73] V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000. 34

[74] Leslie G. Valiant. **A theory of the learnable**. *Communications of the ACM*, **27**(11):1134–1142, 1984. 55, 58

[75] David Haussler. **Quantifying inductive bias: AI learning algorithms and Valiant's learning framework**. *Artificial intelligence*, **36**(2):177–221, 1988. 55, 58

[76] Mario Marchand and John Shawe Taylor. **The set covering machine**. *The Journal of Machine Learning Research*, **3**:723–746, 2003. 55

[77] Mohak Shah, Mario Marchand, and Jacques Corbeil. **Feature selection with conjunctions of decision stumps and learning from microarray data**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **34**(1):174–186, 2012. 55

[78] R. Tibshirani. **Regression shrinkage and selection via the lasso**. *J. Royal Statist. Soc.. B*, **58**:267–288, 1996. 79

[79] J. Friedman, T. Hastie, and R. Tibshirani. **A note on the group lasso and a sparse group lasso**. *Arxiv preprint arXiv:1001.0736*, 2010. 79

[80] M. Yuan and Y. Lin. **Model selection and estimation in regression with grouped variables**. *J. Royal Statist. Soc.. B*, **68**(1):49–67, 2006. 79

[81] K. Crammer and Y. Singer. **On the algorithmic implementation of multiclass kernel-based vector machines**. *JMLR*, **2**:265–292, 2002. 82

[82] S.J. Wright, R.D. Nowak, and M.A.T. Figueiredo. **Sparse reconstruction by separable approximation**. *Signal Processing, IEEE Transactions on*, **57**(7):2479–2493, 2009. 82, 84

[83] A. Beck and M. Teboulle. **A fast iterative shrinkage-thresholding algorithm for linear inverse problems**. *SIAM Journal on Imaging Sciences*, **2**(1):183–202, 2009. 84

[84] I. Daubechies, M. Defrise, and C. De Mol. **An iterative thresholding algorithm for linear inverse problems with a sparsity constraint**. *Communications on pure and applied mathematics*, **57**(11):1413–1457, 2004. 84

[85] P. Sprechmann, I. Ramírez, G. Sapiro, and Y.C. Eldar. **C-HiLasso: A Collaborative Hierarchical Sparse Modeling Framework**. *Signal Processing, IEEE Transactions on*, **59**(9):4183–4198, 2011. 84

[86] L. Xiao. **Dual averaging methods for regularized stochastic learning and online optimization**. *JMLR*, 2010. 85, 88

[87] H. Yang, Z. Xu, I. King, and M. Lyu. **Online learning for group lasso**. *ICML*, 2010. 85

[88] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. **Gradient-based learning applied to document recognition**. *Proceedings of the IEEE*, **86**(11):2278–2324, 1998. 97

[89] J.J. Hull. **A database for handwritten text recognition research**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **16**(5):550–554, 1994. 97

[90] A. Krizhevsky and GE Hinton. **Learning multiple layers of features from tiny images**. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 97

[91] G. Griffin, A. Holub, and P. Perona. **Caltech-256 object category dataset**. 2007. 97

[92] A. Quattoni and A. Torralba. **Recognizing indoor scenes**. *CVPR*, 2009. 97, 98, 99

[93] Aude Oliva and Antonio Torralba. **Modeling the shape of the scene: A holistic representation of the spatial envelope**. *International journal of computer vision*, **42**(3):145–175, 2001. 98

[94] A. Rahimi and B. Recht. **Random features for large-scale kernel machines**. *NIPS*, 2007. 98

[95] P. Gehler and S. Nowozin. **On feature combination for multiclass object classification**. In *ICCV*, 2009. 98, 99

[96] L.J. Li, H. Su, E.P. Xing, and L. Fei-Fei. **Object bank: A high-level image representation for scene classification and semantic feature sparsification**. *NIPS*, 2010. 98, 99

[97] A. Berg, J. Deng, and L. Fei-Fei. **Large scale visual recognition challenge 2010**, 2010. 100, 101, 102, 109, 111

[98] A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. **Learnability and the Vapnik-Chervonenkis dimension**. *Journal of the ACM (JACM)*, **36**(4):929–965, 1989. V

A

Appendix

A.1 Chapter 3 Appendix

A.1.1 Proof of lemma 6

Proof For clarity in the proof below, we omit the domain $\mathcal{X}\times\{0,1\}$ when referring to (x,b) and use the shorthand $hf(X)$ for $h\circ f(X)$.

