Transfer Learning Algorithms for Image Classification

by

Ariadna Quattoni

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author . . . . . .
Department of Electrical Engineering and Computer Science
May 22, 2009

Certified by . . . . . .
Michael Collins
Associate Professor
Thesis Supervisor

Certified by . . . . . .
Trevor Darrell
Associate Professor
Thesis Supervisor

Accepted by . . . . . .
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Transfer Learning Algorithms for Image Classification

by

Ariadna Quattoni

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

An ideal image classifier should be able to exploit complex high dimensional feature representations even when only a few labeled examples are available for training. To achieve this goal we develop transfer learning algorithms that: 1) Leverage unlabeled data annotated with meta-data and 2) Exploit labeled data from related categories.

In the first part of this thesis we show how to use the structure learning framework (Ando and Zhang, 2005) to learn efficient image representations from unlabeled images annotated with meta-data.

In the second part we present a joint sparsity transfer algorithm for image classification. Our algorithm is based on the observation that related categories might be learnable using only a small subset of shared relevant features. To find these features we propose to train classifiers jointly with a shared regularization penalty that minimizes the total number of features involved in the approximation.

To solve the joint sparse approximation problem we develop an optimization algorithm whose time and memory complexity is $O(n \log n)$ with n being the number of parameters of the joint model.

We conduct experiments on news-topic and keyword prediction image classification tasks. We test our method in two settings: a transfer learning and multitask learning setting and show that in both cases leveraging knowledge from related categories can improve performance when training data per category is scarce. Furthermore, our results demonstrate that our model can successfully recover jointly sparse solutions.

Thesis Supervisor: Michael Collins
Title: Associate Professor

Thesis Supervisor: Trevor Darrell
Title: Associate Professor

Acknowledgments

I would like to thank my two advisors: Professor Michael Collins and Professor Trevor Darrell for their advice. They have both given me a great deal of independence in pursuing my ideas and contributed towards their development. My gratitude also goes to my thesis reader Professor Antonio Torralba.

I would also like to thank my dearest friend Paul Nemirovsky for introducing me to computer science and giving me constant encouragement throughout my years at MIT.

Finally, a very special thanks goes to Xavier Carreras who has contributed to the ideas of this thesis and who has given me unconditional support.

Contents

Chapter 1

Introduction

An ideal image classifier should be able to exploit complex high dimensional feature representations even when only a few labeled examples are available for training. Learning with small training sets is important because there are many real world applications where only a few labeled examples might be available. For example, when building a system that classifies images in a personal photo collection based on user defined categories, an appealing application would ask the user to label a handful of images only.

As the photo collection application illustrates, there is a pressing need for "sample efficient" learning algorithms. The photo collection application also illustrates another point: we might only have a few labeled examples for a given task, but plenty of labeled data for hundreds of related tasks might be available. Ideally, an image classifier should be able to exploit all of these resources, just as humans can exploit previous experience when learning some concepts.

For example, when a child learns to recognize a new letter of the alphabet he will use examples provided by people with different hand-writing styles using pens of different colors and thicknesses. Without any prior knowledge a child would need to consider a large set of features as potentially relevant for learning the new concept, so we would expect the child to need a large number of examples. But if the child has previously learnt to recognize other letters, he can probably discern the relevant attributes (e.g. number of lines, line curvatures) from the irrelevant ones (e.g. the color of the lines) and learn the new concept with a few examples. This observation suggests that a "sample efficient" image

classification algorithm might need to exploit knowledge gained from related tasks.

Otherwise, in the absence of prior knowledge a typical image classification task would require the exploration of a large and complex feature space. In such high dimensional spaces some features will be discriminative but most probably a large number of them will be irrelevant. If we knew what the irrelevant features were we might be able to learn a concept from fewer examples. While we do not know a-priori what features are irrelevant, related tasks might share irrelevant features and we can use training data from these tasks to discover them.

The goal of this thesis is to develop efficient transfer learning algorithms for image classification that can exploit rich feature representations. Our emphasis is on minimizing the amount of supervised training data necessary to train such classifiers by discovering shared representations among tasks.

1.1 Problem: Transfer Learning for Image Classification

1.1.1 Image Classification

In an image classification task our goal is to learn a mapping from images x to class labels y. In general we will be considering binary classification tasks, and thus $y = \{+1, -1\}$. For example, in the news domain setting we might define a class for every news story. In this case a task consists of predicting whether an image belongs to a particular story or not. Similarly, we can consider a caption word to be a class and predict whether a given word is a proper annotation for an image.

Because our goal is to develop an approach that can handle a wide range of image classification tasks, we work with general, rich feature representations (i.e. not tailored or engineered for a particular problem) and delegate the task of finding useful features to the training algorithm. In other words, our philosophy is to use a flexible and general feature extraction process which generates a large number of features and let the training algorithm discover what features are important in a given domain.

1.1.2 Transfer Learning

We now introduce some notation that will be useful in setting the transfer learning problem. We assume that we have a collection of tasks and a set of supervised training samples for each of them. Our data is of the form: $\mathcal{D} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$ where $\mathcal{T}_k = \{(\mathbf{x}_1^k, y_1^k), (\mathbf{x}_2^k, y_2^k), \ldots, (\mathbf{x}_{n_k}^k, y_{n_k}^k)\}$ is the training set for task k. Each supervised training sample consists of some input point $\mathbf{x} \in \mathbb{R}^d$ and its corresponding label $y \in \{+1, -1\}$. We will usually refer to the individual dimensions of $\mathbf{x}$ as features.

Broadly speaking a transfer algorithm will exploit commonality among tasks to learn better classifiers. The transfer algorithm achieves this by training classifiers on $\mathcal{D}$ jointly; the meaning of joint training depends on the transfer learning framework.

In a *feature sharing* framework (Evgeniou and Pontil, 2004; Jacob et al., 2008; Jebara, 2004; Obozinski et al., 2006; Torralba et al., 2006) the relatedness assumption is that tasks share relevant features. In a *learning hidden representations* framework (Thrun, 1996; Ando and Zhang, 2005; Argyriou et al., 2006; Amit et al., 2007) the relatedness assumption is that there exists a mapping from the original input space to an underlying shared feature representation. In a *hierarchical bayesian learning* framework (Bakker and Heskes, 2003; Raina et al., 2006; Fei-Fei et al., 2006) tasks are assumed to be related by means of sharing a common prior distribution over classifiers' parameters

The work that we present in chapter 4 is an instance of *learning hidden representations* and the transfer algorithm presented in chapters 5 and 6 is an instance of *feature sharing*.

We consider two transfer learning settings which we call *symmetric transfer* and *asymmetric transfer* (Thrun, 1996). In a *symmetric transfer* setting there are a few training samples for each task and the goal is to share information across tasks in order to improve the average performance over all tasks. We also refer to this setting as multitask learning.

In contrast, in an *asymmetric transfer* setting there is a set of tasks for which a large amount of supervised training data is available, we call these tasks *auxiliary* (Thrun, 1996). The goal is to use training data from the *auxiliary* tasks to improve the classification performance of a *target* task T_0 (Thrun, 1996) for which training data is scarce.

It has been shown (Ando and Zhang, 2005) that *asymmetric transfer* algorithms can be

used for semi-supervised learning by automatically deriving *auxiliary* training sets from unlabeled data. The *auxiliary* tasks are designed so that they can be useful in uncovering important latent structures.

In chapter 4 we will show a semi-supervised application where we learn an image representation using unlabeled images annotated with some form of meta-data that is used to derive *auxiliary* tasks.

1.2 Thesis Contributions

In this thesis we study efficient transfer algorithms for image classification. These algorithms can exploit rich (i.e. high dimensional) feature spaces and are designed to minimize the amount of supervised training data necessary for learning.

In the first part of this work we apply the *structure learning* framework of Ando and Zhang (2005) to a semi-supervised image classification setting. In the second part we present a transfer learning model for image classification based on joint regularization.

- Learning Image Representations using Unlabeled Data annotated with Meta-Data:
  Current methods for learning visual categories work well when a large amount of labeled data is available, but can run into severe difficulties when the number of labeled examples is small. When labeled data is scarce it may be beneficial to use unlabeled data to learn an image representation that is low-dimensional, but nevertheless captures the information required to discriminate between image categories.

  We present a semi-supervised image classification application of *asymmetric transfer* where the *auxiliary* tasks are derived automatically using unlabeled data annotated with meta-data.

  In particular, we consider a setting where we have thousands of images with associated captions, but few images annotated with story labels. We take the prediction of content words from the captions to be our *auxiliary* tasks and the prediction of a story label to be our *target* task.

Our objective is to use the *auxiliary* tasks to learn a lower dimensional representation that still captures the relevant information necessary to discriminate between different stories. To this end we applied the *structure learning* framework of Ando and Zhang. This framework is based on using data from the *auxiliary* tasks to learn a subspace of the original input space that is discriminative for all *auxiliary* tasks. Projecting into this space will give us a new image representation that will be useful for learning *target* classifiers.

Our results indicate that the representations learned via *structure learning* on the *auxiliary* tasks can improve the performance of the *target* story classifiers. In particular, we show that the induced representation can be used to minimize the amount of supervised *target* training data necessary for learning accurate image classifiers.

Our experiments show that our method significantly outperforms a fully-supervised baseline model and a model that ignores the captions and learns a visual representation by performing PCA on the unlabeled images alone.

In brief, we show that when meta-data labels are suitably related to a *target* task, the *structure learning* learning method can discover feature groupings that speed learning of the *target* task. Our current work concentrated on captions as the source of meta-data, but more generally other types of meta-data could be used.

- A Transfer Algorithm based on $l_{1,\infty}$ Joint Regularization:
  In the second part of this thesis we developed a joint regularization transfer algorithm for image classification. Our algorithm is based on the observation that related tasks might be learnable using only a small subset of shared relevant features. To find these features we propose to train tasks jointly using a shared regularization penalty. The shared regularization penalty is used in our model to induce solutions where only a few shared features are used by any of the classifiers.

  Previous approaches to joint sparse approximation (Obozinski et al., 2006; Torralba et al., 2006) have relied on greedy coordinate descent methods. In contrast, our objective function can be expressed as a linear program and thus a globally optimal solution can be found with an off-the-shelf package.

We test our algorithm in an image classification *asymmetric transfer* learning problem. The classification problem consists of predicting topic labels for images. We assume that we have enough training data for a set of *auxiliary* topics but less data for a *target* topic. Using data from the *auxiliary* topics we select a set of discriminative features that we utilize to train a classifier for the *target* topic.

- An Efficient Training Algorithm for $l_{1,\infty}$ regularization:
  While the training algorithm for joint regularization described in chapter 5 is feasible for small problems, it becomes impractical for high dimensional feature spaces. In chapter 6 we address this problem and develop a time and memory efficient general optimization algorithm for training $l_{1,\infty}$ regularized models.

  The algorithm has $O(n \log n)$ time and memory complexity with n being the number of parameters of the joint model. This cost is comparable to the cost of training m independent sparse classifiers. Thus our work provides a tool that makes implementing a joint sparsity regularization penalty as easy and almost as efficient as implementing the standard l_1 and l_2 penalties.

  For our final set of experiments we consider a *symmetric transfer* learning setting where each task consists of predicting a keyword for an image. Our results show that $l_{1,\infty}$ regularization leads to better performance than both l_1 and l_2 regularization and that it is effective in discovering jointly sparse solutions in high dimensional spaces.

1.3 Outline of the thesis

The structure of the thesis is as follows: chapter 2 provides the necessary background on linear prediction models and regularization. Chapter 3 reviews existing work on general transfer algorithms and transfer algorithms for image classification. Chapter 4 describes our work on learning image representations from unlabeled data annotated with meta-data . Chapter 5 presents our joint regularization transfer algorithm based on $l_{1,\infty}$ regularization. Chapter 6 develops a more general efficient training algorithm for $l_{1,\infty}$ regularization. Finally, in chapter 7 we draw conclusions and discuss future lines of research.

Chapter 2

Background

In this chapter we provide the general machine learning background necessary to understand the remaining chapters. Section 2.3 describes the supervised learning setting that will be assumed throughout the thesis and the empirical risk minimization framework. This section shows that to ensure that a classifier will generalize to unseen samples one must control the complexity of the class of functions explored by the training algorithm.

Section 2.3 describes the setting of linear classification together with a regularization framework designed to control the complexity of a function class. The joint regularization model that we present in chapter 5 is an instantiation of this regularization framework to the transfer learning setting.

Finally, in Section 2.4 we describe a general optimization framework for training regularized models. While multiple algorithms have been developed for this purpose we focus our attention on primal projected gradient methods. The optimization algorithm for joint regularization presented in chapter 6 follows this approach.

2.1 Notation

We will use uppercase letters to denote matrices and lowercase bold letters to denote vectors, for example we will write $\mathbf{a}_k$ for the k-th column of matrix A and $\mathbf{a}^k$ for the k-th row. Sets will be represented using uppercase italic letters.

We write $||\mathbf{w}||_p$ to indicate the p norm of $\mathbf{w} \in \mathbb{R}^d$, that is: $||\mathbf{w}||_p = (\sum_{i=1}^{d} |w_i^p|)^{\frac{1}{p}}$, for

example $||\mathbf{w}||_2$ is used to denote the Euclidean norm and $||\mathbf{w}||_1$ is used to denote the l_1 norm.

We write $\mathsf{Loss}(f(\mathbf{x}), y)$ for a loss function and when useful we adopt the notation $f^{\mathbf{w}}(\mathbf{x})$ to indicate that function f is parameterized by $\mathbf{w}$.

2.2 Empirical Risk Minimization

In a supervised learning framework the goal is to learn a mapping $h : \mathcal{X} \rightarrow \mathcal{Y}$ from some input space $\mathcal{X}$ to labels $\mathcal{Y}$. In this chapter we consider binary classification tasks, i.e. $\mathcal{Y} = \{+1, -1\}$. To learn this mapping, the training algorithm is given a set of n pairs $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ where $\mathbf{x} \in \mathbb{R}^d$ is some training pattern and $y \in \mathcal{Y}$ is its corresponding label. It is common to assume that each labeled example is drawn independently at random from some fixed probability distribution $p(\mathbf{x}, y)$.

Consider learning a function from a fixed hypothesis class $\mathcal{H}$. For example $\mathcal{H}$ could be the class of linear prediction models of the form $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$, for some parameter vector $\mathbf{w} \in \mathbb{R}^d$. A learning algorithm takes a training set $\mathcal{D}$ as input and finds a hypothesis $h \in \mathcal{H}$ that will have small error on future samples drawn from $p(\mathbf{x}, y)$.

More formally, assume that we have a loss function $\mathsf{Loss}(h(\mathbf{x}), y)$ that returns the cost of making prediction $h(\mathbf{x})$ when the true label for the sample is y. These are some examples of popular classification loss functions:

- The zero-one loss : $\mathsf{Loss}_{0-1}(h(\mathbf{x}), y) = 1$ if $sign(h(\mathbf{x})) \neq y$ and 0 otherwise
- The hinge loss : $\mathsf{Loss}_{hinge}(h(\mathbf{x}), y) = max(0, 1 - h(\mathbf{x})y)$
- The exponential loss : $\mathsf{Loss}_{exp}(h(\mathbf{x}), y) = exp(-h(\mathbf{x})y)$
- The logistic loss: $\mathsf{Loss}_{log}(h(\mathbf{x}), y) = ln(1 + exp(-h(\mathbf{x})y))$

We now define the expected risk of a hypothesis h to be:

$$R(h) = \mathbf{E}[\mathsf{Loss}(h(\mathbf{x}), y)] \tag{2.1}$$

where $\mathbf{E}$ denotes the expectation over random variables $(\mathbf{x}, y)$ drawn from the distribution $p(\mathbf{x}, y)$. Ideally, we would like to find the hypothesis with minimum expected risk:

$$h^* = \text{argmin}_{h \in \mathcal{H}} \mathbf{E}[\mathsf{Loss}(h(\mathbf{x}), y)] \tag{2.2}$$

Since $p(\mathbf{x}, y)$ is unknown the expected risk can not be minimized directly, instead we will find a hypothesis that minimizes the empirical risk on $\mathcal{D}$:

$$\hat{h} = \text{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \mathsf{Loss}(h(\mathbf{x}_i), y_i) \tag{2.3}$$

For this strategy to work we need to make sure that given a large enough training set the empirical risk will be a good estimate of the expected risk. Consider a $[0, 1]$ bounded real value loss function $\mathsf{Loss}(h(\mathbf{x}), y)$, this is a random variable with expected value $R(h) = \mathbf{E}[\mathsf{Loss}(h(\mathbf{x}), y)]$. Since it is a bounded random variable, we can use Hoeffding 's inequality to bound the deviation of its empirical mean from its expected value. By doing that we obtain the following result (e.g. see Anthony and Bartlett (1999)):

$$\forall h \in \mathcal{H} \quad P(|\frac{1}{n} \sum_{i=1}^{n} \mathsf{Loss}(h(\mathbf{x}_i), y_i) - R(h)| \geq \epsilon) \leq 2 \exp(-2\epsilon^2 n) \tag{2.4}$$

Thus the probability that the empirical loss of a function h is ϵ greater than its expected loss decays exponentially with the size of the training set. Recall that the expected risk is the expected loss of the best hypothesis in $\mathcal{H}$. Therefore, to guarantee convergence to the expected risk it suffices to ensure that there is no hypothesis $h \in \mathcal{H}$ such that the difference between its empirical loss and its expected loss is greater than ϵ. That is, we must bound the following:

$$P(\max_{h \in \mathcal{H}} |\frac{1}{n} \sum_{i=1}^{n} \mathsf{Loss}(h(\mathbf{x}_i), y_i) - R(h)| \geq \epsilon) \tag{2.5}$$

$$= P(\bigcup_{h \in \mathcal{H}} \{|\frac{1}{n} \sum_{i=1}^{n} \mathsf{Loss}(h(\mathbf{x})i), y_i) - R(h)| \geq \epsilon\}) \tag{2.6}$$

By combining (2.4) with a union bound we can state that for a finite hypothesis class $\mathcal{H}$ (e.g. see (Anthony and Bartlett, 1999)):

$$P(\max_{h\in\mathcal{H}}|\frac{1}{n}\sum_{i=1}^{n}\mathsf{Loss}(h(\mathbf{x}_i),y_i)-R(h)|>\epsilon)\leq 2|\mathcal{H}|exp(-2\epsilon^2 n) \tag{2.7}$$

Notice that the complexity of a hypothesis class (i.e. $|\mathcal{H}|$) is closely related to the speed of convergence of the empirical risk to the expected risk. The difference between the empirical risk and the expected risk is usually referred as estimation or generalization error. Similar results linking the complexity of a hypothesis class $\mathcal{H}$ to the generalization error can be derived for infinite hypothesis classes using combinatorial measures of complexity. In particular, for the class of linear classifiers the generalization error can be bounded in terms of the norm of the parameter vector $\mathbf{w}$ (Zhang, 2002).

The bound (2.7) suggests that to obtain a small generalization error it is best to search over a small hypothesis class. However, to ensure that there is an $h \in \mathcal{H}$ with small expected risk it is best to consider a large hypothesis class, i.e. we want the approximation error to be small. The optimal complexity of $\mathcal{H}$ should be chosen to balance the tradeoff between approximation and generalization error. In the next section we describe a regularization technique designed with this goal in mind.