$$
\begin{aligned}
&Er^P(h\circ g\circ f) = P(\{(x,b) : hgf(x)\neq b\}) = \\
&P(\{(x,b) : hgf(x)\neq b, hf(x)=1\}) + P(\{(x,b) : hgf(x)\neq b, hf(x)=0\}) = \quad \text{(A.1)}\\
&P(\{(x,b) : hg(x)hf(x)\neq b, hf(x)=1\}) + P(\{(x,b) : hgf(x)\neq b, hf(x)=0\}) = \\
&P(\{(x,b) : hg(x)\neq b, hf(x)=1\}) + P(\{(x,b) : hgf(x)\neq b, hf(x)=0\}) = \\
&P(\{(x,b) : hg(x)\neq b, hf(x)=1\}) + P(\{(x,b) : hgf(x)\neq b, hf(x)=0\}) + \\
&P(\{(x,b) : hg(x)\neq b, hf(x)=0\}) - P(\{(x,b) : hg(x)\neq b, hf(x)=0\}) = \\
&P(\{(x,b) : hg(x)\neq b\}) + P(\{(x,b) : hgf(x)\neq b, hf(x)=0\}) - \\
&P(\{(x,b) : hg(x)\neq b, hf(x)=0\}) = \quad \text{(A.2)}\\
&P(\{(x,b) : hg(x)\neq b\}) + P(\{(x,b) : b=1, hg(x)=1, hf(x)=0\}) + \\
&P(\{(x,b) : b=1, hg(x)=0, hf(x)=0\}) - P(\{(x,b) : b=1, hg(x)=0, hf(x)=0\}) - \\
&P(\{(x,b) : b=0, hg(x)=1, hf(x)=0\}) = \\
&P(\{(x,b) : hg(x)\neq b\}) + P(\{(x,b) : b=1, hg(x)=1, hf(x)=0\}) - \\
&P(\{(x,b) : b=0, hg(x)=1, hf(x)=0\}) = \\
&Er^P(h\circ g) - P(\{(x,b) : b=0, hg(x)=1, hf(x)=0\}) + \\
&P(\{(x,b) : b=1, hg(x)=1, hf(x)=0\}) = \\
&Er^P(h\circ g) + R^{gf}.
\end{aligned}
$$

Derivations (A.1) and (A.2) above follow from the *transformation-multiplicativity*. □

A.1.2 Indifference Sufficient Conditions Cont.

In this section we continue our analysis of the sufficient conditions of the *indifference* property. We introduce the following notations: $g^* = \arg\min_{g \in \mathcal{G}} Er^P(h \circ g)$, and $\epsilon = Er^P(h \circ g^*) = \epsilon^-_{g^*} + \epsilon^+_{g^*}$ where:

- $\epsilon^-_{g^*} = P(\{(x,b) : b = 1, hg^*(x) = 0\})$
- $\epsilon^+_{g^*} = P(\{(x,b) : b = 0, hg^*(x) = 1\})$

For simplicity we also define:

- $\epsilon^-_f = P(\{(x,b) : b = 1, hf(x) = 0\})$
- $e = P(\{(x,b) : b = 0, hf(x) = 0\})$

Recall the example from Section 3.2.3 where $\epsilon^-_{g^*} = 0$ and $\epsilon^-_f = 0$, in the following we would like to consider richer scenarios when we cannot assume an errorless choice of $g \in \mathcal{G}$ given the specific distribution P and hypothesis $h \in \mathbb{H}$. Whereas this scenario is typically more interesting it is of particular interest in our hierarchical setting where even if we assume that each of the single tasks is realizable the task representing their union might not be.

All sufficient conditions presented in this section are summarized in Table A.1.

We rewrite (3.10) as:

$$\begin{aligned} R^{g^* f} - R^{gf} = [P(\{(x,b) : b = 1, hg(x) = 0\}) - \epsilon^-_{g^*}] * \epsilon^-_f + \\ [P(\{(x,b) : b = 0, hg(x) = 1\}) - \epsilon^+_{g^*}] * e \end{aligned} \tag{A.3}$$

We are interested in analyzing the conditions under which *indifference* occurs. Thus

$$\begin{aligned} [P(\{(x,b) : b = 1, hg(x) = 0\}) - \epsilon^-_{g^*}] * \epsilon^-_f + [P(\{(x,b) : b = 0, hg(x) = 1\}) - \epsilon^+_{g^*}] * e \leq \\ [P(\{(x,b) : b = 1, hg(x) = 0\}) - \epsilon^-_{g^*}] + [P(\{(x,b) : b = 0, hg(x) = 1\}) - \epsilon^+_{g^*}] \end{aligned} \tag{A.4}$$

From the optimality of g^* we know that at least one of its errors is smaller than any other g. It is easy to see that inequality (A.4) is true if both error types of g^* are smaller than g.

Next we analyze the case where g^* has only one type of errors which is smaller than the error of g. We focus our analysis on the behavior of transformation f with respect to distribution P. Specifically, we are interested in the true negative component e and the false negative component ϵ_f^-.

In our hierarchical setting, at each level l of the hierarchy we define a unified task by the distribution $P_{\mathbf{a}_l}$ defined in (3.1). In our analysis of the optimality of the cascade approach in Section 3.2.4, we are interested in the optimal transformation from $\mathcal{G}_l$ which is optimal with respect to the unified task distribution $P_{\mathbf{a}_l}$ and with the transformation from $\mathcal{F}_l$ which is optimal with respect to the distribution of a specific task belonging to level l. Thus, the following discussion of *indifference* focuses on the case where the transformation f is optimal with respect to distribution P_{l_i}, but e and ϵ_f^- are calculated according to distribution $P_{\mathbf{a}_l}$. To clarify this point, consider ϵ_f^- which captures the rate of false negative classification with respect to $P_{\mathbf{a}_l}$; it does not, however, capture the rate of false negative classification with respect to P_{l_i} when some error is due to samples from other tasks.

Another example for a sufficient condition for the *indifference* property is the case where $e = \epsilon_f^-$, when clearly inequality (A.4) is true.

Note that *indifference* occurs when $e = \epsilon_f^-$ regardless of the behavior of the optimal transformation $g^* \in \mathcal{G}$. This is not the case if $e \leq \epsilon_f^-$ or $e \geq \epsilon_f^-$. The conditions concerning $g^* \in \mathcal{G}$ in each of these cases are analyzed in Corollary 1 and 2.

Corollary 1 describes the sufficent condition on g given $e \leq \epsilon_f^-$.

Corollary 1 When $P(\{(x,b) : b = 0, hf(x) = 0\}) \leq P(\{(x,b) : b = 1, hf(x) = 0\})$, inequality (A.4) is true if $P(\{(x,b) : b = 1, hg(x) = 0\}) \leq \epsilon_{g^*}^-$

Proof

$$e \leq \epsilon_f^- \Rightarrow \tag{A.5}$$

$$0 \leq [\epsilon_{g^*}^- - P(\{(x,b) : b = 1, hg(x) = 0\})] * (\epsilon_f^- - 1) +$$
$$[\epsilon_{g^*}^- - P(\{(x,b) : b = 1, hg(x) = 0\})] * (1 - e) \Rightarrow \tag{A.6}$$

$$0 \leq [P(\{(x,b) : b = 1, hg(x) = 0\}) - \epsilon_{g^*}^-] * (1 - \epsilon_f^-) +$$
$$[\epsilon_{g^*}^- - P(\{(x,b) : b = 1, hg(x) = 0\})] * (1 - e) \Rightarrow$$

$$0 \leq [P(\{(x,b) : b = 1, hg(x) = 0\}) - \epsilon_{g^*}^-] * (1 - \epsilon_f^-) +$$
$$[\epsilon_{g^*}^- + \epsilon_{g^*}^+ - P(\{(x,b) : b = 1, hg(x) = 0\}) - \epsilon_{g^*}^+] * (1 - e) \Rightarrow \tag{A.7}$$

$$0 \leq [P(\{(x,b) : b = 1, hg(x) = 0\}) - \epsilon_{g^*}^-] * (1 - \epsilon_f^-) +$$
$$[P(\{(x,b) : b = 0, hg(x) = 1\}) - \epsilon_{g^*}^+] * (1 - e) \tag{A.8}$$

A.5 $\Rightarrow$ A.6, due to the assumption that $P(\{(x,b) : b = 1, hg(x) = 0\}) \leq \epsilon_{g^*}^-$. A.7 $\Rightarrow$ A.8 due to the sub-optimality of g: $P(\{(x,b) : b = 1, hg(x) = 0\}) + P(\{(x,b) : b = 0, hg(x) = 1\}) > \epsilon_{g^*}^- + \epsilon_{g^*}^+$. □

Corollary 2 describes the sufficient condition on g given $\epsilon_f^- \leq e$.

Corollary 2 When $P(\{(x,b) : b = 1, hf(x) = 0\}) \leq P(\{(x,b) : b = 0, hf(x) = 0\})$, inequality (A.4) is true if $P(\{(x,b) : b = 0, hg(x) = 1\}) \leq \epsilon_{g^*}^+$

Proof symmetric to the proof of Corollary 1.