2.3 Linear Classifiers and Regularization

In this section we consider linear classifiers of the form $h(\mathbf{x}) = \mathbf{w}\cdot\phi(\mathbf{x})$; for some $\mathbf{w}\in\mathbb{R}^d$. This type of classifier is linear on any arbitrary function of the input $\phi(\mathbf{x})$ but for notational simplicity we will refer to $\phi(\mathbf{x})$ as $\mathbf{x}$.

In a regularization framework the tradeoff between approximation and generalization error is achieved by introducing a complexity penalty. More precisely, one minimizes a penalized version of the empirical risk:

$$\min_{h\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^{n}\mathsf{Loss}(\mathbf{w}\cdot\mathbf{x}_i,y_i)+\gamma\Phi(\mathbf{w}) \tag{2.8}$$

Notice that this formulation is the same as (2.3) except that we have added a regularization penalty term $\Phi(\mathbf{w})$. Generally speaking, this term measures the complexity of a function h and the constant γ in (2.8) controls the tradeoff between generalization and

approximation error (Kearns and Vazirani, 1994).

In the case of linear classification the two most commonly used penalty terms are:

- $\Phi_{l_2}(\mathbf{w}) = \sum_{j=1}^{d} w_j^2$
- $\Phi_{l_1}(\mathbf{w}) = \sum_{j=1}^{d} |w_j|$

To give a concrete example, notice that by combining an l_2 penalty with a hinge loss we get the well known SVM primal objective (Vapnik, 1995):

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \mathsf{Loss}_{hinge}(\mathbf{w} \cdot \mathbf{x}_i, y_i) + \gamma \sum_{j=1}^{d} w_j^2 \tag{2.9}$$

When the regularization penalty can be expressed as a convex constraint on the parameters $\mathbf{w}$, the regularization problem can be formulated as a constrained convex optimization:

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \mathsf{Loss}(\mathbf{w} \cdot \mathbf{x}_i, y_i) : \mathbf{w} \in \Omega \tag{2.10}$$

where Ω is some convex set. For example, for the SVM objective we would get the corresponding convex constrained formulation:

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \mathsf{Loss}(\mathbf{w} \cdot \mathbf{x}_i, y_i) : \sum_{j=1}^{d} w_j^2 \leq C \tag{2.11}$$

Similarly an l_1 regularized problem can be expressed as:

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \mathsf{Loss}(\mathbf{w} \cdot \mathbf{x}_i, y_i) : \sum_{j=1}^{d} |w_j| \leq C \tag{2.12}$$

Here C plays a role analogous to that of γ in the previous formulation in that it controls the tradeoff between approximation and generalization error. Both l_2 and l_1 regularized linear models have been widely studied by the learning theory community (Kakade et al., 2008). The main results show that it is possible to derive generalization bounds for these models where the generalization error can be expressed as a function of C.

So far we have refrained from making any computational considerations, that is we have not considered how would a training algorithm optimize (2.8) and (2.10). Section 2.4 addresses this by describing a general optimization method for (2.10).

2.3.1 Finding Sparse Solutions

Until now we have reviewed structural risk minimization formulations based on convex regularization penalties. However, when the goal is to obtain sparse solutions (i.e. solutions with few non-zero parameters) a natural penalty to consider is the l_0 penalty given by:

$$||\mathbf{w}||_0 = |\{j =: w_j \neq 0\}| \tag{2.13}$$

While this penalty seems to be the most natural choice to induce sparsity, it has the drawback that finding an exact solution becomes a hard combinatorial problem. In general, solving for a given l_0 norm is known to be an NP hard problem (Donoho, 2004). To address this computational challenge there exists two main sparse approximation approaches:

- **Greedy Schemes**: This approach is based on iterative optimization algorithms that add one feature at the time. At iteration t the algorithm adds to the model the feature that most reduces some form of residual loss. Broadly speaking, the residual loss at iteration t is what can not be predicted from the model with $t-1$ features. [1]

- **Convex Relaxation**: These approaches use the convex l_1 penalty to approximate the l_0 penalty. It has been shown that under certain conditions the l_1 convex relaxation is equivalent to the l_0 penalty (Donoho, 2004). [2]

While under certain conditions approximation guarantees have been proven for some greedy algorithms (Tropp, 2004), proving such guarantees is in general not a trivial task. In contrast, for the l_1 convex relaxation provably optimal solutions can be be easily obtained by standard polynomial-time optimization algorithms (Tropp, 2006b).

In other words, by using a convex formulation of the the sparse approximation problem we can take advantage of the vast amount of results in convex optimization theory. In chapter 5 we will follow this approach and propose a transfer algorithm based on a convex

[1] Multiple greedy schemes have been proposed by various communities with different names, statistics: forward stagewise regression, approximation theory: greedy algorithms, learning theory: boosting methods, signal processing: projection pursuit methods

[2] By equivalent we mean that the optimal solution to the l_1 regularized problem will be a solution with minimum l_0 penalty.

relaxation of a *joint sparsity* inducing regularization penalty. Using standard methods in convex optimization we develop in chapter 6 an efficient training algorithm that can scale to large number of examples and dimensions.

2.4 Optimization: Projected SubGradient Methods

There are multiple primal and dual methods for optimizing a convex function subject to convex constraints. In this section we focus on primal projected subgradient methods because they are the most relevant to this work. A subgradient of a function f at a point x is any vector g satisfying: $f(y) \geq f(x) + g^T(y - x)$ for all y.

Projected subgradient methods have been recently revived in the machine learning community for solving classification and regression problems involving large number of features. For some of these large scale problems the alternative dual interior point methods impose computational and memory demands that makes them unpractical. Furthermore, while the convergence rates of gradient based methods are known to be relatively slow, in practice the approximate solutions obtained after a few iterations are good enough for most classification and regression applications (Shalev-Shwartz and Srebro, 2008).

Broadly speaking, projected subgradient methods iterate between performing unconstrained subgradient updates followed by projections to the convex feasible set. For example in the case of l_2 regularization (2.11), the convex set would be an l_2 ball of radius C. Similarly, for l_1 regularization (2.12), the convex set would be an l_1 ball of radius C.

More formally, a projected subgradient method is an algorithm for minimizing a convex function $f(\mathbf{w})$ subject to convex constraints of the form $\mathbf{w} \in \Omega$, where Ω is a convex set (Bertsekas, 1999). For the regularization problems described in the previous section; $f(\mathbf{w})$ is some convex loss function, $\mathbf{w}$ is a parameter vector and Ω is the set of all vectors satisfying a convex constraint penalty, i.e. $r(\mathbf{w}) \leq C$.

A projected subgradient algorithm works by generating a sequence of solutions $\mathbf{w}^t$ via $\mathbf{w}^{t+1} = \mathrm{P}_\Omega(\mathbf{w}^t - \eta_t \nabla f(\mathbf{w}^t))$. Here $\nabla f(\mathbf{w}^t)$ is a subgradient of f at $\mathbf{w}^t$ and $\mathrm{P}_\Omega(\mathbf{w})$ is the

Euclidean projection of $\mathbf{w}$ onto Ω, given by:

$$\min_{\mathbf{w}' \in \Omega} ||\mathbf{w}' - \mathbf{w}||^2 = \min_{\mathbf{w}' \in \Omega} \sum_j (w'_j - w_j)^2 \tag{2.14}$$

Finally, η_t is the learning rate at iteration t that controls the amount by which the solution changes at each iteration. The following theorem shows the convergence properties of the projected subgradient method (Boyd and Mutapcic, 2007).

Define $\mathbf{w}^* = \text{argmin}_{\mathbf{w} \in \Omega} f(\mathbf{w})$, and $\mathbf{w}^t_{\text{best}} = \text{argmin}_{\mathbf{w}^i, 1 \leq i \leq t} f(\mathbf{w}^i)$, where $\mathbf{w}^i$ are the parameter vectors found by the projected subgradient method after each iteration.

Lemma 1 *Assume a convex set Ω and constants R and G such that:*

$$||\mathbf{w}^1 - \mathbf{w}^*||_2 \leq R \tag{2.15}$$

and that for all $\mathbf{w} \in \Omega$:

$$||\nabla f(\mathbf{w})||_2 \leq G \tag{2.16}$$

Then $\mathbf{w}^t_{\text{best}}$ satisfies:

$$f(\mathbf{w}^t_{\text{best}}) - f(\mathbf{w}^*) \leq \frac{R^2 + G^2 \sum_{i=1}^t \eta_i^2}{2 \sum_{i=1}^t \eta_i} \tag{2.17}$$

Proof:

Let $z^{t+1} = \mathbf{w}^t - \eta_t \nabla f(\mathbf{w}^t)$ be the subgradient update before the projection step. We have that:

$$\begin{aligned} ||z^{t+1} - \mathbf{w}^*||_2^2 &= ||\mathbf{w}^t - \eta_t \nabla f(\mathbf{w}^t) - \mathbf{w}^*||_2^2 \\ &= ||\mathbf{w}^t - \mathbf{w}^*||_2^2 - 2\eta_t \nabla f(\mathbf{w}^t)^T \cdot (\mathbf{w}^t - \mathbf{w}^*) + \eta_t^2 ||\nabla f(\mathbf{w}^t)||_2^2 \\ &\leq ||\mathbf{w}^t - \mathbf{w}^*||_2^2 - 2\eta_t (f(\mathbf{w}^t) - f(\mathbf{w}^*)) + \eta_t^2 ||\nabla f(\mathbf{w}^t)||_2^2 \end{aligned}$$

where the last step follows directly from the definition of the subgradient which gives us: $\nabla f(\mathbf{w}^t)^T \cdot (\mathbf{w}^t - \mathbf{w}^*) \geq f(\mathbf{w}^t) - f(\mathbf{w}^*)$

Notice that the projection $\text{P}_\Omega(\mathbf{w})$ can only move us closer to an optimal point $\mathbf{w}^*$:

$$||\mathbf{w}^{t+1} - \mathbf{w}^*||_2 = ||\text{P}_\Omega(z^{t+1}) - \mathbf{w}^*||_2 \leq ||z^{t+1} - \mathbf{w}^*||_2 \tag{2.18}$$

Combining (2.18) with (2.18) we obtain:

$$||\mathbf{w}^{t+1} - \mathbf{w}^*||_2^2 \leq ||\mathbf{w}^t - \mathbf{w}^*||_2^2 - 2\eta_t(f(\mathbf{w}^t) - f(\mathbf{w}^*)) + \eta_t^2||\nabla f(\mathbf{w}^t)||_2^2 \tag{2.19}$$

Applying the above inequality recursively we get that:

$$||\mathbf{w}^{t+1} - \mathbf{w}^*||_2^2 \leq ||\mathbf{w}^1 - \mathbf{w}^*||_2^2 - 2\sum_{i=1}^{t} \eta_t(f(\mathbf{w}^i) - f(\mathbf{w}^*)) + \sum_{i=1}^{t} \eta_t^2||\nabla f(\mathbf{w}^t)||_2^2 \tag{2.20}$$

and therefore:

$$2\sum_{i=1}^{t} \eta_t(f(\mathbf{w}^i) - f(\mathbf{w}^*)) \leq R^2 + \sum_{i=1}^{t} \eta_t^2||\nabla f(\mathbf{w}^t)||_2^2 \tag{2.21}$$

Combining this with:

$$\sum_{i=1}^{t} \eta_i(f(\mathbf{w}^i) - f(\mathbf{w}^*)) \geq (\sum_{i=1}^{t} \eta_i) \min_i((f(\mathbf{w}^i) - f(\mathbf{w}^*))) = (\sum_{i=1}^{t} \eta_i)(f(\mathbf{w}^t_{\text{best}}) - f(\mathbf{w}^*)) \tag{2.22}$$

we get 2.17. □

If we set the learning rate to $\eta_i = \frac{1}{\sqrt{i}}$ the above theorem gives us:

$$f(\mathbf{w}^t_{best}) - f(\mathbf{w}^*) \leq \frac{R^2 + G^2 \sum_{i=1}^{t} \frac{1}{i}}{2\sum_{i=1}^{t} \frac{1}{\sqrt{i}}} \leq \frac{R^2 + G^2 \log(t+1)}{2\sqrt{t}} \tag{2.23}$$

The last inequality follows from two facts. First, we use the upper bound $\sum_{i=1}^{t} \frac{1}{i} \leq \log(t+1)$, which we prove by induction. For $t = 1$ it follows trivially. For the inductive step, for $t > 1$, we need to show that $\log(t) + \frac{1}{t} \leq \log(t+1)$, which is easy to see by exponentiating both sides. Second, we use the lower bound $\sum_{i=1}^{t} \frac{1}{\sqrt{i}} \geq \sqrt{t}$, which we also prove by induction. For $t = 1$ it follows trivially. For the inductive step we want to show that $\sqrt{t-1} + \frac{1}{\sqrt{t}} \geq \sqrt{t}$. Squaring each side and rearranging terms we get $\frac{4(t-1)}{t} + \frac{1}{t^2} + \frac{4\sqrt{t-1}}{t\sqrt{t-1}} \geq 1$, which is true because $\frac{4t-1}{t} \geq 1$ and the $\frac{1}{t^2} + \frac{4\sqrt{t-1}}{t\sqrt{t-1}} \geq 0$.

Consider the computational cost of each iteration of the projected subgradient method. Each iteration involves two steps: In the first step we must compute the gradient of the objective function $\nabla f(\mathbf{w}^t)$. For the hinge loss this can be done in $O(nd)$ time and memory where n is the total number of examples and d is the dimensionality of $\mathbf{w}$. In the second

step we must compute the projection to the convex set: $\mathrm{P}_{\Omega}(\mathbf{w})$. Continuing with the SVM example, for the l_2 penalty computing the projection is trivial to implement and can be done in $O(d)$ time and memory.

The projected subgradient method has been applied to multiple regularized classification problems. Shalev-Shwartz et al. (2007) developed an online projected gradient method for l_2 regularization. Duchi et al. (2008) proposed an analogous algorithm for l_1 showing that computing projections to the l_1 ball can be done in $O(d \log d)$ time.

The results in (Shalev-Shwartz et al., 2007; Duchi et al., 2008) show that for large scale optimization problems involving l_1 and l_2 regularization, projected gradient methods can be significantly faster than state-of-the-art interior point solvers.

Chapter 3

Previous Work

In this chapter we review related work on general transfer learning algorithms as well as previous literature on transfer learning algorithms for image classification. The chapter is organized as follows: section 3.1 provides the necessary notation used throughout the chapter, section 3.2 gives a high level overview of the three main lines of research in transfer learning: *learning hidden representations*, *feature sharing* and *hierarchical bayesian learning*. Each of these lines of work is described in more detail in sections 3.3, 3.4 and 3.5 respectively. Finally, section 3.6 highlights the main contributions of this thesis in the context of the previous literature.

3.1 Notation

In transfer learning, we assume that we have a multitask collection of m tasks and a set of supervised training samples for each of them: $\mathcal{D} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$, where $\mathcal{T}_k = \{(\mathbf{x}_1^k, y_1^k), (\mathbf{x}_2^k, y_2^k), \ldots, (\mathbf{x}_{n_k}^k, y_{n_k}^k)\}$ is the training set for task k. Each supervised training sample consists of some input point $\mathbf{x} \in \mathbb{R}^d$ and its corresponding label $y \in \{+1, -1\}$, we will usually refer to the dimensions of $\mathbf{x}$ as features.

In a *symmetric transfer* setting there are a few training samples for each task and the goal is to share information across tasks to improve the average performance. We also refer to this setting as multitask learning. In *asymmetric transfer* there is a set of tasks for which a large amount of supervised training data is available, we call these tasks *auxil-*

iary. The goal is to use training data from the *auxiliary* tasks to improve the classification performance of a *target* task T_0 for which training data is scarce.

3.2 A brief overview of transfer learning

Transfer learning has had a relative long history in machine learning. Broadly speaking, the goal of a transfer algorithm is to use supervised training data from related tasks to improve performance. This is usually achieved by training classifiers for related tasks jointly. What is meant by joint training depends on the transfer learning framework; each of them develops a particular notion of relatedness and the corresponding transfer algorithms designed to exploit it.

We can distinguish three main lines of research in transfer learning: *learning hidden representations*, *feature sharing* and *hierarchical bayesian learning*. The work that we present in chapter 4 is an instance of *learning hidden representations* and the transfer algorithm presented in chapters 5 and 6 is an instance of *feature sharing*.

In *learning hidden representations* the relatedness assumption is that there exists a mapping from the original input space to an underlying shared feature representation. This latent representation captures the information necessary for training classifiers for all related tasks. The goal of a transfer algorithm is to simultaneously uncover the underlying shared representation and the parameters of the task-specific classifiers.

For example, consider learning a set of image classifiers for predicting whether an image belongs to a particular story on the news or not. To achieve this goal we start with a basic low level image representation such as responses of local filters. In a *learning hidden representations* framework we assume that there exists a hidden transformation that will produce a new representation that is good for training classifiers in this domain. In this example, the underlying representation would capture the semantics of an image and map semantically-equivalent features to the same high level meta-feature.

Broadly speaking, in a *feature sharing* framework the relatedness assumption is that tasks share relevant features. More specifically, we can differentiate two types of feature sharing approaches: *parameter tying* approaches and *joint sparsity* approaches.

In the *parameter tying* framework we assume that optimal classifier parameters for related tasks will lie close to each other. This is a rather strong assumption since it requires related tasks to weight features similarly. To relax this assumption we could only require the tasks to share the same set of non-zero parameters. This is what is assumed in the *joint sparsity* framework, i.e. the relatedness assumption is that to train accurate classifiers for a set of related tasks we only need to consider a small subset of relevant features. The goal of a transfer algorithm is to simultaneously discover the subset of relevant features and the parameter values for the task-specific classifiers.

Returning to the news story prediction problem, consider using a non-parametric or kernel based image representation as our starting point. More precisely, consider a representation where every image is represented by its similarity to a large set of unlabeled images. For this example, the transfer assumption states that there is a subset of prototypical unlabeled images such that knowing the similarity of a new image to this subset would suffice for predicting its story label.

One of the main differences between *learning hidden representations* and *feature sharing* approaches is that while the first one infers hidden representations the latter one chooses shared features from a large pool of candidates. This means that a *feature sharing* transfer algorithm must be able to efficiently search over a large set of features.

Feature sharing transfer algorithms are suitable for applications where we can generate a large set of candidate features from which to select a shared subset. Furthermore, this approach is a good fit when finding the subset of relevant features is in itself a goal. For example, in computational biology one might be interested in finding a subset of genes that are markers for a group of related diseases.

In a *hierarchical bayesian learning* framework tasks are assumed to be related by means of sharing a common prior distribution over classifiers's parameters. This shared prior can be learned in a classical hierarchical bayesian setting using data from related tasks. By sharing information across different tasks the prior parameter distribution can be better estimated.

3.3 Learning Hidden Representations

3.3.1 Transfer Learning with Neural Networks

One of the earliest works on transfer learning was that of Thrun (1996) who introduced the concept of *asymmetric transfer*. Thrun proposed a transfer algorithm that uses $\mathcal{D}$ to learn an underlying feature representation: $v(\mathbf{x})$. The main idea is to find a new representation where every pair of positive examples for a task will lie close to each other while every pair of positive and negative examples will lie far from each other.