	$g \in \mathcal{G}$	$f \in \mathcal{F}$
1	$P(\{(x,b) : b = 1, hg^*(x) = 0\}) = 0\})$	$P(\{(x,b) : b = 1, hf(x) = 0\}) = 0\})$
2	$\epsilon_{g^*}^- < P(\{(x,b) : b = 1, hg(x) = 0\})$ $\epsilon_{g^*}^+ < P(\{(x,b) : b = 0, hg(x) = 1\})$	- -
3	-	$e = \epsilon_f^-$
4	$P(\{(x,b) : b = 1, hg(x) = 0\}) \leq \epsilon_{g^*}^-$	$e \leq \epsilon_f^-$
5	$P(\{(x,b) : b = 0, hg(x) = 1\}) \leq \epsilon_{g^*}^+$	$e \geq \epsilon_f^-$

Table A.1: A summary of sufficient conditions for the *indifference* property assuming $g \in \mathcal{G}$ and $f \in \mathcal{F}$ are statistically independent (see (3.9)). Each row represents a single set of sufficient conditions. '-' specifies there is no constraint on $g \in \mathcal{G}$ or $f \in \mathcal{F}$.

A.1.3 Learning a Subset of Rectangle Dimensions

The problem of learning an axis-aligned rectangle in R^d has VC-dimension of 2d ([98]). In the following corollary we show that the subproblem of learning $n \leq d$ dimension of the rectangle given the $d - n$ dimensions is an easier learning task.

Corollary 3 Let r be an axis-aligned rectangle in R^d, and let $F(r)$ be the class of all Euclidean shifts and Scale of a chosen set of $n \leq d$ dimensions of r, where all other $d - n$ dimensions are fixed. Then $VCdim(F(r)) \leq 2n$.

Proof Suppose $F(r)$ shatters set U. Thus there exists $h \in F(r)$ such that $\forall x \in U$, $h(x) = 1$. This means that the axis-aligned faces of the rectangle in the $d - n$ fixed dimensions must be located such that all points are within the boundaries of these faces.

Thus for any point $x \in U$ to be labeled zero there has to exist $h \in F(r)$ such that $h(x) = 0$, hence the zero labeling is obtained by shift and scaling of the n dimensions. Assume $VCdim(F(r)) \geq 2n + 1$, thus all 2n+1 points have to be within the boundaries of the fixed faces of the rectangle. We note that there are only $2n$ extremum values, points can have when considering only n dimensions. Thus, for $2n + 1$ points there has to exist at least one point, $y \in U$, which does not attain the maximum value nor minimum value of any of the n dimensions. For $h \in F(r)$ if $h(y) = 0$ there has to exist another point $x \in U$ for which $h(x) = 0$, thus U cannot be shattered. □

A.2 Chapter 4 Appendix

A.2.1 Finding Equivalence Sets

In this section we present an algorithm for finding all equivalences between sets within a given group of sets $\mathbf{S}$. Each set $\mathbf{s} \in \mathbf{S}$, contains several elements from a finite set $\mathbf{P}$, thus $\mathbf{s} \subseteq \mathbf{P}$. Two sets are said to be equivalent if they are identical in the elements that they contain. The output of the algorithm is a grouping $\mathbf{E_P}$ of all equivalence sets. The elements in each set in the input are assumed to be sorted according to some predefined ordering of P. Given such sorted sets equivalence can be found simply by a sequential pass over all elements in each set, each time comparing the k'th element in all sets of size k or bigger. Sets which are equivalent will be identical in all elements. An efficient implementation has runtime of $O(|\mathbf{P}| + \sum |\mathbf{s}|)$.

A.2.2 Algorithm Analysis Continued

In the following we show that Algorithm 3 indeed finds all possible intersections and represents the partial order correctly maintaining all constraints. For clarity we shall start by defining the following notations. A hidden class is denoted by $\mathbf{C}$. Γ^{in} denotes the set of all classes in the input. $\mathbf{P_C}$ denotes the set of properties of class $\mathbf{C}$. $\Gamma_{\mathbf{C}} \subset \Gamma^{in}$ denotes all classes in the input which share the same properties as $\mathbf{C}$, i.e. $\forall \mathbf{A} \in \Gamma_{\mathbf{C}}$, $\mathbf{P_C} \subset \mathbf{P_A}$, hence every $\mathbf{A} \in \Gamma_{\mathbf{C}}$ is a subclass of the hidden general class $\mathbf{C}$. We shall say that a path between a known class (e.g. $\mathbf{A} \in \Gamma^{in}$) to one of its properties- $p \in \mathbf{P_A}$ is active at the i'th iteration of Algorithm 3 if there exists a path between the known class to a node in $in(p)$.

Lemma 1 For each edge $(\mathbf{u}, \mathbf{v})$ in the output graph $\mathbf{G}^{out} =< \mathbf{V}^{out}, \mathbf{E}^{out} >$ where both $\mathbf{u}, \mathbf{v} \in \mathbf{V}^{out} \backslash \mathbf{V}^{in}$, $\mathbf{u}$ was discovered before $\mathbf{v}$.

This is easy to see as nodes are discovered using intersection of properties and $\mathbf{R_v} \subset \mathbf{R_u}$, thus $\mathbf{u}$ has to be discovered first.

Lemma 2 For a class $\mathbf{C}$ discovered at the i'th iteration of Algorithm 3 there are at least i+1 classes from Γ^{in} that share the properties $\mathbf{P_C}$.

Algorithm 8 Find Equivalence

Input :

P set of possible elements

S group of sets, for each set $i \in \mathbf{S}$, $\mathbf{s} \subseteq \mathbf{P}$

Output:

$\mathbf{E_P}$ A grouping of the sets in **S** into maximal equivalence sets according to identity on elements of **P**.

1. Lists = $|\mathbf{P}|$ empty lists.
2. $\mathbf{E_P}$ = FindEq(**S**,Lists,1);

FindEq(**S**,Lists,ind)

1. init empty list $\mathbf{E_P}$
2. add all $\mathbf{s} \in \mathbf{S}$ for which $|\mathbf{s}| = ind - 1$ to $\mathbf{E_P}$
3. add each **s** with $|\mathbf{s}| \geq ind$ to Lists($\mathbf{s}(ind)$)
4. add each non empty list as a list to TempList, and empty Lists
5. for each list in TempList compute $\mathbf{E_P^k}$ = FindEq(TempList(k),Lists,ind+1)
6. add all $\mathbf{E_P^k}$ to $\mathbf{E_P}$

This lemma is straight forward from the fact that at each iteration at least two nodes in the graph are intersected in the process of discovering a new node.

Lemma 3 When all paths between classes in $\Gamma_{\mathbf{C}}$ to all properties in $\mathbf{P_C}$ are active, either $\mathbf{C}$ will be discovered or a class $\mathbf{B}$ where $\mathbf{P_C} \subset \mathbf{P_B}$ will be discovered.

Proof When all paths are active we know that an intersection between a subset or all classes in $\Gamma_{\mathbf{C}}$ can be computed as all classes share the properties in $\mathbf{P_C}$. From lemma 1, when two (or more) classes share a large set of properties, this intersection is discovered first (lemma 1).

Theorem 1 Given $\mathbf{G}^{in}$ maintaining all constraints (4.1- 4.5), the output graph $\mathbf{G}^{out}$ also maintains all constraints (4.1- 4.5) .

The different operations used during the algorithm are tailored to maintain the constraints- The data consistency constraint (4.1) is maintained by being data driven- only adding nodes representing intersection of existing ones. While replacing each single edge in $\mathbf{G}^{in}$ with a path in $\mathbf{G}^{out}$, thus maintaining all the original connectivity without loosing any information.

The merging operations verify that both types of vertex minimality (4.2 and 4.3) constraints are maintained at each iteration. The fact that each intersection is computed in only one single iteration of the algorithm (lemma 7) maintain the vertex minimality constraints throughout the run of the algorithm.

The max order constraint (4.4) is maintained by discovering nodes according to the partial order (lemma 1) and connecting these nodes to all nodes contributing to their discovery (step 2.f).

The edge minimality constraint (4.5) is maintained by the EdgeMin operation.

Lemma 4 By the i'th iteration of step 2 in algorithm 3 all intersections of $i+1$ classes from Γ^{in} have been discovered.

Proof By induction on the number of iterations. We will start by proving that at the first iteration all intersections of pairs of classes from Γ^{in} are found. At step 2.a. nodes are created for each pair of classes which intersect and for each property in the

intersection. At step 2.b. nodes corresponding to the same pair of classes are merged and the result is list of nodes, FWMERG, which contains nodes corresponding to all pairwise non empty intersections. Each node in FWMERG is kept or unified with another node with the same representation. Thus, all pairwise intersections are discovered and represented in the graph either by a node of their own or by a node representing an intersection of a larger set of classes which share the same properties as an intersection of unified pairs.

Let's assume the Lemma is true for the i-1 iteration.