Let $\mathcal{P}^k$ be the set of positive samples for the k-th task and $\mathcal{N}^k$ the set of negative samples, Thrun's transfer algorithm minimizes the following objective:

$$\min_{v \in \mathcal{V}} \sum_{k=1}^{m} \sum_{\mathbf{x}_i \in P^k} \sum_{\mathbf{x}_j \in P^k} ||v(\mathbf{x}_i) - v(\mathbf{x}_j)||_2 - \sum_{\mathbf{x}_i \in P^k} \sum_{\mathbf{x}_j \in N^k} ||v(\mathbf{x}_i) - v(\mathbf{x}_j)||_2 \tag{3.1}$$

where $\mathcal{V}$ is the set of transformations encoded by a two layer neural network. The transformation $v(\mathbf{x})$ learned from the *auxiliary* training data is then used to project the samples of the *target* task: $T_0' = \{(v(\mathbf{x}_1), y_1), \ldots, (v(\mathbf{x}_n), y_n)\}$. Classification for the *target* task is performed by running a nearest neighbor classifier in the new space.

The paper presented experiments on a small object recognition task consisting of recognizing 10 different objects. The results showed that when labeled data for the *target* task is scarce, the representation obtained by running their transfer algorithm on *auxiliary* training data could improve the classification performance of a *target* task.

3.3.2 Structural Learning

The ideas of Thrun were further generalized by several authors (Ando and Zhang, 2005; Argyriou et al., 2006; Amit et al., 2007). The three works that we review in the reminder of this section can all be casted under the framework of *structure learning* (Ando and Zhang, 2005) [1]. We start by giving an overview of this framework, followed by a discussion of three particular instantiations of the approach.

[1] Note that *structure learning* is different from structure learning which refers to learning in structured output domains, e.g. parsing

In a *structure learning* framework one assumes the existence of task-specific parameters $\mathbf{w}_k$ for each task and shared parameters θ that parameterize a family of underlying transformations. Both the structural parameters and the task-specific parameters are learned together via joint risk minimization on some supervised training data $\mathcal{D}$ for m related tasks.

Consider learning linear predictors of the form : $h_k(\mathbf{x}) = \mathbf{w}_k^T \, v(\mathbf{x})$ for some $\mathbf{w} \in R^z$ and some transformation $v : \mathbb{R}^d \rightarrow \mathbb{R}^z \in \mathcal{V}$. In particular, let $\mathcal{V}$ be the family of linear transformations: $v^\theta(\mathbf{x}) = \theta \, \mathbf{x}$ where θ is a z by d matrix that maps a d dimensional input vector to a z dimensional space[2].

We can now define the task-specific parameters matrix: $W = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ where $\mathbf{w}_k \in \mathbb{R}^z$ are the parameters for the $k\text{-}th$ task and $w_{j,k}$ is the parameter value for the $j\text{-}th$ hidden feature and the $k\text{-}th$ task. A *structure learning* algorithm finds the optimal task-specific parameters W^* and structural parameters θ^* by minimizing a jointly regularized empirical risk:

$$\text{argmin}_{W,\theta} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathsf{Loss}(\mathbf{w}_k^T \, \theta \, \mathbf{x}_i^k, y_i^k) + \gamma \Phi(W) + \lambda \Psi(\theta) \tag{3.2}$$

The first term in (3.2) measures the mean error of the m classifiers by means of some loss function Loss. The second term is a regularization penalty on the task-specific parameters W and the last term is a regularization penalty on the structural parameters θ. Different choices of regularization functions $\Phi(W)$ and $\Psi(\theta)$ result on different *structure learning* algorithms.

Sharing a Low Dimensional Feature Representation

Ando and Zhang (2005) combine an l_2 regularization penalty on the task-specific parameters: $\sum_{k=1}^{m} ||\mathbf{w}_k||_2^2$ with an orthonomal constraint on the structural parameters: $\theta\theta^T = I$, resulting in the following objective:

[2]For some of the approaches reviewed in this section $z < d$ and thus the transformation projects the inputs to a shared lower dimensional space. For other approaches $z = d$ and feature sharing will be realized by other means.

$$\text{argmin}_{W,\theta} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathsf{Loss}(\mathbf{w}_k^T \, \theta \, \mathbf{x}, y_i^k) + \gamma \sum_{k=1}^{m} ||\mathbf{w}_k||_2^2 \tag{3.3}$$

$$s.t. \theta\theta^T = I \tag{3.4}$$

where θ is a z by d matrix, z is assumed to be smaller than d and its optimal value is found using a validation set. Thus, feature sharing in this approach is realized by mapping the high dimensional feature vector $\mathbf{x}$ to a shared low dimensional feature space $\theta\mathbf{x}$.

Ando and Zhang proposed to minimize (3.4) using an alternating minimization procedure. Their algorithm will be described in more detail in chapter 4 where we will apply their approach to an image classification task.

In Ando and Zhang (2005) this transfer algorithm is applied in the context of *asymmetric transfer* where *auxiliary* training sets are utilized to learn the structural parameter θ. The structural parameter is then used to project the samples of the *target* task and train a classifier on the new space.

The paper presented experiments on text categorization where the *auxiliary* training sets were automatically derived from unlabeled data. More precisely, the *auxiliary* tasks consisted of predicting frequent content words for a set of unlabeled documents.

Note that given that the *auxiliary* training sets are automatically derived from unlabeled data, their transfer algorithm can be regarded as a semi-supervised training algorithm. Their results showed that the semi-supervised method gave significant improvements over a baseline method that trained on the labeled data ignoring the *auxiliary* training sets.

Sharing Hidden Features by a Sparse Regularization on the Latent Space

Argyriou et al. (2006) proposed an alternative model to learn shared hidden representations. In their approach the structural parameter θ is assumed to be a d by d matrix, i.e. the linear transformation does not map the inputs $\mathbf{x}$ to a lower dimensional space. Instead, sharing of hidden features across tasks is realized by a regularization penalty imposed on the task-specific parameters W.

Consider the matrix $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]$ and the joint minimization problem on $\mathcal{D}$. The $\mathbf{w}^j$ row of W corresponds to the parameter values for hidden feature j across the m

tasks. Requiring only a few hidden features to be used by any task is equivalent to requiring only a few rows of W to be non-zero.

We can achieve this goal by imposing a regularization penalty on the parameter matrix W. Argyriou et al. proposed the use of the following matrix norm: $l_{1,2}(W) = \sum_{j=1}^{z} ||\mathbf{w}^j||_2$. The $l_{1,2}$ norm is known to promote row sparsity in W (see section 3.4).

With these ingredients the problem of learning a few hidden features shared across tasks can be formulated as:

$$\min_{W,\theta} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathsf{Loss}(\mathbf{w}_k^T \, \theta \, \mathbf{x}, y_i^k) + \gamma l_{1,2}(W) \tag{3.5}$$

The constant γ controls the amount of regularization in the joint model.

The authors showed that (3.5) is equivalent to a convex problem for which they developed an alternating minimization algorithm. The algorithm involves two steps, in the first step the parameters W for each classifier are trained independently of each other. In a second step, for a fixed W they find the hidden structure θ by solving an eigenvalue problem.

The paper presented experiments on a product rating problem where the goal is to predict ratings given by different subjects. In the context of multitask learning predicting the ratings for a single subject can be regarded as a task. The transfer learning assumption is that predictions made by different subjects are related. The results showed that their transfer algorithm gave better performance than a baseline model where each task was trained independently with an l_1 penalty.

Learning Shared Structures by Finding Low Rank Parameters Matrices

Amit et al. (2007) proposed a regularization scheme for transfer learning based on a trace norm regularization penalty. Consider the following m by d parameter matrix $W = [\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^m]$, where each row corresponds to the parameters of one task.

Using the above regularization penalty Amit et al. transfer algorithm minimizes the following jointly regularized objective:

$$\min_{W} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathsf{Loss}(\mathbf{w}_k^T \mathbf{x}_i^k, y_i^k) + \gamma \Omega(W) \tag{3.6}$$

where $\Omega(W) = \sum_i |\gamma_i|$, and γ_i is the i-th eigenvalue of W. This norm is used because it is known to induce solution matrices W of low rank (Srebro and Jaakkola., 2003). Recall that the rank of a d by m matrix W is the minimum z such that W can be factored as $W = W'^t\,\theta$, for a z by m matrix W' and a z by d matrix θ.

Notice that θ is no longer in (3.6), this is because in this formulation of *structure learning* we do not search explicitly for a transformation θ. Instead, we utilize the regularization penalty $\Omega(W)$ to encourage solutions where the task-specific parameters W can be expressed as the combination of a few basis shared across tasks.

The optimization problem in (3.6) can be expressed as a semi-definite program and solved with an interior-point method. However, the authors argue that interior point methods scale poorly with the size of the training set and proposed a gradient based method to solve (3.6). The gradient method minimizes a smoothed approximation of (3.6).

In addition to the primal formulation, the paper presents a kernelized version of (3.6). It is shown that although the weigh vectors $\mathbf{w}'_k$ can not be directly retrieved from the dual solution, they can be found by solving a linear program on m variables.

The authors conducted experiments on a multiclass image classification task where the goal is to distinguish between 72 classes of mammals. The performance of their transfer learning algorithm is compared to that of a baseline svm-multiclass classifier. Their results show that the trace-norm penalty can improve multiclass accuracy when only a few samples are available for training.

3.3.3 Transfer by Learning A Distance Metric

In Fink (2004) the authors proposed an *asymmetric transfer* algorithm. Similar to Thrun (1996), this algorithm learns a distance metric using *auxiliary* data. This distance metric is then utilized by a nearest neighbor classifier for a *target* task for which it is assumed that there is a single positive and negative training example.

While Thrun's transfer algorithm followed a neural network approach, Fink's transfer

algorithm follows a max-margin approach. Consider learning a function $d : X \times X \rightarrow \mathbb{R}$ which has the following properties: (i) $d(x, x') \geq 0$, (ii) $d(x, x') = d(x', x)$ and (iii) $d(x, x') + d(x', x'') \geq d(x, x'')$. A function satisfying these three properties is called a pseudo-metric.

Fink's transfer algorithm learns a pseudo-metric using $\mathcal{D}$. Ideally, we would like to learn a function d that assigns a smaller distance to pairs having the same label than to pairs with different labels. More precisely, we could require the difference in distance to be a least γ for every $\mathcal{T}_k \in \mathcal{D}$. That is for every *auxiliary* task we must have that:

$$\forall_{(\mathbf{x}_i,1),(\mathbf{x}_j,1)\in\mathcal{T}_k} \; \forall_{(\mathbf{x}_q,-1)\in\mathcal{T}_k} \, d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_q) - \gamma \tag{3.7}$$

In particular, if we restrict ourselves to pseudo-metrics of the form: $d(\mathbf{x}_i, \mathbf{x}_j)^2 = ||\theta\,\mathbf{x}_i - \theta\,\mathbf{x}_j||_2^2$ we can reduce the problem of learning a pseudo-metric on $\mathcal{D}$ to learning a linear projection θ that achieves γ separation.

The underlying transfer learning assumption is that a projection θ that achieves γ separation on the *auxiliary* tasks will most likely achieve γ separation on the *target* task. Therefore, if we project the target samples using θ and run a nearest neighbor classifier in the new space we are likely to get a good performance.

For learning θ, the authors chose the online metric learning algorithm of Shalev-Shwartz et al. (2004). One of the advantages of this algorithm is that it has a dual form that allows the use of kernels. This dual version of the algorithm is the one used by the authors. The paper presented experiments on a character recognition dataset where they showed that the learned pseudo-metric could significantly improve performance on the *target* task.

3.4 Feature Sharing

3.4.1 Enforcing Parameter Similarity by Minimizing The Euclidean Distance between Parameter Vectors

We start this section with two *parameter tying* transfer algorithms. Evgeniou and Pontil (2004) proposed a simple regularization scheme for transfer learning that encourages the

task-specific parameters of classifiers for related tasks to be similar to each other. More precisely, consider training m linear classifiers of the form:

$$h_k(\mathbf{x}) = (\mathbf{v} + \mathbf{w}_k)^T \mathbf{x} \tag{3.8}$$

The parameter vector $\mathbf{v} \in \mathbb{R}^d$ is shared by the m tasks while the parameters $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]$ for $\mathbf{w}_k \in \mathbb{R}^d$ are specific to each task. As in *structure learning*, the goal of the transfer algorithm is to estimate the task-specific parameters W and the shared parameter $\mathbf{v}$ simultaneously using supervised training data from $\mathcal{D}$.

Let us define the parameter matrix $W' = [(\mathbf{w}_1 + \mathbf{v}), (\mathbf{w}_2 + \mathbf{v}), \ldots, (\mathbf{w}_m + \mathbf{v})]$ where $(\mathbf{w}_k + \mathbf{v}) \in \mathbb{R}^d$ are the parameters for the k-th task. Notice that any matrix can be written in the above form for some offset $\mathbf{v}$, thus without imposing a regularization scheme on $\mathbf{v}$ and $\mathbf{w}_k$ training classifiers of the form (3.8) reduces to training m independent linear classifiers. Evgeniou and Pontil proposed to minimize the following regularized joint objective:

$$\min_{W'} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathsf{Loss}(\mathbf{w}_k'^T \mathbf{x}_i^k, y_i^k) + \frac{\gamma}{m} \sum_{k=1}^{m} ||\mathbf{w}_k||_2^2 + \lambda ||\mathbf{v}||_2^2 \tag{3.9}$$

Intuitively, the l_2 regularization penalty on shared parameters $\mathbf{v}$ controls the norm of the average classifier for the m tasks while the l_2 penalty on $\mathbf{w}_k$ controls how much the task-specific parameters: $(\mathbf{v} + \mathbf{w}_k)$ differ from this average classifier.

The ratio between the regularization constants γ and λ determines the amount of parameter sharing enforced in the joint model. When $\frac{\gamma}{m} > \lambda$ the model penalizes more strongly deviations from the average model. Thus a large γ favors solutions where the parameters of each classifier $\mathbf{w}_k'$ are similar to each other. On the other hand when $\lambda > \frac{\gamma}{m}$ the regularization penalty will favor solutions where $\mathbf{v}$ is close to zero, making the task-parameters more dissimilar to each other.

In other words, when $\frac{\gamma}{m}$ tends to infinity the transfer algorithm reduces to pooling all supervised data in $\mathcal{D}$ to train a single classifier with parameters $\mathbf{v}$. On the other hand, when λ tends to infinity the transfer algorithm reduces to solving m independent tasks.

When the hinge loss is used in optimization (3.9) its Lagrangian reveals that at the optimal solution W'^*:

$$\mathbf{v}^* = \frac{\lambda}{(\lambda+\gamma)\,m} \sum_{k=1}^{m} \mathbf{w}_k \tag{3.10}$$

This suggests that the minimization in (3.9) can be expressed solely in terms of W, in particular the authors show that (3.9) is equivalent to:

$$\min_{W} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathsf{Loss}(\mathbf{w}_k'^T \mathbf{x}_i^k, y_i^k) + \gamma \sum_{k=1}^{m} ||\mathbf{w}_k||_2^2 + \lambda \sum_{k=1}^{m} ||\mathbf{w}_k - \frac{1}{m} \sum_{k'=1}^{m} \mathbf{w}_{k'}||_2^2 \tag{3.11}$$

The first regularization term is the usual l_2 penalty on the parameters of each classifier while the second regularization term favors solutions were the parameters of each classifier are similar to each other.

The problem (3.11) can be expressed as a quadratic program and solved with standard interior point methods. In addition to the primal formulation the authors present a dual version of (3.11) that enables the use of kernels.

The transfer algorithm was tested on the school data from the UCI machine learning repository. The goal with this data is to predict exam scores of students from different schools. When modeled as a multitask problem predicting scores for the students of a given school is regarded as a task. The authors compared their transfer algorithm with the transfer algorithm of Bakker and Heskes (2003). Their results showed that sharing information across tasks using their approach lead to better performance.

Notice that this algorithm makes a very strong relatedness assumption (i.e. the weights of the classifiers must be similar to each other). For this reason it is only appropriate for transfer learning applications where the tasks are closely related to each other.

3.4.2 Sharing Parameters in a Class Taxonomy

For multiclass problems where classes are organized in a hierarchical taxonomy, Cai and Hofmann proposed a max-margin approach that uses discriminant functions structured to mirror the class hierarchy.

More specifically, assume a multiclass problem with m classes organized in a tree class taxonomy: $H =< \mathcal{G}, \mathcal{E} >$, where $\mathcal{G}$ is a set of nodes and $\mathcal{E}$ is a set of edges. The leaves

of H correspond to the m classes and for convenience we will label the leave nodes using indexes from 1 to m. In addition, assume a weight matrix $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{|\mathcal{G}|}]$, where $\mathbf{w}_n$ is a weight vector associated with node $n \in \mathcal{G}$. Consider a discriminant function that measures the compatibility between an input $\mathbf{x}$ and a class label $y \in \{1, 2, \ldots, m\}$ of the form:

$$f(\mathbf{x}, y, W) = \sum_{j=1}^{|\mathcal{P}_y|} w_{\mathcal{P}_y^j}^T \mathbf{x} \tag{3.12}$$

where $\mathcal{P}_y^j$ is the j-th node on the path from the root node to node y.

The discriminant function in (3.12) causes the parameters for internal nodes of the class taxonomy to be shared across classes . It is easy to see that (3.12) can be written as a standard discriminant function $f(\phi(\mathbf{x}, y), W)$ for some mapping $\phi(\mathbf{x}, y)$ and hence standard SVM optimization methods can be used.

The paper presented results on document categorization showing the advantages of leveraging the taxonomy.

3.4.3 Clustering Tasks

Evgeniou and Pontil's transfer algorithm assumes that the parameters of all tasks are similar to each other, i.e. it is assumed that there is a single cluster of parameter vectors with mean $\mathbf{v}$. For some multitask settings such assumption might be too restrictive because not all tasks might be related to each other. However, a given a set of tasks might contain subsets or clusters of related tasks.

The work of Jacob et al. (2008) addresses this problem and proposes a regularization scheme that can take into account the fact that tasks might belong to different clusters. While Evgeniou and Pontil's transfer algorithm searches for a single mean vector $\mathbf{v}$, Jacob et al.'s transfer algorithm searches for r cluster means. More precisely, their algorithm searches for r cluster means $\mathbf{v}_r$ and cluster assignments for the task parameters such that the sum of the differences between mean $\mathbf{v}_p$ and each of the parameter vectors assigned to cluster p is minimized.

Searching for the optimal cluster assignments is a hard combinatorial problem, instead

the authors propose an approximation algorithm based on a convex approximation of the k-means objective.

3.4.4 Sharing A Feature Filter

Jebara (2004) developed a *joint sparsity* transfer algorithm under the Maximum entropy discrimination (MED) formalism. In a risk minimization framework one finds the optimal linear classifier parameters $\mathbf{w}^*$ by minimizing a regularized loss function on some training set $\mathcal{T}$. In contrast, in a MED framework we regard the classifier parameters $\mathbf{w}$ as a random variable and find a distribution $p(\mathbf{w})$ such that the expected loss on $\mathcal{T}$ is small, where the expectation is taken with respect to $p(\mathbf{w})$.