From lemma 2 we know that all intersections discovered at step greater than step i will contain at least $i+2$ properties. Hence either an intersection of length $i+1$ is discovered prior to or during step i, or its not discovered at all. Let's denote by $\mathbf{C}$ an intersection of $i+1$ classes which was not discovered prior to or during step i. From lemma 2 we can also conclude that no class $\mathbf{B}$ for which $\mathbf{P_C} \subset \mathbf{P_B}$, can be discovered in step i, as it will be have less than $i+1$ classes in its intersections. Putting this together with lemma 3 we can conclude that if all paths between classes in $\Gamma_\mathbf{C}$ to all properties in $\mathbf{P_C}$ are active, $\mathbf{C}$ has to be discovered up until step i. Thus for class $\mathbf{C}$ not to be discovered at least one path in not active. Let's denote $\mathbf{A} \in \Gamma_\mathbf{C}$ as a class in the origin of such a path and denote by $j < i+1$ the latest iteration class $\mathbf{A}$ had an active path to all the features in $\mathbf{P_C}$. Thus, at iteration $j+1$ of the algorithm no intersection between $\mathbf{A}$ to any of the classes in $\Gamma_\mathbf{C}$ was found. Such an intersection contains all features in $\mathbf{P_C}$ thus, if existed, it would have kept all paths from $\mathbf{A}$ to the properties in $\mathbf{P_C}$ alive. From lemma 1 we know that till $\mathbf{C}$ is discovered, only more specific classes can be discovered. From the induction assumption we know that all such specific classes are discovered. Thus, before $\mathbf{C}$ is discovered, all paths between classes in $\Gamma_\mathbf{C}$ end up at nodes in $in(p)$ which are more specific than $\mathbf{C}$. Thus there always exists at least one intersection between them before $\mathbf{C}$ is discovered (the intersection of all classes in $\Gamma_\mathbf{C}$). Thus all paths have to remain active before $\mathbf{C}$ is discovered, which contradicts the assumption that such a class $\mathbf{A}$ with $j < i+1$ exists. □

The following theorem states that all possible intersections among the original group of classes is discovered by Algorithm 3.

Theorem 2 For each subset $\Gamma_{\mathbf{C}} \subset \Gamma^{in}$ such that $\cap_{\mathbf{A}\in\Gamma_{\mathbf{C}}}\mathbf{P_A} \neq \emptyset$ there exists a corresponding node in $\mathbf{v} \in \mathbf{V}^{out}$ with $\mathbf{R_v} = \cap_{\mathbf{A}\in\Gamma_{\mathbf{C}}}\mathbf{P_A}$

The proof is straightforward from lemma 4.

A.2.3 Run Time Analysis

Our proposed algorithm simultaneously produces an ordering over classes while finding all possible intersections of representation sets between all known classes. For each such intersection both its representation and the maximal group of classes intersected are maintained in our graph structure.

For our run time analysis let's denote the input as:

Γ a family of $\mathbf{m}$ classes.

$\mathbf{P}$ the set of all possible $\mathbf{n}$ features, used for the representation of classes in Γ.

$\mathbf{c}_i \in \Gamma$ where $i = 1..\mathbf{m}$, a single class.

$\mathbf{p}_i \subset \mathbf{P}$ where $i = 1..\mathbf{m}$, the representing feature set of each class.

Finding all possible intersections among the classes in Γ while for each intersection maintaining the features in $\mathbf{P}$ and all classes in the intersection which share these features, has a run time which is at least the size of the desired output. Let's denote the group of all possible intersections with a unique set of features as Ω. The desired output is $\{(\hat{\mathbf{c}}_i, \hat{\mathbf{p}}_i)\}_{i=1}^{|\Omega|}$ where:

$\hat{\mathbf{c}}_i \in \Omega$ a group of classes from Γ which have a non empty intersection of their corresponding set of representing features.

$\hat{\mathbf{p}}_i \subset \mathbf{P}$ the features in the intersection of the representation of all classes in $\hat{\mathbf{c}}_i$.

Note, that Ω is defined such that $\forall i \neq j$, $\hat{\mathbf{p}}_i \not\equiv \hat{\mathbf{p}}_j$. Let's denote the description length of $\{(\hat{\mathbf{c}}_i, \hat{\mathbf{p}}_i)\}_{i=1}^{|\Omega|}$ as $\mathbf{Dl}$.

$\mathbf{Dl}$ can be computed by summing over the number of features and classes in all intersections in Ω, thus $\mathbf{Dl} = \sum_{k=1}^{n-1} \mathbf{k}\#_k$. Here k denotes the size of the intersection (number of features), and $\#_k$ denotes the sum of group sizes which have an

intersection with k features, thus $\#_k = \sum_{\hat{\mathbf{c}_i}^k \in \Delta_k} |\hat{\mathbf{c}_i}^k|$. $\Delta_k \subseteq \Omega$ denotes the group of intersections with k features, thus $\forall \hat{\mathbf{c}_i}^k \in \Delta_k$, $|\hat{\mathbf{p}_i}^k| = k$.

Now let's define $\hat{\mathbf{Dl}}$ as $\mathbf{Dl}$ where $\#_k = \sum_{\hat{\mathbf{c}_i}^k \in \Delta_k} |\hat{\mathbf{c}_i}^k|^2$. The difference between $\hat{\mathbf{Dl}}$ and $\mathbf{Dl}$ is that we sum over the square of the number of classes in a single intersection of k features.

We start by analyzing a single iteration of step 2 in the algorithm. Let $\Omega^{i-1} \subseteq \Omega$ denote the set of intersections found during the i'th-1 iteration of step 2. Let $\mathbf{S}^i$ denote the set of all pairs of group of classes in Ω^{i-1} which have a non empty intersection found at the i'th iteration of step 2. Thus, $\forall \hat{\mathbf{c}_k}, \hat{\mathbf{c}_j} \in \Omega^{i-1}, j \neq k$ we define $\mathbf{c}^i_{kj} = \hat{\mathbf{c}_k} \cup \hat{\mathbf{c}_j}$ and its corresponding set of features as $\mathbf{p}^i_{kj} = \hat{\mathbf{p}_k} \cap \hat{\mathbf{p}_j}$. $\mathbf{p}^i_{kj} \neq \emptyset$ iff $\mathbf{c}^i_{kj} \in \mathbf{S}^i$. Note that $\mathbf{S}^i$ is not the same as Ω^i but rather a temporary set computed during the consecutive Split and ForwradMerge operations; this point will be explained in details in the following proof of lemma 4. For simplicity we mark the set of nodes which were intersected in a single round as $\cup_{\mathbf{s}}\mathbf{in}(\mathbf{s})$, ignoring the case where $|\mathbf{in}(\mathbf{s})| = 1$ as such a node won't be intersected and won't affect the runtime analysis.

Lemma 5 The run time of the i'th iteration of step 2 in Alg 3 is $O(\sum_{\mathbf{c}^i_j \in \mathbf{S}^i} |\mathbf{p}^i_j| + \sum_{\mathbf{c}^{i+1}_j \in \mathbf{S}^{i+1}} |\mathbf{p}^{i+1}_j|)$.

Proof This can be seen by following the different phases taken during the i'th iteration of step 2. During the split phase only edges that connect to new graph nodes (nodes representing Ω^{i-1}) created during the i-1 iteration of step 2 undergo a split. Each two edges coupled together during the split represent an intersection between two sets in Ω^{i-1} and share a single feature. For each intersection of two sets in Ω^{i-1} the number of resulting couplings of edges during the split phase is the number of features in the intersection. This is done for every intersecting pair from Ω^{i-1}. Thus the Split phase takes: $\sum_{\mathbf{c}^i_j \in \mathbf{S}^i} |\mathbf{p}^i_j|$.

Following the above analysis of the Forward-Merge one can construct $\mathbf{E}_{in}$ (the group of equivalence sets of nodes according to the incoming edges) and preform the node updates simultaneously in $O(|\cup_{\mathbf{s}} \mathbf{in}(\mathbf{s})| + |\mathbf{U}|)$. Here $\mathbf{U}$ represents the sets of nodes built during the Split phase, hence $|\mathbf{U}| = \sum_{\mathbf{c}^i_j \in \mathbf{S}^i} |\mathbf{p}^i_j|$, and $\cup_{\mathbf{s}}\mathbf{in}(\mathbf{s})$ represents

the set of nodes which were intersected during the Split phase, thus all sets in Ω^{i-1} which have at least one intersection with another set from Ω^{i-1}. It is easy to see that $|\cup_{\mathbf{s}} \mathbf{in}(\mathbf{s})| \leq |\mathbf{S}^i| \leq \sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i|$, so we can conclude that the runtime of each Forward-Merge is $O(\sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i|)$.

Backward-Merge is applied to the output of the Forward-Merge which is $\mathbf{S}^i$. Following the above analysis of the Backward-Merge we can construct $\mathbf{E}_{out}$ (the group of equivalence sets of nodes according to the outgoing edges) and perform the node updates simultaneously in $O(|\cup_{\mathbf{s}-was-split} \mathbf{s}| + \sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i|)$. Here $\cup_{\mathbf{s}-was-split}\mathbf{s}$ represents the sets of nodes which were split, thus $|\cup_{\mathbf{s}-was-split}\mathbf{s}| < \sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i|$. Note that in the run time analysis of Backward-Merge we used the notation $\sum_{\mathbf{u}} |\mathbf{O}ut(u)|$ which is equivalent to $\sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i|$, as $\mathbf{U}$ represents $\mathbf{S}^i$, $\mathbf{u}$ represents $\mathbf{c}_j^i$ and thus $\mathbf{O}ut(u)$ and $\mathbf{p}_j^i$ are the same. We can conclude that the runtime of Backward-Merge is $O(\sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i|)$