In addition in a MED framework one assumes some prior distribution $p_0(\mathbf{w})$ on the parameter vectors. The MED objective tries to find a distribution $p(\mathbf{w})$ which has the following properties: 1) it has small expected loss on $\mathcal{T}$ and 2) is close to the prior distribution, i.e. $p(\mathbf{w})$ has small relative Shannon entropy with respect to $p_0(\mathbf{w})$.

Putting it all together, the single task MED objective for the hinge loss is given by:

$$\min_{p(\mathbf{w}),\epsilon} \int_{\mathbf{w}} p(\mathbf{w}) \ln(\frac{p(\mathbf{w})}{p_0(\mathbf{w})}) \ \Delta\mathbf{w} \tag{3.13}$$

$$s.t. \forall_{i=1:n} \int_{\mathbf{w}} p(\mathbf{w}) \left[y_i \, \mathbf{w}^T x_i + \varepsilon_i \right] \geq 1 \tag{3.14}$$

where $\epsilon = [\epsilon_1, \epsilon_2, \ldots, \epsilon_n]$ are the standard slack variables resulting from the hinge loss. When the prior distribution $p_0(\mathbf{w})$ is assumed to be a zero-mean gaussian we recover the standard SVM objective.

To perform feature selection consider incorporating a binary feature filter into the linear classifier: $h(\mathbf{x}) = \sum_{j=1}^{d} s_j w_j x_j$. The feature filter is given by $\mathbf{s} = [s_1, s_2, \ldots, s_d]$, where each entry $s_j \in \{0, 1\}$ indicates whether a feature should be selected or not. We can now define a joint prior distribution over parameters $\mathbf{w}$ and feature filter $\mathbf{s}$: $p_0(\mathbf{w}, \mathbf{s}) = p_0(\mathbf{w}) \prod_{j=1}^{d} p_0(s_j)$.

A natural choice for $p_0(\mathbf{w})$ is to assume a zero-mean normal distribution, for $p_0(s_j)$ we will assume a Bernoulli distribution given by: $p_0(s_j) = \kappa^{s_j} \, (1-\kappa)^{1-s_j}$. Since $\mathbf{s}$ is a feature filter, the constant κ controls the amount of sparsity in the model.

To generalize the single task feature selection approach to the multitask case we simply assume that the feature filter $\mathbf{s}$ is shared across the m tasks. The joint prior for parameter matrix W and feature filter $\mathbf{s}$ is given by:

$$p_0(W, \mathbf{s}) = \prod_{k=1}^{m} p_0(\mathbf{w}_k) \prod_{j=1}^{d} p_0(s_j) \tag{3.15}$$

Putting it all together the joint feature selection transfer algorithm finds the task-specific parameter distribution $p(W)$ and shared feature filter distribution $p(\mathbf{s})$ by solving:

$$\min_{p(W,\mathbf{s}),[\epsilon_1,\ldots\epsilon_m]} \int_{\mathbf{w}} p(W, \mathbf{s}) \ln(\frac{p(W, \mathbf{s})}{p_0(W, \mathbf{s})}) \Delta(\mathbf{ws}) \tag{3.16}$$

$$s.t. \forall_{k=1:m} \forall_{i=1:n_k} \int_{\mathbf{w}_k} p(\mathbf{w}_k) [y_i^k \sum_{j=1}^{d} w_j^k s_j x_j^k + \varepsilon_i] \geq 1 \tag{3.17}$$

The authors propose to solve (3.17) using a gradient based method.

The paper presented experiments on the UCI multiclass dermatology dataset, where a one-vs-all classifier was trained for each class. Their results showed that multitask feature selection can improve the average performance.

3.4.5 Feature Sharing using $l_{1,2}$ Regularization

Obozinski et al. (2006) proposed a *joint sparsity* transfer algorithm based on $l_{1,2}$ regularization. Recall from section 3.3.2 that for a parameter matrix $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]$ the $l_{1,2}$ regularization penalty is defined as $l_{1,2}(W) = \sum_{j=1}^{d} ||\mathbf{w}^j||_2$. This norm has been shown to promote row sparsity.

In particular, Obozinski et al. (2008) studied the properties of $l_{1,2}$ regularization in the context of joint training of m regression functions. His study reveals that under certain conditions the $l_{1,2}$ regularized model can discover the subset of features (i.e. rows of W) that are non-zero in at least one of the m regression tasks.

In other words we can regard the $l_{1,2}$ norm as a convex relaxation to an l_{r0} penalty given by:

$$l_{r0}(W) = |\{j = 1 : d | max_k(w_{j,k}) \neq 0\}| \tag{3.18}$$

While in Argyriou et al. (2006) the regularization penalty worked on parameters corresponding to hidden features, in Obozinski et al. (2006) the regularization penalty is imposed directly on parameters corresponding to features x_j. This results in the following jointly regularized objective:

$$\min_{W} \sum_{k=1}^{m} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathsf{Loss}(\mathbf{w}_k^T \mathbf{x}_i^k, y_i^k) + \gamma \sum_{j=1}^{d} ||\mathbf{w}^j||_2 \tag{3.19}$$

where $\mathbf{w}^j$ are the coefficients of one feature across the m tasks. To optimize (3.19) the authors extended the path following coordinate descend algorithm of Zhao et al. (2007). Broadly speaking, a coordinate descend method is an algorithm that greedily optimizes one parameter at a time. In particular, Zhao et al.'s algorithm iterates between two steps: 1) a forward step which finds the feature that most reduces the objective loss and 2) a backward step which finds the feature that most reduces the regularized objective loss.

To use this algorithm in the context of multitask learning the authors modified the backward step to ensure that the parameters of one feature across the m tasks are simultaneously updated. Thus in the backward step they select the feature (i.e. $\mathbf{w}^j$ row of W) with largest directional derivative with respect to the $l_{1,2}$ regularized objective in (3.19).

The authors conducted experiments on a handwritten character recognition dataset, containing samples generated by different writers. Consider the binary task of distinguishing between a pair of characters, one possibility for solving this task is to ignore the different writers and learn a single l_1 regularized classifier pooling examples from all writers (i.e the pooling model). Another possibility is to train an l_1 classifier for each writer (i.e. the independent l_1 model). Yet another possibility is to train classifiers for all writers jointly with an $l_{1,2}$ regularization (i.e. the $l_{1,2}$ model). The paper compares these three approaches showing that joint $l_{1,2}$ regularization results in improved performance.

In addition, the paper presents results on a gene expression cancer dataset. Here the task is to find genetic markers (i.e subset of genes) that are predictive of four types of cancers. The data consists of gene signatures for both healthy and ill individuals. As in the handwritten recognition case, we could consider training independent l_1 classifiers to detect each type of cancer or training classifiers jointly with an $l_{1,2}$ regularization penalty. The

experiments showed that in terms of performance both approaches are indistinguishable. However, when we look at feature selection the $l_{1,2}$ selects significantly fewer genes.

3.4.6 Sharing Features Via Joint Boosting

Torralba et al. (2006) proposed a *feature sharing* transfer algorithm for multiclass object recognition based on boosting. The main idea is to reduce the computational cost of multiclass object recognition by making the m boosted classifiers share weak learners.

Let us first consider training a single classifier with a boosting algorithm. For this, we define $\mathcal{F} = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_q(\mathbf{x})\}$ to be a set of candidate weak learners. For example, $\mathcal{F}$ could be the family of weighted stumps, i.e. functions of the form:

$$f(\mathbf{x}) = a \; if \; (x_j > \beta) \tag{3.20}$$

$$= b \; if \; (x_j \leq \beta) \tag{3.21}$$

for some real weights $a, b \in \mathbb{R}$, some threshold $\beta \in \mathbb{R}$ and some feature index $j \in \{1, 2, \ldots, d\}$.

Assume that we wish to train an additive classifier of the form: $h(\mathbf{x}) = \sum_{f \in \Phi} f(\mathbf{x})$ that combines the outputs of some subset of weak learners $\Phi \subseteq \mathcal{F}$.

Boosting provides a simple way to sequentially add one weak learner at the time so as to minimize the exponential loss on the training set $\mathcal{T}$:

$$\min_{\Phi} \sum_{i=1}^{n} \exp^{-y_i \; \sum_{f \in \Phi} f(\mathbf{x})} \tag{3.22}$$

Several boosting algorithms have been proposed in the literature, this paper uses gentle boosting (Friedman et al., 1998). Gentle boosting performs an iterative greedy optimization, where at each step t we add a weak learner $f_t(\mathbf{x})$ to obtain a new classifier: $h^t(\mathbf{x}) = \sum_{j=1}^{t-1} f_j(\mathbf{x}) + f_t(\mathbf{x})$. The weak learner that is added at step t is given by:

$$argmin_{f_t \in F} \sum_{i=1}^{n} \exp^{-y_i \sum_{j=1}^{t-1} f_j(\mathbf{x}_i)} \left(y_i - f_t(\mathbf{x}_i)\right)^2 \tag{3.23}$$

In the multiclass case, the standard boosting approach is to learn an additive model for each class: $h^k(\mathbf{x}) = \sum_{f \in \Phi_k} f^k(\mathbf{x})$ by minimizing the joint exponential loss:

$$\min_{\{\Phi_1,\Phi_2,\ldots,\Phi_m\}} \sum_{i=1}^{n}\sum_{k=1}^{m} \exp^{-y_i^k \sum_{f\in\Phi_k} f(\mathbf{x})} \tag{3.24}$$

Notice that each class has its own set of weak learners Φ_k, i.e. there is no sharing of weak learners across classes. Torralba et al. proposes to enforce sharing of weak learners across classes by modifying the structure of the class specific additive models $h^k(\mathbf{x})$.

In particular, let us define $\mathcal{R}$ to be a subset of tasks: $\mathcal{R} \subseteq \{1, 2, \ldots, m\}$. For each such subset consider a corresponding additive classifier $h_{\mathcal{R}}(\mathbf{x}) = \sum_{f\in\Phi_{\mathcal{R}}} f(\mathbf{x})$ that performs the binary task of deciding whether an example belongs to any class in $\mathcal{R}$ or not.

Using these basic classifiers we can define an additive classifier for the k class of the form: $h_k(\mathbf{x}) = \sum_{\mathcal{R}:k\in\mathcal{R}} h_{\mathcal{R}}(\mathbf{x})$. At iteration t the joint boosting algorithm will add a new weak learner to one of the 2^m additive models $h_{\mathcal{R}}$ so as to minimize the joint loss on $\mathcal{D}$:

$$\sum_{k=1}^{m}\sum_{i=1}^{n} \exp^{-y_i^k \sum_{\mathcal{R}:k\in\mathcal{R}} h_{\mathcal{R}}(\mathbf{x})} \tag{3.25}$$

A naive implementation of (3.25) would require exploring all possible subsets of classes and would therefore have a computational cost of $O(d\,2^m)$ [3]. The authors show that when the weak learners are decision stumps a $O(d\,m^2)$ time search heuristic can be used to approximate (3.25).

The paper presented experiments on an object recognition dataset containing 21 object categories. The basic features (i.e. features utilized to create the decision stumps) used where normalized filter responses computed at different locations of the image.

They compare their joint boosting algorithm to a baseline that trains a boosted classifier for each class independently of the others. For the independent classifiers they limit the iterations of boosting so that the total number weak learners used by the m independently trained classifiers is the same as the total number of weak learners used by the classifiers trained with joint boosting.

Their results showed that for a fix total number of weak learners (i.e. for a given multi-class run-time performance) the accuracy of joint boosting is superior to that of independent boosting. One interesting result is that joint boosting tends to learn weak classifiers that are

[3] we have an $O(d)$ cost because every feature in X needs to be evaluated to find the best decision stump.

general enough to be useful for multiple classes. For example joint boosting will learn weak classifiers that can detect particular edge patterns, similar to the response properties of $V1$ cells.

3.5 Hierarchical Bayesian Learning

3.5.1 Transfer Learning with Hidden Features and Shared Priors

Bakker and Heskes (2003) presented a *hierarchical bayesian learning* model for transfer learning of m regression tasks (i.e. y is a real valued output). In the proposed model some of the parameters are assumed to be shared directly across tasks (like in the *structure learning* framework) while others are more loosely connected by means of sharing a common prior distribution. The idea of sharing a prior distribution is very similar to the approach of Argyriou et al. (2006) where we assume the existence of a shared mean parameter vector. The main difference is that in addition to the shared prior, Bakker and Heskes (2003) learns a hidden shared transformation.

The prior distribution and the shared parameters are inferred jointly using data from all related tasks in a maximum likelihood framework.

In particular, consider linear regression models of the form: $h(\mathbf{x}) = \sum_{j=1}^{z} w_j g_j^{\theta}(\mathbf{x})$ where z is the number of hidden features in the model and $g_j^{\theta}(\mathbf{x})$ is a function that returns the $j\text{-}th$ hidden feature.

As in most transfer algorithms we will have task-specific parameters $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]$ and shared parameters θ. In addition, we are going to assume that the task-specific parameters $\mathbf{w}_k$ are sampled from a shared gaussian prior distribution $\mathsf{N}(\mathbf{m}, \Sigma)$ with z-dimensional mean $\mathbf{m}$ and z by z covariance matrix Σ.

The joint distribution of task-specific model parameters W and train data $\mathcal{D}$ conditioned on shared parameters $\Lambda = (\theta, \mathbf{m}, \Sigma)$ is given by:

$$p(\mathcal{D},\, W|\Lambda) = \prod_{k=1}^{m} p(\mathcal{T}_k|\mathbf{w}_k, \theta)\, p(\mathbf{w}_k|\mathbf{m}, \Sigma) \tag{3.26}$$

where we are assuming that given shared parameters Λ the m tasks are independent.

To obtain optimal parameters Λ^* we integrate the task-specific parameters W and find the maximum likelihood solution. Once the maximum likelihood parameters Λ^* are known we can compute the task-specific parameters $\mathbf{w}_k^*$ by:

$$argmax_{\mathbf{w}_k} p(\mathbf{w}_k|\mathcal{T}_k, \Lambda^*) \tag{3.27}$$

We have assumed so far that all task-specific parameters are equally related to each other, i.e. they are all samples from a a single prior distribution $\mathsf{N}(\mathbf{m}, \Sigma)$. However, in real applications all tasks might not be related to each other but there might be some underlying clustering of tasks. The authors address this problem by replacing the single gaussian prior distribution with a mixture of q Gaussian distributions. Thus in the task clustering version of the model each task-specific parameter $\mathbf{w}_k$ is assumed to be sampled from: $\mathbf{w}_k \sim \sum_{r=1}^{q} \alpha_q \mathsf{N}(\mathbf{m}_r, \Sigma_r)$.

The paper presented experiments on the school dataset of the UCI repository and showed that their transfer algorithm improved performance as compared to training each task independently. In addition, the experiments showed that their transfer algorithm was able to provide a meaningful clustering of the different tasks.

3.5.2 Sharing a Prior Covariance

Raina et al. (2006) proposed a Bayesian logistic regression algorithm for *asymmetric transfer*. Their algorithm uses data from *auxiliary* tasks to learn a prior distribution on model parameters. In particular, the learnt prior encodes useful underlying dependencies between pairs of parameters and it can be used to improve performance on a *target* task.

Consider a binary logistic regression model making predictions according to: $p(y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1+exp^{-\mathbf{w}^T \mathbf{x}}}$. For this model, the MAP parameters $\mathbf{w}^*$ are given by:

$$argmax_{\mathbf{w}} \sum_{i=1}^{n} \log(p(y = y^i|\mathbf{x}, \mathbf{w})) + \lambda p_0(\mathbf{w}) \tag{3.28}$$

The left hand term is the likelihood of the training data under our model. The right hand term is a prior distribution on parameter vectors $\mathbf{w}$. The most common choice for $p_0(\mathbf{w})$ is a zero-mean multivariate gaussian prior $\mathsf{N}(\mathbf{0}, \Sigma)$. In general, Σ is defined to be an

identity covariance matrix, i.e we assume that the parameters are independent of each other and have equal prior variance.

The main idea of Raina et al. (2006) is to use data from *auxiliary* tasks to learn a more informative covariance matrix Σ. To give an intuition of why this might be helpful, think about a text categorization task where the goal is to predict whether an article belongs to a given topic or not. Consider a bag of words document representation, where every feature encodes the presence or absence of a word from some reference vocabulary. When training classifiers for the *auxiliary* tasks we might discover that the parameters for *moon* and *rocket* are positively correlated, i.e. when they occur in a document they will typically predict the same label. The transfer assumption is that the parameter correlations learnt on the *auxiliary* tasks can be predictive of parameter correlations on the *target* task.

In particular, the authors propose the following monte-carlo sampling algorithm to learn each entry: $E[w_j^*\ w_q^*]$ of Σ from *auxiliary* training data:

- Input: $\mathcal{D},\ j,\ q$
- Output: $E[w_j^*\ w_q^*]$
- for $p = 1$ to $p < z$
  - Choose a random *auxiliary* task: k
  - Chose a random subset of features: Λ that includes feature pair (j, q)
  - Train a logistic classifier on dataset $\mathcal{T}_k$ using only the features in Λ to obtain estimates: $w_j^p\ w_q^p$
- end
- return:$E[w_j^*\ w_q^*] = \frac{1}{z}\sum_{p=1}^{z} w_j^p\ w_q^p$

The paper presented results on a text categorization dataset containing documents from 20 different news-groups. The news-groups classes were randomly paired to construct 10 binary classification tasks. The *asymmetric transfer* learning experiment consisted of two steps. In the first step a prior covariance matrix Σ is learned using data from 9 *auxiliary*

tasks. In the second step this prior covariance is used in (3.28) to train a classifier for the 10-th held-out *target* task.

As a baseline model they train a logistic regression classifier for each task using an identity covariance matrix as a prior. Their results showed that for small training sets the covariance learnt from *auxiliary* data lead to lower test errors than the baseline model.

3.5.3 Learning Shape And Appearance Priors For Object Recognition

Fei-Fei et al. (2006) presented a *hierarchical bayesian learning* transfer algorithm for object recognition. Their transfer algorithm uses *auxiliary* training data to learn a prior distribution over object model parameters. This prior distribution is utilized to aid training of a *target* object classifier for which there is a single positive and negative training example.

Consider learning a probabilistic generative model for a given object class. We are going to assume that for every image we first compute a set of u **interesting regions** which we will use to construct an appearance and shape representation. In particular, we will represent each image using an appearance matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_u]$ where $\mathbf{a}_j$ is a feature representation for the j-th **region**, and a u by 2 shape matrix S containing the center location of each **region**.

A constellation object model assumes the existence of q latent object parts. More precisely, we define an assignment vector $\mathbf{r} = [r_1, r_2, \ldots, r_q]$ where $r_j \in \mathcal{R} = \{1, 2, \ldots u\}$ assigns one of the u **interesting regions** to the j-th latent object part. Using the latent assignment $\mathbf{r}$ and assuming appearance parameters $\mathbf{w}^A$ and shape parameters $\mathbf{w}^S$ we can write the generative object model as:

$$p(S, A, \mathbf{w}^A, \mathbf{w}^S) = \sum_{r \in \mathcal{R}} \prod_{p=1}^{q} p(\mathbf{a}_p | \mathbf{w}^A)\ p(S|\mathbf{w}^S)\, p(\mathbf{r})\ \ p(\mathbf{w}^A) p(\mathbf{w}^S) \tag{3.29}$$

Consider a normal distribution for both appearance and shape and let $\mathbf{w}^A = \{(\mu_1^A, \Sigma_1^A), \ldots, (\mu_q^A, \Sigma_q^A)\}$ and $\{\mu^S, \Sigma^S\}$ be the appearance and shape parameters respectively, we can re-write the generative object model as:

$$p(S, A, \mathbf{w}^A, \mathbf{w}^S) = \sum_{r \in \mathcal{R}} \prod_{p=1}^{q} \mathsf{N}(\mu_p^A, \Sigma_p^A)\ \mathsf{N}(\mu^S, \Sigma^S)\, p(\mathbf{r})\ \ p(\mathbf{w}^A) p(\mathbf{w}^S) \tag{3.30}$$

The first term in (3.30) is the likelihood of the observations $\{A, S\}$ under the object model. The second term is a prior distribution over appearance parameters $\mathbf{w}^A$ and shape parameters $\mathbf{w}^S$, these priors can be learned from *auxiliary* tasks, i.e. related object classes.

The prior distribution over a part mean appearance is assumed to be normally distributed: $\mu_j^A \sim \mathsf{N}(\beta^A, \Lambda^A)$ where β and Λ are the appearance hyper-parameters. Similarly, for the shape prior we have that: $\mu^S \sim \mathsf{N}(\gamma^S, \Upsilon^S)$ where γ^S and Υ^S are the shape hyper-parameters.

The proposed *asymmetric transfer* algorithm learns the shape and appearance hyper-parameters from related object classes and uses them when learning a *target* object model. In particular, consider learning an object model for the m *auxiliary* tasks using uniform shape and appearance priors and let μ_k^{S*} be the maximum likelihood parameters for object k. Their transfer algorithm computes the shape hyper-parameters: $\gamma^S \sim \frac{1}{m} \sum_{k=1}^{m} \mu_k^{S*}$. The appearance hyper-parameters are computed in an analogous manner using the *auxiliary* training data.

The paper presents results on a dataset of 101 object categories. The shared prior hyper-parameters are learnt using data from four object categories. This prior is then used when training objet models for the remaining target object classes. For these object classes they assume there is only one positive example available for training. The results showed that when only one sample is available for training the transfer algorithm can improve classification performance.

3.6 Related Work Summary

In this chapter we have reviewed three main approaches to transfer learning: *learning hidden representations* , *feature sharing* and *hierarchical bayesian learning* . As we would expect, each approach has its advantages and limitations.

The *learning hidden representations* approach is appealing because it can exploit latent

structures in the data. However, such a power comes at a relatively high computational cost: both Ando and Zhang (2005) and Argyriou et al. (2006) joint training algorithms make use of alternating optimization schemes where each iteration involves training m classifiers.

The *feature sharing* approach might not be able to uncover latent structures but as we will show in chapter 6 it can be implemented very efficiently, i.e. we can derive joint learning algorithms whose computational cost is in essence the same as independent training. This approach is appealing for applications were we can easily generate a large number of candidate features to choose from. This is the case in many image classification problems, for example in chapter 5 we generate a large set of candidate features using a kernel function and unlabeled examples.

The Bayesian approach has the advantage that it falls under a well studied family of hierarchical Bayesian methods. However, assuming a prior parametric distribution over model parameters might pose important limitations. For example, assuming a shared gaussian distribution over model parameters (Bakker and Heskes, 2003; Fei-Fei et al., 2006) implies that different tasks will weight features similarly. We will see in the experiments in chapter 6 that this assumption rarely holds. That is, it is common to find features that are useful for a pair of tasks but whose corresponding weights have opposite signs.

The other main limitation of Bayesian approaches is their computational cost; both Raina et al.'s and Fei-Fei et al.'s algorithms involve costly monte-carlo approximations. For example, each monte-carlo iteration of Raina et al.'s algorithm involves training a set of m classifiers.

The transfer algorithm that we present in chapters 5 and 6 falls under the *feature sharing* approach. We have reviewed four such algorithms: Jebara (2004) proposes an interesting formulation for joint feature selection but it has the limitation that it assumes that the probability of a feature being selected must follow a bernoulli distribution. Argyriou et al. proposed a simple and efficient *parameter tying* transfer algorithm. However, their algorithm suffers from the same limitation as Bakker and Heskes's, i.e. the transfer learning assumption is two strong since two related tasks might share the same relevant features but might weight them differently.

The two transfer algorithms most closely related to our work are the *joint sparsity*

algorithms of Obozinski et al. and Torralba et al.. Both these algorithms and the algorithm presented in chapters 5 and 6 can be thought as approximations to an l_{r0} regularized joint objective. The optimization algorithms proposed by Obozinski et al. and Torralba et al. are greedy (i.e. coordinate descend algorithms). In contrast, our transfer algorithm finds a global optimum by directly optimizing a convex relaxation of the l_{r0} penalty.

More precisely, we develop a simple and general optimization algorithm which for any convex loss has guaranteed convergence rates of $O(\frac{1}{\epsilon^2})$. One of the most attractive properties of our transfer algorithm is that its computational cost is comparable to the cost of training m independent sparse classifiers (with the most efficient algorithms for the task (Duchi et al., 2008)).

Chapter 4

Learning Image Representations Using Images with Captions

The work presented in this chapter was published in Quattoni et al. (2007).

In this chapter we will investigate a semisupervised image classification application of *asymmetric transfer* where the *auxiliary* tasks are derived automatically using unlabeled data. Thus any of the transfer learning algorithms described in 6.4 can be used in this semisupervised context.

In particular, we will use the Reuters Data described in chapter 2 and consider a setting where we have thousands of images with associated captions, but few images annotated with story news labels. We take the prediction of content words from the captions to be our *auxiliary* tasks and the prediction of a story label to be our *target* tasks.

Our goal is to leverage the *auxiliary* unlabeled data to derive a lower dimensional representation that still captures the relevant information necessary to discriminate between different stories. We will show how the *structure learning* framework of Ando and Zhang (2005) can be used to learn such a representation.

This chapter is organized as follows: Section 4.1 motivates the need for semi-supervised algorithms for image classification, Section 4.2 presents our model for learning image representations from unlabeled data annotated with meta-data based on *structure learning*, Section 4.3 illustrates the approach by giving some toy examples of the types of representations that it can learn and Section 4.5 describes experiments on image classification

where the goal is to predict the news-topic of an image and the meta-data are image captions. Finally section 4.7 summarizes the results of this chapter and highlights some open questions for future work.

4.1 Introduction

When few labeled examples are available, most current supervised learning methods classification may work poorly —for example when a user defines a new category and provides only a few labeled examples. To reach human performance, it is clear that knowledge beyond the supervised training data needs to be leveraged.

When labeled data is scarce it may be beneficial to use unlabeled data to learn an image representation that is low-dimensional, but nevertheless captures the information required to discriminate between image categories.

There is a large literature on semi-supervised learning approaches, where unlabeled data is used in addition to labeled data. We do not aim to give a full overview of this work, for a comprehensive survey article see (Seeger, 2001). Most semi-supervised learning techniques can be broadly grouped into three categories depending on how they make use of the unlabeled data: density estimation, dimensionality reduction via manifold learning and function regularization. Generative models trained via EM can naturally incorporate unlabeled data for classification tasks (Nigam et al., 2000; Baluja, 1998). In the context of discriminative category learning, Fisher kernels (Jaakkola and Haussler, 1998; Holub et al., 2005) have been used to exploit a learned generative model of the data space in an SVM classifier.

In some cases unlabeled data may contain useful meta-data that can be used to learn a low-dimensional representation that reflects the semantic content of an image. As one example, large quantities of images with associated natural language captions can be found on the web. This chapter describes an algorithm that uses images with captions or other meta-data to derive an image representation that allows significantly improved learning in cases where only a few labeled examples are available.

In our approach the meta-data is used to induce a representation that reflects an under-

lying part structure in an existing, high-dimensional visual representation. The new representation groups together synonymous visual features—features that consistently play a similar role across different image classification tasks.

Ando and Zhang introduced the *structure learning* framework, which makes use of *auxiliary* problems in leveraging unlabeled data. In this chapter we introduce *auxiliary* problems that are created from images with associated captions. Each *auxiliary* problem involves taking an image as input, and predicting whether or not a particular content word (e.g, *man*, *official*, or *celebrates*) is in the caption associated with that image. In *structure learning* , a separate linear classifier is trained for each of the *auxiliary* problems and manifold learning (e.g., SVD) is applied to the resulting set of parameter vectors, finding a low-dimensional space which is a good approximation to the space of possible parameter vectors. If features in the high-dimensional space correspond to the same semantic part, their associated classifier parameters (weights) across different *auxiliary* problems may be correlated in such a way that the basis functions learned by the SVD step collapse these two features to a single feature in a new, low-dimensional feature-vector representation.

In a first set of experiments, we use synthetic data examples to illustrate how the method can uncover latent part structures. We then describe experiments on classification of news images into different topics. We compare a baseline model that uses a bag-of-words SIFT representation of image data to our method, which replaces the SIFT representation with a new representation that is learned from 8,000 images with associated captions. In addition, we compare our method to (1) a baseline model that ignores the meta-data and learns a new visual representation by performing PCA on the unlabeled images and (2) a model that uses as a visual representation the output of word classifiers trained using captions and unlabeled data. Note that our goal is to build classifiers that work on images alone (i.e., images which *do not* have captions), and our experimental set-up reflects this, in that training and test examples for the topic classification tasks include image data only. The experiments show that our method significantly outperforms baseline models.

4.2 Learning Visual Representations

A good choice of representation of images will be crucial to the success of any model for image classification. The central focus of this chapter is a method for automatically learning a representation from images which are unlabeled, but which have associated meta-data, for example natural language captions. We are particularly interested in learning a representation that allows effective learning of image classifiers in situations where the number of training examples is small. The key to the approach is to use meta-data associated with the unlabeled images to form a set of *auxiliary* problems which drive the induction of an image representation. We assume the following scenario:

• We have labeled (supervised) data for an image classification task. We will call this the *target* task. For example, we might be interested in recovering images relevant to a particular topic in the news, in which case the labeled data would consist of images labeled with a binary distinction corresponding to whether or not they were relevant to the topic. We denote the labeled examples as the *target* set $\mathcal{T}_0 = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i)\}$ where $(\mathbf{x}_i, y_i)$ is the i'th image/label pair. Note that test data points for the *target* task contain image data alone (these images do not have associated caption data, for example).

• We have m *auxiliary*training sets, $\mathbf{T}_k = \{(\mathbf{x}^k_1, y^k_1), \ldots, (\mathbf{x}^k_{n_k}, y^i_{n_k})\}$ for $k = 1 \ldots m$. Here $\mathbf{x}^k_i$ is the k-th image in the k-th *auxiliary* training set, y^k_i is the label for that image, and n_k is the number of examples in the k-th training set. The *auxiliary* training sets consist of binary classification problems, distinct from the *target* task, where each y^0_i is in $\{-1, +1\}$. Shortly we will describe a method for constructing *auxiliary* training sets using images with captions.

• The aim is to learn a representation of images, i.e., a function that maps images $\mathbf{x}$ to feature vectors $v(\mathbf{x})$. The *auxiliary* training sets will be used as a source of information in learning this representation. The new representation will be applied when learning a classification model for the *target* task.

In the next section we will describe a method for inducing a representation from a set of *auxiliary* training sets. The intuition behind this method is to find a representation which is relatively simple (i.e., of low dimension), yet allows strong performance on the *auxiliary*

training sets. If the *auxiliary* tasks are sufficiently related to the *target* task, the learned representation will allow effective learning on the *target* task, even in cases where the number of training examples is small.

4.2.1 Learning Visual Representations from Auxiliary Tasks

This section describes the *structure learning* learning algorithm [1] for learning a representation from a set of *auxiliary* training sets.

We assume that $\mathbf{x} \in \mathbb{R}^d$ is a *baseline* feature vector representation of images. In the experiments in this chapter $\mathbf{x}$ is a SIFT histogram representation (Sivic et al., 2005). In general, $\mathbf{x}$ will be a "raw" representation of images that would be sufficient for learning an effective classifier with a large number of training examples, but which performs relatively poorly when the number of training examples is small. For example, with the SIFT representation the feature vectors $\mathbf{x}$ are of relatively high dimension (we use $d = 1,000$), making learning with small amounts of training data a challenging problem without additional information.

Note that one method for learning a representation from the unlabeled data would be to use PCA—or some other density estimation method—over the feature vectors $\mathcal{U} = \{\mathbf{x}_1, \ldots, \mathbf{x}_q\}$ for the set of unlabeled images (we will call this method the *data-SVD* method). The method we describe differs significantly from PCA and similar methods in its use of meta-data associated with the images, for example captions. Later we will describe synthetic experiments where PCA fails to find a useful representation, but our method is successful. In addition we describe experiments on real image data where PCA again fails, but our method is successful in recovering representations which significantly speed learning.

Given the baseline representation, the new representation is defined as $v(x) = \theta\mathbf{x}$ where θ is a projection matrix of dimension $z \times d$.[1] The value of z is typically chosen such that $z \ll d$. The projection matrix is learned from the set of *auxiliary* training sets, using the *structure learning* learning approach described in (Ando and Zhang, 2005). Figure 4-1

[1]Note that the restriction to linear projections is not necessarily limiting. It is possible to learn non-linear projections using the kernel trick; i.e., by expanding feature vectors $\mathbf{x}$ to a higher-dimensional space, then taking projections of this space.

Input 1: *auxiliary* training sets $\{(\mathbf{x}_1^k, y_1^k), \ldots, (\mathbf{x}_{n_k}^k, y_{n_k}^k)\}$ for $k = 1 \ldots m$. Here $\mathbf{x}_i^k$ is the i-th image in the k-th training set, y_i^k is the label for that image. n_k is the number of examples in the k-th training set. We consider binary classification problems, where each y_i^k is in $\{-1, +1\}$. Each image $\mathbf{x} \in \mathbb{R}^d$ some a feature representation for the image.

Input 2: *target* training set $\mathcal{T}_0\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$

Structural learning using *auxiliary* training sets:

Step 1: Train m linear classifiers. For $k = 1 \ldots m$, choose the optimal parameters on the k-th training set to be $\mathbf{w}_i^* = \arg\min_{\mathbf{w}} \mathsf{Loss}_k(\mathbf{w})$ where

$$\mathsf{Loss}_k(\mathbf{w}) = \sum_{i=1}^{n_k} \mathsf{Loss}(\mathbf{w} \cdot \mathbf{x}_i^k, y_i^k) + \frac{\gamma}{2}||\mathbf{w}||_2^2$$

(See section 4.2.1 for more discussion.)

Step 2: Perform SVD on the Parameter Vectors.

Form a matrix $\mathbf{W}$ of dimension $d \times m$, by taking the parameter vectors $\mathbf{w}_i^*$ for $k = 1 \ldots m$. Compute a projection matrix θ of dimension $z \times d$ by taking the first z eigenvectors of $\mathbf{WW}'$.

Output: The projection matrix $\theta \in \mathbb{R}^{z \times d}$.

Train using the *target* training set:

Define $v(\mathbf{x}) = \theta\mathbf{x}$.

Choose the optimal parameters on the *target* training set to be

$\mathbf{v}^* = \arg\min_{\mathbf{v}} \mathsf{Loss}(\mathbf{v})$ where

$$\mathsf{Loss}(\mathbf{v}) = \sum_{i=1}^{n} \mathsf{Loss}(\mathbf{v} \cdot v(\mathbf{x}_i), y_i) + \frac{\gamma_2}{2}||\mathbf{v}||_2^2$$

Figure 4-1: The *structure learning* learning algorithm (Ando and Zhang, 2005).

shows the algorithm.

In a first step, linear classifiers $\mathbf{w}_k^*$ are trained for each of the m *auxiliary* problems. In several parameter estimation methods, including logistic regression and support vector machines, the optimal parameters $\mathbf{w}^*$ are taken to be $\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathsf{Loss}(\mathbf{w})$ where $\mathsf{Loss}(\mathbf{w})$ takes the following form:

$$\mathsf{Loss}(\mathbf{w}) = \sum_{i=1}^{n} \mathsf{Loss}(\mathbf{w} \cdot (\mathbf{x}_i), y_i) + \frac{\gamma}{2}||\mathbf{w}||_2^2 \tag{4.1}$$

Here $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ is a set of training examples, where each $\mathbf{x}_i$ is an image and each y_i is a label. The constant $\gamma_2 > 0$ dictates the amount of regularization in the model. The function $\mathsf{Loss}(\mathbf{w} \cdot \mathbf{x}, y)$ is some measure of the loss for the parameters $\mathbf{w}$ on the example $(\mathbf{x}, y)$. For example, in support vector machines (Cortes and Vapnik, 2005) Loss is the hinge-loss, defined as $\mathsf{Loss}(m, y) = (1 - ym)_+$ where $(z)_+$ is z if $z >= 0$, and is 0 otherwise. In logistic regression the loss function is

$$\mathsf{Loss}(m, y) = -\log \frac{\exp\{ym\}}{1 + \exp\{ym\}}. \tag{4.2}$$

Throughout this chapter we use the loss function in (Eq. 4.2), and classify examples with $sign(\mathbf{w} \cdot \mathbf{x})$ where $sign(z)$ is 1 if $z \geq 0$, -1 otherwise.

In the second step, SVD is used to identify a matrix θ of dimension $z \times d$. The matrix defines a linear subspace of dimension z which is a good approximation to the space of induced weight vectors $\mathbf{w}_1^*, \ldots, \mathbf{w}_m^*$. Thus the approach amounts to *manifold learning in classifier weight space*. Note that there is a crucial difference between this approach and the data-SVD approach: in data-SVD SVD is run over the data space, whereas in this approach SVD is run over the space of parameter values. This leads to very different behaviors of the two methods.

The matrix θ is used to constrain learning of new problems. As in (Ando and Zhang, 2005) the parameter values are chosen to be $\mathbf{w}^* = \theta' \mathbf{v}^*$ where $\mathbf{v}^* = \arg\min_{\mathbf{v}} \mathsf{Loss}(\mathbf{v})$ and

$$\mathsf{Loss}(\mathbf{v}) = \sum_{i=1}^{n} \mathsf{Loss}((\theta' \mathbf{v}) \cdot \mathbf{x}_i, y_i) + \frac{\gamma}{2}||\mathbf{v}||^2 \tag{4.3}$$

This essentially corresponds to constraining the parameter vector $\mathbf{w}^*$ for the new problem to lie in the sub-space defined by θ. Hence we have effectively used the *auxiliary* training

problems to learn a sub-space constraint on the set of possible parameter vectors.

If we define $v(\mathbf{x}) = \theta\mathbf{x}$, it is simple to verify that

$$\mathsf{Loss}(\mathbf{v}) = \sum_{i=1}^{n} \mathsf{Loss}(\mathbf{v} \cdot v(\mathbf{x}_i), y_i) + \frac{\gamma}{2}||\mathbf{v}||_2^2 \tag{4.4}$$

and also that $sign(\mathbf{w}^* \cdot \mathbf{x}) = sign(\mathbf{v}^* \cdot v(\mathbf{x}))$. Hence an alternative view of the algorithm in figure 4-1 is that it induces a new representation $v(\mathbf{x})$. In summary, the algorithm in figure 4-1 derives a matrix θ that can be interpreted either as a sub-space constraint on the space of possible parameter vectors, or as defining a new representation $v(\mathbf{x}) = \theta\mathbf{x}$.