The EdgeMin operation is applied to the output of the Backward-Merge which contains at most $|\mathbf{S}^i|$ nodes. For each node $\mathbf{u} \in \mathbf{U}$ (the input set of EdgeMin) there is at least one node in $\mathbf{S}^i$ which has the same features as in the $\mathbf{O}ut(u)$ set. Thus, $\sum_i |\mathbf{O}ut(u^i)| \leq \sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i|$. The size of the list calculated for each feature p marking the nodes represented by this feature is the same as the size of $\mathbf{in}(\mathbf{p})$ at round i+1. Thus $\sum_{\mathbf{p} \in \mathbf{O}ut(u^i)} |L(p)|$ is the number of intersections node $\mathbf{u}^i$ will have in round i+1, and therefore going over all nodes which would be intersected during round i+1 we get that $\sum_i \sum_{\mathbf{p} \in \mathbf{O}ut(u^i)} |L(p)| = \sum_{\mathbf{c}_j^{i+1} \in \mathbf{S}^{i+1}} |\mathbf{p}_j^{i+1}|$ is the runtime of the Split operation during stage i+1. As discussed above, in order to delete efficiently the edges during the EdgeMin operation we need a hash table of size $|\cup_i \mathbf{I}n(u^i)|$. As in the Forward-Merge it is easy to see that $|\cup_i \mathbf{I}n(u^i)| \leq |\mathbf{S}^i|$. Thus, initializing such a table and preforming the delete operation at runtime $O(|\mathbf{I}n(u^i)| + |\mathbf{I}n(u^j)|)$ can be bounded by $|\mathbf{S}^i| \leq \sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i|$. To conclude, the runtime of the EdgeMin operation is $O(\sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i| + \sum_{\mathbf{c}_j^{i+1} \in \mathbf{S}^{i+1}} |\mathbf{p}_j^{i+1}|)$.

Summing up the runtime of all procedures done during a single iteration of step 2, we get that the run time of the i'th iteration is $O(\sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i| + \sum_{\mathbf{c}_j^{i+1} \in \mathbf{S}^{i+1}} |\mathbf{p}_j^{i+1}|)$. □

Lemma 6 Empty intersections are never computed.

Proof This is easy to see as intersections are computed by looking at each features to obtain the classes that have these features. Classes which don't share features will never be considered to have an intersection.

Lemma 7 A non empty intersection between any $\Delta \subseteq \Omega$ is discovered exactly once.

Proof Once an intersection is discovered the paths between all classes in Δ to the properties shared by all these classes are updated and all go through a single node representing Δ. No other path between these classes to the features is kept alive.

Lemma 8 $\forall \hat{\mathbf{c}_j} \in \Omega$ the corresponding $\hat{\mathbf{p}}_j$ can be calculated at most $|\hat{\mathbf{c}_j}|^2$ times.

Proof The lemma follows from the fact that we start by looking at intersections of pairs. In the worst case all subgroups of classes intersected when creating the group $\hat{\mathbf{c}_j}$, contain a single class.

Theorem 3 Alg 3 has runtime O($\hat{\mathbf{Dl}}$).

Proof From lemma 8 and lemma 5 we can bound the runtime of a single iteration of step 2 by $\sum_{\mathbf{c}_j^i \in \mathbf{S}^i} |\mathbf{p}_j^i| < \sum_{\hat{\mathbf{c}_j} \in \Omega^i} |\hat{\mathbf{c}_j}|^2 |\hat{\mathbf{p}_j}|$, as $|\hat{\mathbf{c}_j}|^2$ is the upper bound on the number of times $|\hat{\mathbf{p}_j}|$ is summed in the left part of the equation. From theorem 2 we know that all intersections between classes in Γ are found. From lemma 7 we get that each intersection of a group of classes is calculated exactly once, and no empty intersections are calculated. Thus the intersections calculated during all the iterations of step 2 are exactly the intersections represented in Ω and by switching from summation over iterations to summation accourding to $|\hat{\mathbf{p}_j}|$ we get, $\sum_i \sum_{\hat{\mathbf{c}_j} \in \Omega^i} |\hat{\mathbf{c}_j}|^2 |\hat{\mathbf{p}_j}| = \sum_{k=1}^{n-1} \mathbf{k} \sum_{\hat{\mathbf{c}_i}^k \in \Delta_k} |\hat{\mathbf{c}_i}^k|^2 = \hat{\mathbf{Dl}}$. It follows that the runtime of Step 2 of the algorithm is $O(\hat{\mathbf{Dl}})$. In Step 3 we perform a single pass over all nodes in the graph which takes $|\Omega|$ steps. As $|\Omega| < |\hat{\mathbf{Dl}}|$ we can conclude that the runtime of the algorithm is dominated by the runtime of Step 2 $O(\hat{\mathbf{Dl}})$ $\square$.