4.2.2 Metadata-derived *auxiliary* problems

A central question is how *auxiliary* training sets can be created for image data. A key contribution of this chapter is to show that unlabeled images which have associated text captions can be used to create *auxiliary* training sets, and that the representations learned with these unlabeled examples can significantly reduce the amount of training data required for a broad class of topic-classification problems. Note that in many cases, images with captions are readily available, and thus the set of captioned images available may be considerably larger than our set of labeled images.

Formally, denote a set of images with associated captions as $(x'_1, c_1), \ldots, (x'_q, c_q)$ where (x'_i, c_i) is the i'th image/caption pair. We base our m *auxiliary* training sets on m content words, $(w_1, \ldots, w_m)$. A natural choice for these words would be to choose the m most frequent content words seen within the captions.[2] m *auxiliary* training sets can then be created as follows. Define $I_i[c]$ to be 1 if word w_i is seen in caption c, and -1 otherwise. Create a training set $\mathcal{T}_k = \{(x'_1, I_k[c_1]), \ldots, (x'_q, I_k[c_q])\}$ for each $k = 1 \ldots m$. Thus the $k\text{-}th$ training set corresponds to the binary classification task of predicting whether or not the word w_k is seen in the caption for an image x'.

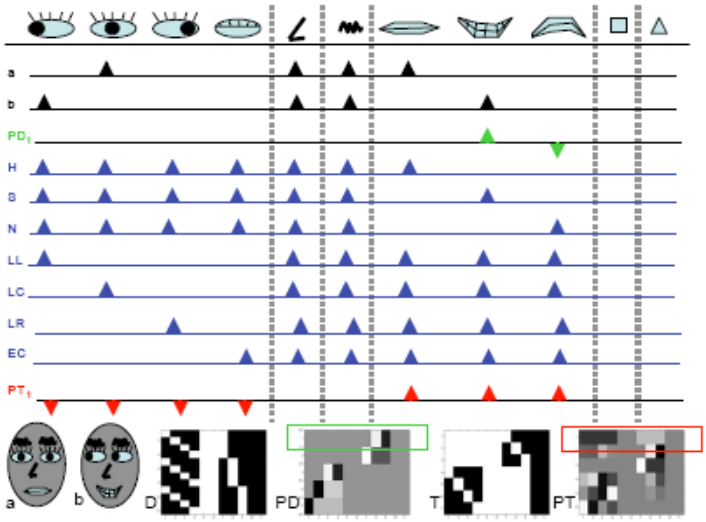

Figure 4-2: Concept figure illustrating how when appropriate *auxiliary* tasks have already been learned manifold learning in classifier weight space can group features corresponding to functionally defined visual parts. Parts (eyes, nose, mouth) of an object (face) may have distinct visual appearances (the top row of cartoon part appearances). A specific face (e.g., a or b) is represented with the boolean indicator vector as shown. Matrix D shows all possible faces given this simple model; PCA on D is shown row-wise in PD (first principal component is shown also above in green as PD_1.) No basis in PD groups together eye or mouth appearances; different part appearances never co-occur in D. However, idealized classifiers trained to recognize, e.g., faces with a particular mouth and any eye (H,S,N), or a particular eye and mouth (LL,LC,LR,EC), will learn to group features into parts. Matrix T and blue vectors above show these idealized boolean classifier weights; the first principal component of T is shown in red as PT_1, clearly grouping together the four cartoon eye and the three cartoon mouth appearances. Positive and negative components of PT_1 would be very useful features for future learning tasks related to faces in this simple domain because they group together different appearances of eyes and mouths.

(a)

a	A	*A*	A	**A**
b	***b***	***B***	**b**	*b*
c	C	*c*	c	***c***
		...		
j	J	*J*	J	**J**

(b)

A *b* **c**	*a b* D
A **b** *c*	**A** *b* D
a ***b*** c	a b d
A d E	*b c* f
A D E	***B*** c *f*
A D *e*	*b* C *f*

Figure 4-3: Synthetic data involving objects constructed from letters. (a) There are 10 possible parts, corresponding to the first 10 letters of the alphabet. Each part has 5 possible observations (corresponding to different fonts). (b) Each object consists of 3 distinct parts; the observation for each part is drawn uniformly at random from the set of possible observations for that part. A few random draws for 4 different objects are shown.

4.3 Examples Illustrating the Approach

Figure 4-2 shows a concept figure illustrating how PCA in a classifier weight space can discover functional part structures given idealized *auxiliary* tasks. When the tasks are defined such that to solve them they need to learn to group different visual appearances, the distinct part appearances will then become correlated in the weight space, and techniques such as PCA will be able to discover them. In practice the ability to obtain such ideal classifiers is critical to our method's success. Next we will describe a synthetic example where the method is successful; in the following section we present real-world examples where *auxiliary* tasks are readily available and yield features that speed learning of future tasks.

We now describe experiments on synthetic data that illustrate the approach. To generate the data, we assume that there is a set of 10 possible *parts*. Each *object* in our data consists of 3 distinct parts; hence there are $\binom{10}{3} = 120$ possible objects. Finally, each of the 10 parts has 5 possible *observations*, giving 50 possible observations in total (the observations for each part are distinct).

[2]In our experiments we define a content word to be any word which does not appear on a "stop list" of common function words in English.

As a simple example (see figure 4-3), the 10 parts might correspond to 10 letters of the alphabet. Each "object" then consists of 3 distinct letters from this set. The 5 possible observations for each part (letter) correspond to visually distinct realizations of that letter; for example, these could correspond to the same letter in different fonts, or the same letter with different degrees of rotation. The assumption is that each observation will end up as a distinct visual word, and therefore that there are 50 possible visual words.

The goal in learning a representation for object recognition in this task would be to learn that different observations from the same part are essentially equivalent—for example, that observations of the letter "a" in different fonts should be collapsed to the same point. This can be achieved by learning a projection matrix θ of dimension 10×50 which correctly maps the 50-dimensional observation space to the 10-dimensional part space. We show that the use of *auxiliary* training sets, as described in section 4.2.1, is successful in learning this structure, whereas PCA fails to find any useful structure in this domain.

To generate the synthetic data, we sample 100 instances of each of the 120 objects as follows. For a given object y, define P_y to be the set of parts that make up that object. For each part $p \in P_y$, generate a single observation uniformly at random from the set of possible observations for p. Each data point generated in this way consists of an object label y, together with a set of three observations, x. We can represent x by a 50-dimensional binary feature vector $\mathbf{x}$, where only 3 dimensions (corresponding to the three observations in x) are non-zero.

To apply the *auxiliary* data approach, we create 120 *auxiliary* training sets. The $k\text{-}th$ training set corresponds to the problem of discriminating between the $k\text{-}th$ object and all other 119 objects. A projection matrix θ is learned from the *auxiliary* training sets. In addition, we can also construct a projection matrix using PCA on the data points $\mathbf{x}$ alone. Figures 4-4 and 4-5 show the projections learned by PCA and the *auxiliary* tasks method. PCA fails to learn useful structure; in contrast the *auxiliary* task method correctly collapses observations for the same part to nearby points.

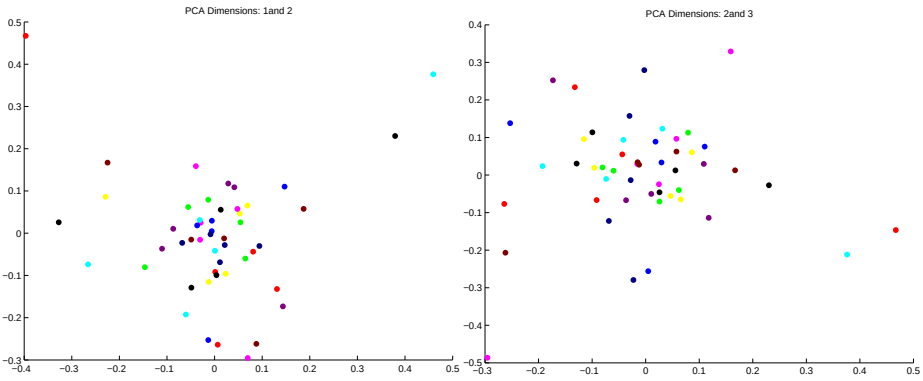

Figure 4-4: The representations learned by PCA on the synthetic data problem. The first figure shows projections 1 vs. 2; the second figure shows projections 2 vs. 3. Each plot shows 50 points corresponding to the 50 observations in the model; observations corresponding to the same part have the same color. There is no discernable structure in the figures. The remaining dimensions were found to similarly show no structure.

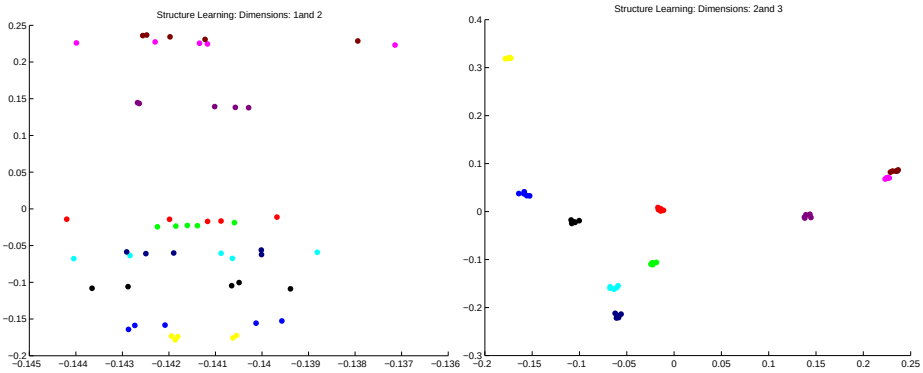

Figure 4-5: The representations learned by *structure learning* learning on the synthetic data problem. The first figure shows projections 1 vs. 2; the second figure shows projections 2 vs. 3. Each plot shows 50 points corresponding to the 50 observations in the model; observations corresponding to the same part have the same color. There is clear structure in features 2 and 3, in that observations corresponding to the same part are collapsed to nearby points in the projected space. The remaining dimensions were found to show similar structure to those in dimensions 2 and 3.

4.4 Reuters Image Dataset

For the experiments in chapters: 4, 5 and 6 we collected an image dataset from the Reuters news website (`http://today.reuters.com/news/`). Images on the Reuters website are associated to stories or topics, which correspond to different topics in the news. An image can belong to more than one topic, although most images are associated with a single topic. In addition to the topic label, every image contains a caption consisting of a few sentences describing the image.

This data has been collected over two different time periods, resulting on two versions of the dataset: Reuters.V1 and Reuters.V2. The Reuters.V1 version contains 10.382 images collected in February 2006 over a period of one week, covering a total of 108 news topics. The Reuters.V2 version contains all the images in Reuters.V1 plus images collected during a week in February 2007, resulting in a total of 20.382 images covering 258 topics. The topics span a wide range of categories including: world, sports, entertainment, and politics news stories.

Figure 4-6 shows some examples of news topics and corresponding images. Figures 4-7 to 4-12 show an image for each of the 258 topics in the dataset. The number next to the topic label indicates the number of images for that topic.

Figure 4-13 shows some images with their corresponding captions and Figure 4-14 shows some examples of the most frequent words appearing on the captions and corresponding images.

4.5 Experiments on Images with Captions

4.5.1 Data

For these section we used the dataset Reuters.V1 described in Section 4.4. The experiments involved predicting the topic variable y for test images. We reserved $8,000$ images as a source of training data, and an additional $1,000$ images as a potential source of development data. The remaining 1,382 images were used as a test set. Multiple training sets of different sizes, and for different topics, were created as follows. We created training sets $\mathcal{T}_{n,y}$ for

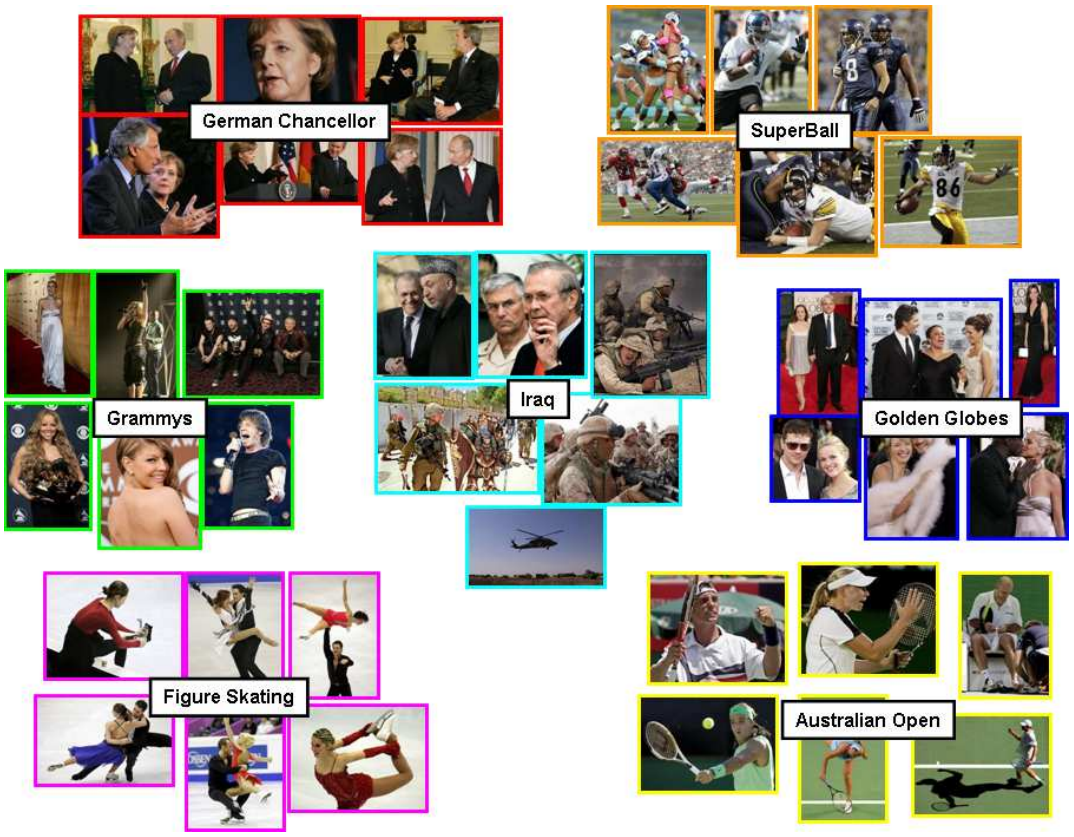

Figure 4-6: Example images from some news topics: German Chancellor, SuperBall, Grammys, Iraq, Golden Globes, Figure Skating and Australian Open

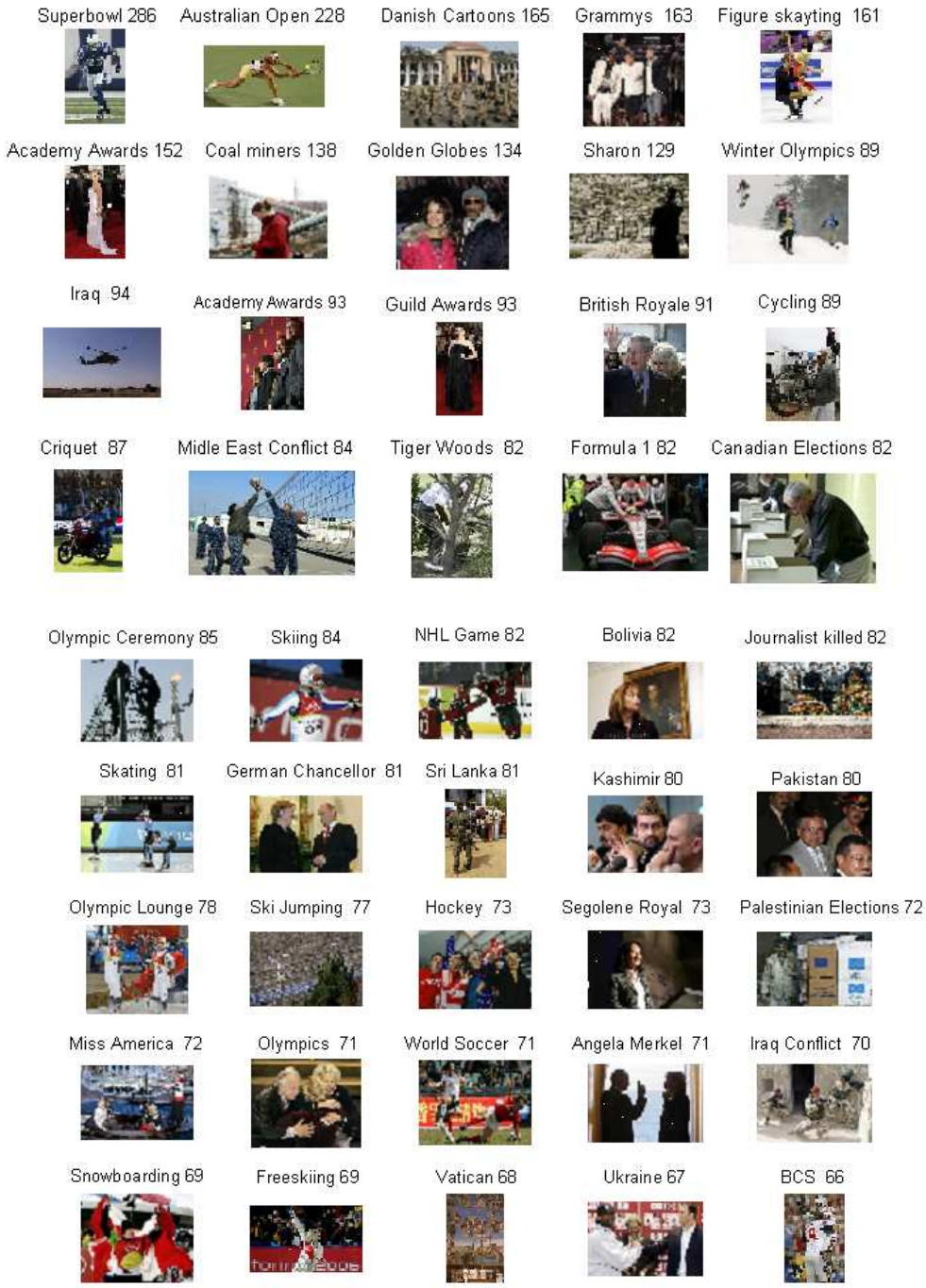

Figure 4-7: This figure shows topics ranked 1-45 in frequency. The number next to the topic label indicates the number of images for that topic.

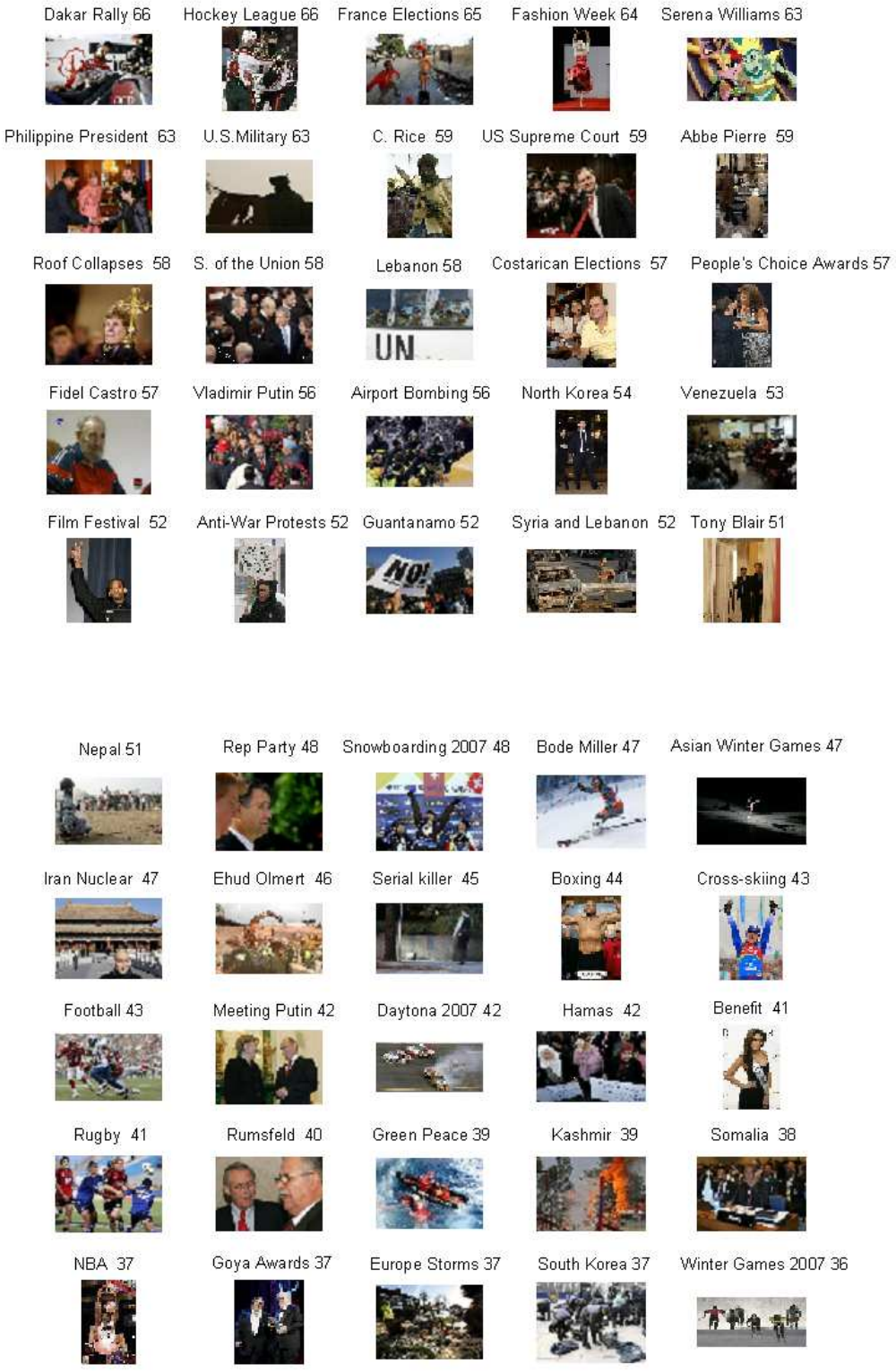

Figure 4-8: This figure shows topics ranked 46-95 in frequency. The number next to the topic label indicates the number of images for that topic.

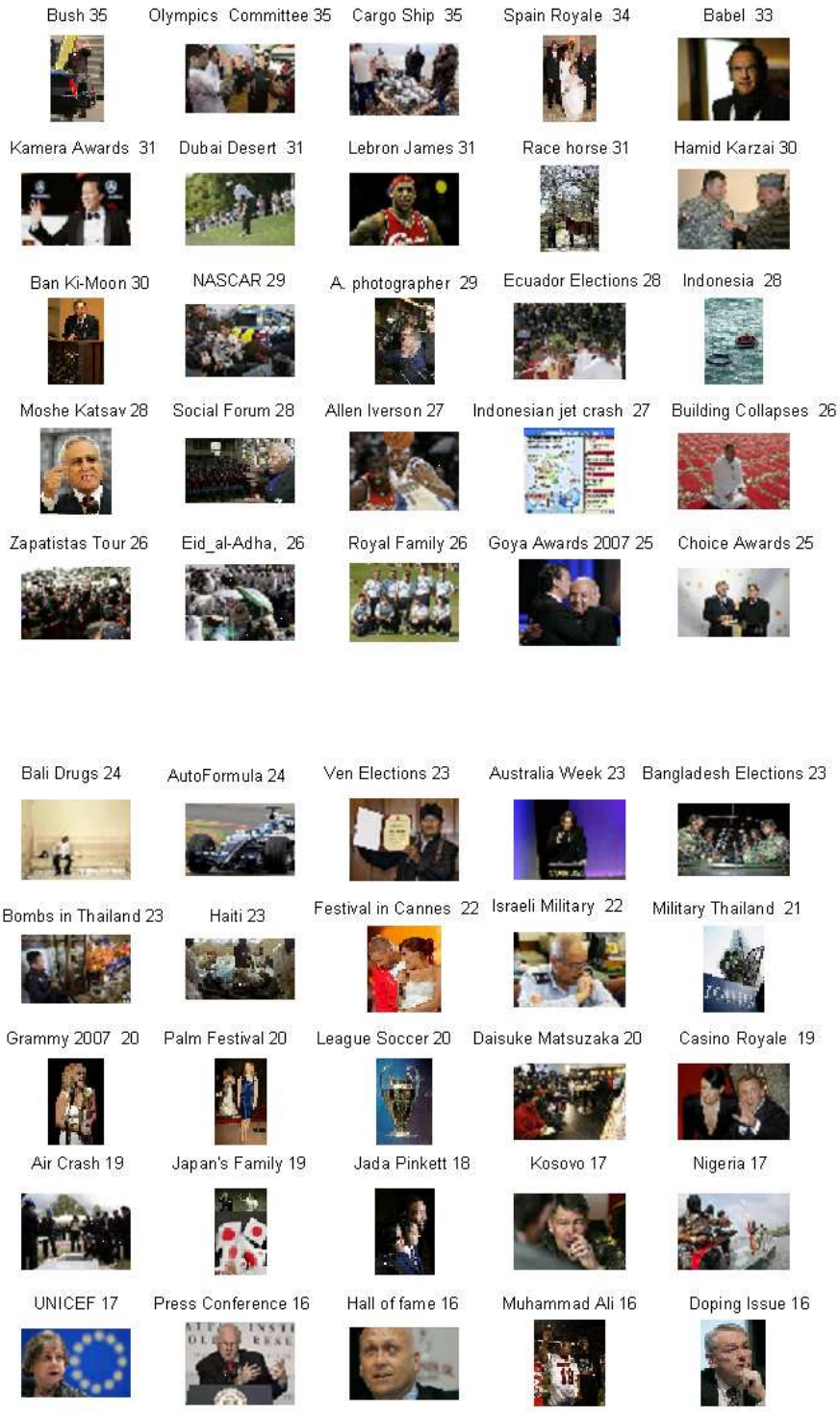

Figure 4-9: This figure shows topics ranked 96-145 in frequency. The number next to the topic label indicates the number of images for that topic.

Figure 4-10: This figure shows topics ranked 146-195 in frequency. The number next to the topic label indicates the number of images for that topic.

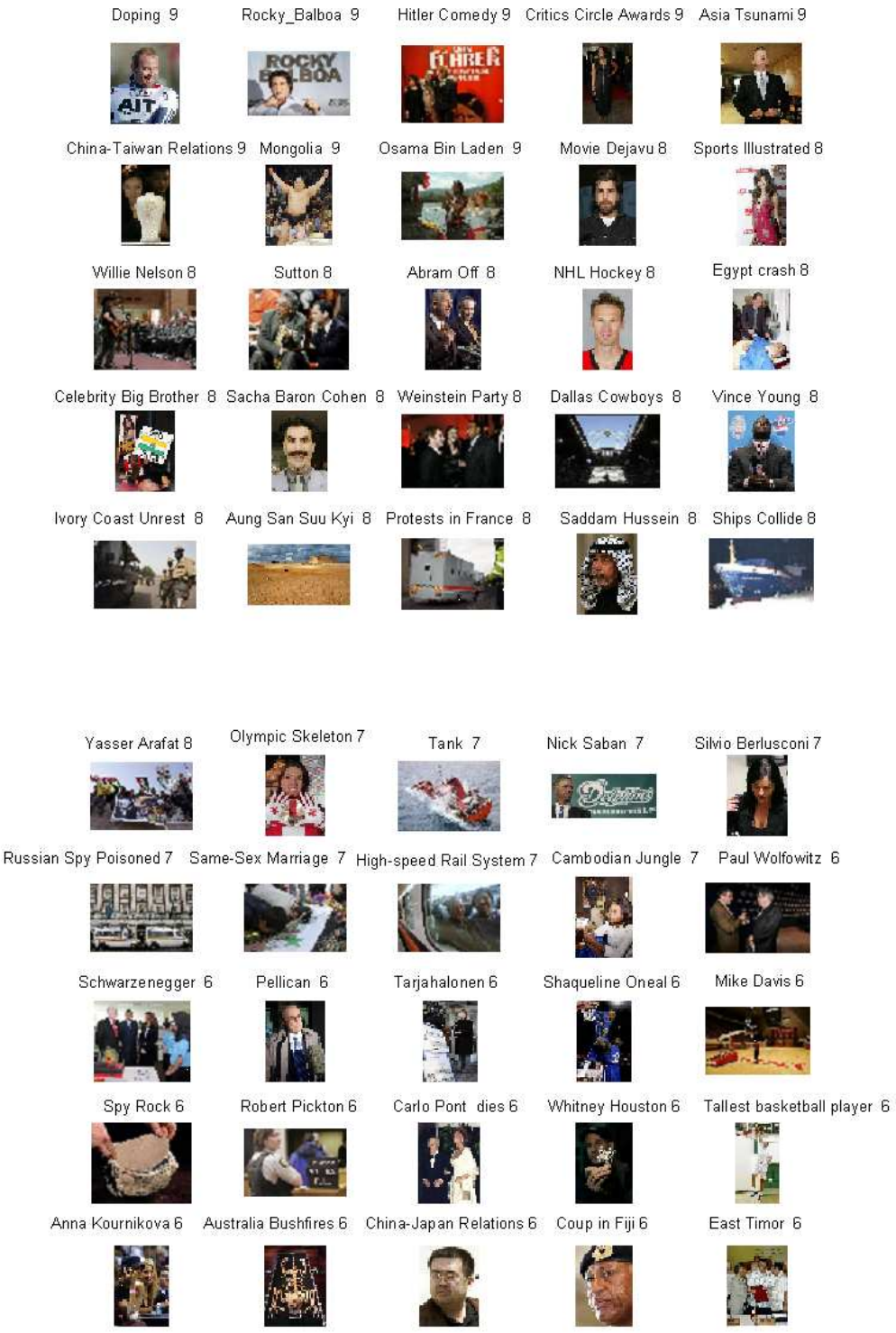

Figure 4-11: This figure shows topics ranked 196-240 in frequency. The number next to the topic label indicates the number of images for that topic.

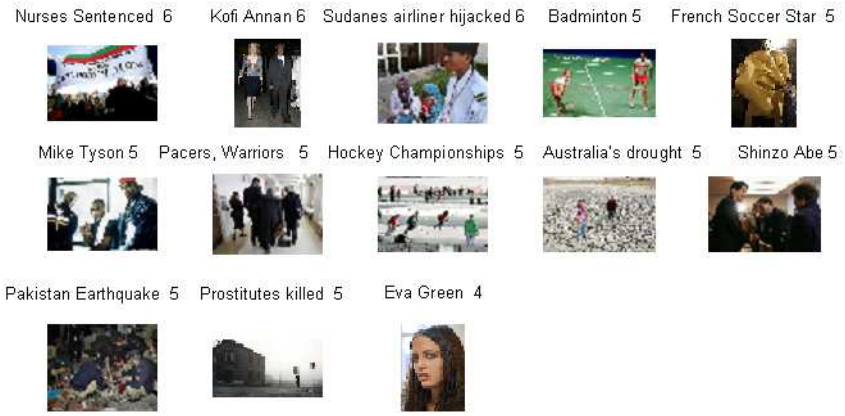

Figure 4-12: This figure shows topics ranked 241-248 in frequency. The number next to the topic label indicates the number of images for that topic.

Figure 4-13: Example images with their corresponding captions.

Figure 4-14: This figure shows examples of images that were assigned a given annotation word (i.e. the given word appeared in its corresponding caption), for words: team, president and actress.

$n = \{1, 2, 4, 8, 16, 32, 64\}$ and $y = \{1, 2, 3, \ldots, 15\}$, where $\mathcal{T}_{n,y}$ denotes a training set for topic y which has n positive examples from topic y, and $4n$ negative examples. The 15 topics corresponded to the 15 most frequent topics in the training data. The positive and negative examples were drawn randomly from the training set of size $8,000$. We will compare various models by training them on each of the training sets $\mathcal{T}_{n,y}$, and evaluating the models on the 1,382 test images.

In addition, each of the $8,000$ training images had associated captions, which can be used to derive an image representation (see section 4.2.1). Note that we make no use of captions on the test or development data sets. Instead, we will use the $8,000$ training images to derive representations that are input to a classifier that uses images alone.

In summary, our experimental set-up corresponds to a scenario where we have a small amount of labeled data for a *target* task (predicting the topic for an image), and a large amount of unlabeled data with associated captions.

4.6 Image Representation

In this chapter we will use $\mathbf{x}$ to denote a feature-vector representation that is based on SIFT features for an image x. A SIFT detector (Mutch and Lowe, 2006) is run over the image x, identifying a set of image patches. This detection step is run over the entire set of 10,382 images. k-means clustering is then performed on the image patches, using a SIFT representation of patches as input to the clustering step. In our experiments we chose the number of clusters k to be $1,000$. Each image is then represented by a feature vector $\mathbf{x} \in \mathbb{R}^{1000}$, where the j'th component of the feature vector $\mathbf{x}_j$ is the number of times a SIFT patch in the j'th cluster is seen in x. Thus $\mathbf{x}$ essentially represents a histogram of counts of each cluster. This representation has been used in several other image classification problems; we will refer to it as a SIFT histogram (SH) representation.

4.6.1 The Baseline Model

A baseline model was trained on all training sets $\mathcal{T}_{n,y}$. In each case the resulting model was tested on the $1,352$ test examples. The baseline model consists of a logistic regression

model over the SIFT features: to train the model we used conjugate gradient descent to find the parameters $\mathbf{w}^*$ which maximize the regularized log-likelihood, see equations 4.1 and 4.2. When calculating equal-error-rate statistics on test data, the value for $P(y = +1|x; \mathbf{w}^*)$ can be calculated for each test image x; this score is then used to rank the test examples.

The parameter γ in Eq. 4.1 dictates the amount of regularization used in the model. For the baseline model, we used the development set of $1,000$ examples to optimize the value of γ for each training set $\mathcal{T}_{n,y}$. Note that this will in practice give an upper bound on the performance of the baseline model, as assuming $1,000$ development examples is almost certainly unrealistic (particulary considering that we are considering training sets whose size is at most 320). The values of γ that were tested were 10^k, for $k = -5, -4, \ldots, 4$.

4.6.2 The Data-SVD Model

As a second baseline, we trained a logistic-regression classifier, but with the original feature vectors $\mathbf{x}$ in training and test data replaced by h-dimensional feature vectors $v(x) = \theta\mathbf{x}$ where θ was derived using PCA. A matrix $\mathbf{F}$ of dimension $1,000 \times 8,000$ was formed by taking the feature vectors $\mathbf{x}$ for the $8,000$ data points; the projection matrix θ was constructed from the first h eigenvectors of $\mathbf{FF}'$. The PCA model has free parameters z and γ. These were optimized using the method described in section 4.6.5. We call this model the *data-SVD* model.

4.6.3 A Model with Predictive Structure

We ran experiments using the structure prediction approach described in section 4.2. We train a logistic-regression classifier on feature vectors $v(x) = \theta\mathbf{x}$ where θ is derived using the method in section 4.2.1. Using the $8,000$ training images, we created 100 *auxiliary*training sets corresponding to the 100 most frequent content words in the captions.[3] Each training set involves prediction of a particular content word. The input to the classifier is the SIFT representation of an image. Next, we trained linear classifiers on each of the

[3]Content words are defined as any words which do not appear on a "stop" list of common function words.

100 *auxiliary* training sets to induce parameter vectors $\mathbf{w}_1 \ldots \mathbf{w}_{100}$. Each parameter vector is of dimension $1,000$; we will use $\mathbf{W}$ to refer to the matrix of size $1,000 \times 100$ which contains all parameter values. The projection matrix θ consists of the z eigenvectors in $\mathbb{R}^d$ which correspond to the z largest eigenvalues of $\mathbf{W}\mathbf{W}'$.

4.6.4 The Word-Classifiers Model

As a third baseline, we trained a logistic-regression classifier with the original feature vectors $\mathbf{x}$ in training and test data replaced by 100-dimensional feature vectors $v(x) = \theta\mathbf{x}$ where $\theta = \mathbf{W}^t$. This amounts to training the topic classifiers with the outputs of the word classifiers. Notice that this model is similar to a model with predictive structure where $z = 100$. The word-classifiers model has a free parameter γ. This was optimized using the method described in section 4.6.5. We call this model the *word-classifiers* model.

4.6.5 Cross-Validation of Parameters

The data-SVD and the predictive structure models have 2 free parameters: the dimensionality of the projection h, and γ.[4] A single topic—the 7th most frequent topic in the training data—was used to tune these parameters for the 3 model types. For each model type the model was trained on all training sets $T_{n,7}$ for $n = 1, 2, 4, 8, ..., 64$, with values for z taken from the set $\{2, 5, 10, 20, 30, 40, 100\}$ and values for γ chosen from $\{0.00001, 0.0001, \ldots, 1000\}$. Define $E^n_{z,\gamma}$ to be the equal-error-rate on the development set for topic 7, when trained on the training set $T_{n,7}$ using parameters z and γ. We choose the value z^* for all experiments on the remaining 14 topics as

$$z^* = \arg\min_z \sum_{i=1,2,...,64} \min_\gamma E^i_{z,\gamma}$$

This corresponds to making a choice of z^* that performs well on average across all training set sizes. In addition, when training a model on a training set with i positive examples, we chose $\gamma^*_i = \arg\min_\gamma E^i_{z^*,\gamma}$ as the regularization constant. The motivation for using a single

[4] We use γ to refer to the parameter γ_2 in figure 4-1. γ_1 in figure 4-1 was set to be 0.1 in all of our experiments.

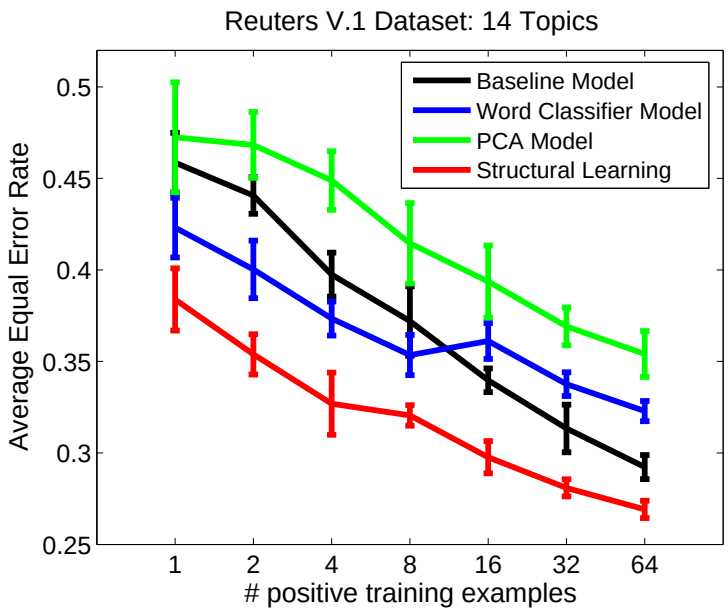

Figure 4-15: Equal error rate averaged across topics, with standard deviations calculated from ten runs for each topic. The equal error rates are averaged across 14 topics; the 7th most frequent topic is excluded as this was used for cross-validation (see section 4.6.5).

topic as a validation set is that it is realistic to assume that a fairly substantial validation set (1,000 examples in our case) can be created for one topic; this validation set can then be used to choose values of z^* and γ_i^* for all remaining topics.

The word-classifiers model has one free parameter: the constant γ used in Eq. 4.1. The value for this parameter was also optimized using the development set.

4.6.6 Results

Figure 4-15 shows the mean equal error rate and standard error over ten runs for the experiments on the Reuters dataset. The equal error rate is the recall occurring when the decision threshold of the classifier is set so that the proportion of false rejections will be equal to the proportion of false acceptances. For example an equal error rate of 70% means that when the proportion of false rejections is equal to the proportion of false acceptances 70% of the positive examples are labeled correctly and 30% of the negative examples are misclassified as positive.

For all training set sizes the *structure learning* learning model improves performance over the three other models. The average performance with one positive training example is around 62% with the *structure learning* learning method; to achieve similar performance with the word-classifiers model requires around four positive examples and with the baseline model between four and eight. Similarly, the performance with four positive examples for the *structure learning* learning method is around 67%; both the word-classifiers model and the baseline model require between 32 and 64 positive examples to achieve this performance. PCA's performance is lower than the baseline model for all training sizes and the gap between the two increases with the size of the training set. The performance of the word-classifiers model is better than the baseline for small training sets but the gap between the two decreases with the size of the training set.

Figures 4-16 and 4-17 show equal error rates for two different topics. The first topic, "Australian Open", is one of the topics that exhibits the most improvement from *structure learning* learning. The second topic, "Winter Olympics", is one of the three topics for which *structure learning* learning does not improve performance. As can be observed from the Australian Open curves the use of *structure learning* features speeds the generalization ability of the classifier. The *structure learning* model trained with only two positive examples performs comparably to the baseline model trained with sixty four examples. For the Winter Olympics topic the three models perform similarly. At least for a small number of training examples, this topic exhibits a slow learning curve; i.e. there is no significant improvement in performance as we increase the size of the labeled training set; this suggests that this is an inherently more difficult class.

4.7 Chapter Summary

Current methods for learning visual categories work well when a large amount of labeled data is available, but can run into severe difficulties when the number of labeled examples is small. When labeled data is scarce it may be beneficial to use unlabeled data to learn an image representation that is low-dimensional, but nevertheless captures the information required to discriminate between image categories.

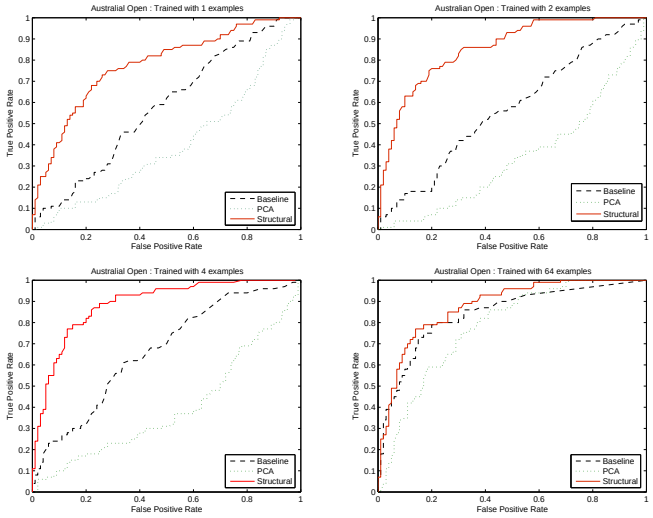

Figure 4-16: Roc Curves for the “Australian Open” topic.

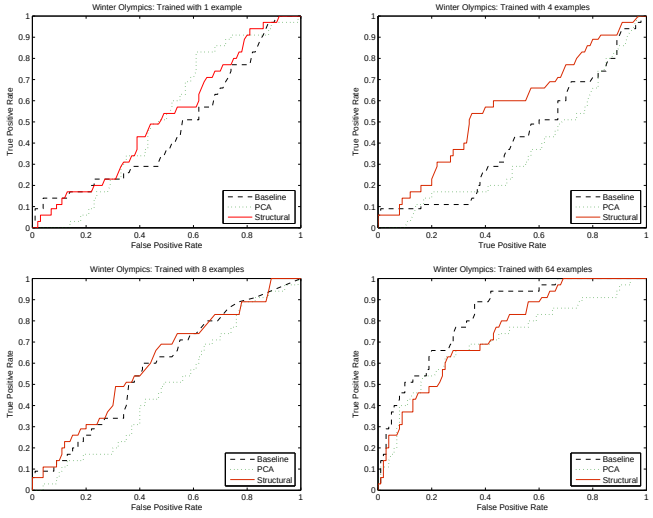

Figure 4-17: Roc Curves for the “Winter Olympics” topic.

This chapter described a method for learning representations from large quantities of unlabeled images which have associated captions; the goal is to improve learning in future image classification problems. The method makes use of *auxiliary* training sets corresponding to different words in the captions, and *structure learning* , which learns a manifold in parameter space.

Experiments show that our method significantly outperforms (1) a fully-supervised baseline model, (2) a model that ignores the captions and learns a visual representation by performing PCA on the unlabeled images alone and (3) a model that uses the output of word classifiers trained using captions and unlabeled data.

In brief, our results show that when meta-data labels are suitably related to a *target* task, the *structure learning* learning method can discover feature groupings that speed learning of the *target*task. Our current work concentrates on captions as the source of meta-data, but more generally other types of meta-data could be used.

Future work includes exploration of automatic determination of relevance between *target*and *auxiliary* tasks, and experimental evaluation of the effectiveness of *structure learning* from more weakly related *auxiliary* domains.

Chapter 5

Transfer Learning for Image Classification with Sparse Prototype Representations

The work presented in this Chapter was published in Quattoni et al. (2008).

Chapter 4 presented a semi-supervised application of a *structure learning* algorithm. In this chapter we follow a different approach and develop a *joint sparsity* transfer algorithm for image classification based on a regularization matrix norm derived from simultaneous sparse signal approximation (Tropp, 2006a). This norm is designed to promote joint sparsity across tasks. While chapter 4 presented a semi-supervised application of a transfer algorithm, we now turn our focus to an *asymmetric transfer* setting.

The remainder sections are organized as follows: In Section 5.1 we give the motivation for the work presented in this chapter, Section 5.2 describes the *joint sparsity* transfer algorithm, Section 5.3 presents experiments on an *asymmetric transfer* image classification task and finally Section 5.4 provides a summary.

5.1 Introduction

In this chapter we develop a *joint sparsity* transfer algorithm that can learn an efficient representation from a set of related tasks and which explicitly takes advantage of unlabeled

data. Our method uses unlabeled data to define a prototype representation based on computing kernel distances to a potentially large set of unlabeled points. However, to train an image classifier in a given domain we might only need to consider distances to a small set of prototypes; if these prototypes were known, we might be able to learn with fewer examples by removing irrelevant prototypes from the feature space.

In general we will not know this a priori, but related tasks may share a significant number of such prototypes. Our transfer algorithm identifies the set of prototypes which are jointly most relevant for a given set of tasks, and uses that reduced set of points as the feature space for future related tasks. Our experiments show that using the transferred representation significantly improves learning with small training sets when compared to the original feature space.

One of the advantages of our transfer learning method is that the prototype representation is defined using an arbitrary kernel function (Balcan et al., 2004). Recent progress has shown that visual category recognition can improve with the use of kernels that are optimized to particular tasks (Varma and Ray, 2007).

We discover an optimal subset of relevant prototypes with a jointly regularized optimization that minimizes the total number of reference prototypes involved in the approximation. Our joint regularization exploits a norm derived from simultaneous sparse signal approximation methods (Tropp, 2006a), leading to an optimization problem that can be expressed as a linear program.

Our approach builds on the work of Balcan et al. (2004), who proposed the use of a representation based on kernel distances to unlabeled datapoints, and the work of Tropp (2006a) on simultaneous sparse approximation. The latter problem involves approximating a set of signals by a linear combination of elementary signals while at the same time minimizing the total number of elementary signals involved in the approximation. Tropp proposed a joint optimization with a shared regularization norm over the coefficients of each signal and gave theoretical guarantees for this approach.

We evaluate our method on a news-topic prediction task, where the goal is to predict whether an image belongs to a particular news-topic. Our results show that learning a representation from previous topics does improve future performance when supervised training

data are limited.

5.2 Learning a sparse prototype representation from unlabeled data and related tasks

We now describe our sparse prototype learning algorithm for learning a representation from a set of unlabeled data points $\mathcal{U} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\}$ and a collection of training sets of related problems $\mathcal{D} = \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$, where $\mathcal{T}_k = \{(\mathbf{x}_1^k, y_1^k), \ldots, (\mathbf{x}_{n_k}^k, y_{n_k}^k)\}$. In all cases, $\mathbf{x} \in \mathcal{X}$ (for example, $\mathcal{X} = \mathbb{R}^d$ and $y \in \{+1, -1\}$.

In the following subsections we describe the three main steps of our algorithm. In the first step, we compute a prototype representation using the unlabeled dataset. In the second step, we use data from related problems to select a small subset of prototypes that is most discriminative for the related problems. Finally, in the third step we create a new representation based on kernel distances to the the selected prototypes. We will use the sparse prototype representation to train a classifier for a target problem. Algorithm 1 provides pseudo-code for the algorithm. The next three sub-sections describe the three steps of the algorithm in detail.

5.2.1 Computing the prototype representation

Given an unlabeled data set $\mathcal{U} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\}$ and a kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, the first step of our algorithm computes a prototype representation based on kernel distances to the unlabeled data points in $\mathcal{U}$.

We create the prototype representation by first computing the kernel matrix $\mathcal{K}$ of all points in $\mathcal{U}$, i.e. $\mathcal{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathcal{U}$. We then create a projection matrix θ formed by taking all the eigenvectors of $\mathcal{K}$ corresponding to non-zero eigenvalues (the eigenvectors are obtained by performing SVD on $\mathcal{K}$). The new representation is then given by:

$$z(\mathbf{x}) = \theta^\top \varphi(\mathbf{x}), \tag{5.1}$$

Input 1: Unlabeled dataset $\mathcal{U} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\}$ for $\mathbf{x}_i \in \mathcal{X}$ (e.g, $\mathcal{X} = \mathbb{R}^d$)

Input 2: Collection of related problems: $\mathcal{D} = \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$

where $\mathcal{T}_k = \{(\mathbf{x}_1^k, y_1^k), (\mathbf{x}_2^k, y_2^k), \ldots, (\mathbf{x}_{n_k}^k, y_{n_k}^k)\}$ for $\mathbf{x} \in \mathcal{X}$ and $y \in \{+1, -1\}$

Input 3: Kernel function: $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

Input 4: Threshold ρ

Input 5: Regularization constants λ_k, for $k = 1 : m$

Step 1: Compute the prototype representation

- Compute the kernel matrix for all unlabeled points :

  $\mathcal{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for $\mathbf{x}_i \in \mathcal{U}$, $\mathbf{x}_j \in U$

- Compute eigenvectors of of $\mathcal{K}$ by performing SVD :

  Compute a projection matrix θ of dimension $q \times q$ by taking the eigenvectors of $\mathcal{K}$; where each column of θ corresponds to an eigenvector.

- Project all points $\mathbf{x}_i^k$ in $\mathcal{D}$ to the prototype space: $z(\mathbf{x}_i^k) = \theta^\top \varphi(\mathbf{x}_i^k)$

  where $\varphi(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_q)]^\top$, $\mathbf{x}_i \in \mathcal{U}$

Step 2: Discover relevant prototypes by joint sparse approximation

Let W be a $q \times m$ matrix where $W_{j,k}$ corresponds to the $j\text{-}th$ coefficient of the $k\text{-}th$ problem.

- Choose the optimal matrix W^* to be:

  $\min_{W,\boldsymbol{\varepsilon}} \ \sum_{k=1}^m \lambda_k \sum_{i=1}^{n_k} \varepsilon_i^k + \sum_{j=1}^q \max_k |W_{j,k}|$

  s.t. for $k = 1 : m$ and $i = i : n_k$

  $y_i^k \mathbf{w}_k^\top z(\mathbf{x}_i^k) \geq 1 - \varepsilon_i^k \quad \varepsilon_i^k \geq 0$

  where $\mathbf{w}_k$ is the $k\text{-}th$ column of W, corresponding to the parameters for problem k.

Step 3: Compute the relevant prototype representation

- Define the set of relevant prototypes to be: $\mathcal{R} = \{r : \max_k |W_{rk}^*| > \rho\}$

- Create projection matrix B by taking all the columns of θ corresponding to the indexes in $\mathcal{R}$. B is then a $q \times h$ matrix, where $h = |\mathcal{R}|$.

- Return the representation given by: $v(\mathbf{x}) = B^\top \varphi(\mathbf{x})$

Output: The function $v(\mathbf{x}) : \mathbb{R}^q \to \mathbb{R}^h$

Algorithm 1: The sparse prototype representation learning algorithm.

where $\varphi(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, x_q)]^\top$, and $\mathbf{x}_i \in \mathcal{U}$. We will refer to the columns of θ as prototypes.

This representation was first introduced by Balcan et al., who proved it has important theoretical properties. In particular, given a target binary classification problem and a kernel function, they showed that if the classes can be separated with a large margin in the induced kernel space then they can be separated with a similar margin in the prototype space. In other words, the expressiveness of the prototype representation is similar to that of the induced kernel space. By means of this technique, our joint sparse optimization can take the advantage of a kernel function without having to be explicitly kernelized.

Another possible method for learning a representation from the unlabeled data would be to create a $q \times h$ projection matrix L by taking the top h eigenvectors of $\mathcal{K}$ and defining the new representation $g(x) = L^\top \varphi(\mathbf{x})$; we call this approach the low rank technique. The method we develop in this chapters differs significantly from the low rank approach in that we use training data from related problems to select discriminative prototypes, as we describe in the next step. In the experimental Section we show the advantage of our approach compared to the low rank method.

5.2.2 Discovering relevant prototypes by joint sparse approximation

In the second step of our algorithm we use a collection of training sets from related problems, $\mathcal{D} = \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$, where $\mathcal{T}_k = \{(\mathbf{x}_1^k, y_1^k), \ldots, (\mathbf{x}_{n_k}^k, y_{n_k}^k)\}$, to find a subset of discriminative prototypes. Our method is based on the search for a sparse representation in the prototype space that is jointly optimal for all problems in $\mathcal{D}$.

Consider first the case of learning a single sparse classifier on the prototype space of the form:

$$f(\mathbf{x}) = \mathbf{w}^\top z(\mathbf{x}), \tag{5.2}$$

where $z(\mathbf{x}) = \theta^\top \varphi(\mathbf{x})$ is the representation described in step 1 of the algorithm. A sparse model will have a small number of prototypes with non-zero coefficients. Given a training set with n examples, a natural choice for parameter estimation in such a setting

would be to take the optimal parameters $\mathbf{w}^*$ to be:

$$\min_{\mathbf{w}} \lambda \sum_{i=1}^{n} \mathsf{Loss}(f(\mathbf{x}_i), y_i) + \sum_{j=1}^{q} |w_j|. \tag{5.3}$$

The left term of Equation (5.3) measures the error that the classifier incurs on training examples, measured in terms of a loss function Loss. In this chapter we will use the hinge loss, given by $\mathsf{Loss}(f(x), y) = \max(0, (1 - yf(x)))$.

The right hand term of Equation (5.3) is an l_1 norm on the coefficient vector which promotes sparsity. In the context of regression Donoho (2004) has proven that the solution with smallest l_1 norm is also the sparsest solution, i.e. the solution with the least number of non-zero coefficients. The constant λ dictates the trade off between sparsity and approximation error on the data.

For transfer learning, our goal is to find the most discriminative prototypes for the problems in $\mathcal{D}$, i.e. find a subset of prototypes $\mathcal{R}$ such that each problem in $\mathcal{D}$ can be well approximated by a sparse function whose non-zero coefficients correspond to prototypes in $\mathcal{R}$. Analogous to the single sparse approximation problem, we will learn jointly sparse classifiers on the prototype space using the training sets in $\mathcal{D}$. The resulting classifiers will share a significant number of non-zero coefficients, i.e active prototypes. Let us define a $q \times m$ coefficient matrix W, where W_{jk} corresponds to the $j\text{-}th$ coefficient of the $k\text{-}th$ problem. In this matrix, the $k\text{-}th$ column of W is the set of coefficients for problem k, which we will refer to as $\mathbf{w}_k$, while the $j\text{-}th$ row corresponds to the coefficients of prototype j across the k problems. The m classifiers represented in this matrix correspond to:

$$f_k(\mathbf{x}) = \mathbf{w}_k^\top z(\mathbf{x}). \tag{5.4}$$

It is easy to see that the number of non-zero rows of W corresponds to the total number of prototypes used by any of the m classifiers. This suggests that a natural way of posing the joint sparse optimization problem would be to choose the optimal coefficient matrix W^* to be:

$$\min_{W} \sum_{k=1}^{m} \lambda_k \sum_{i=1}^{n_k} \mathsf{Loss}(y_i^k, f_k(\mathbf{x}_i^k)) + l_{r0}(W) \tag{5.5}$$

where:

$$l_{r0}(W) = |\{j = 1 : d | max_k(w_{j,k}) \neq 0\}| \tag{5.6}$$

is a pseudo-norm that counts the number of non-zero rows in W [1].

As in the single sparse approximation problem, the two terms in Equation (5.5) balance the approximation error against some notion of sparsity. Here, the left hand term minimizes a weighted sum of the losses incurred by each classifier on their corresponding training sets, where λ_k weights the loss for the $k\text{-}th$ problem. The right hand term minimizes the number of prototypes that have a non-zero coefficient for some of the related problems. Due to the presence of the l_{r0} pseudo-norm in the objective, solving (5.5) might result in a hard combinatorial problem. Instead of solving it directly we use a convex relaxation of the l_{r0} pseudo-norm suggested in the literature of simultaneous sparse approximation (Tropp, 2006a), the $l_{1,\infty}$ norm, which takes the following form:

$$\sum_{j=1}^{q} \max_k |W_{dk}| \tag{5.7}$$

Using the $l_{1,\infty}$ norm we can rewrite Equation (5.5) as:

$$\min_W \sum_{k=1}^{m} \lambda_k \sum_{i=1}^{n_m} \mathsf{Loss}(y_i^k, f_k(\mathbf{x}_i^k)) + \sum_{i=1}^{q} \max_k |W_{jk}| \tag{5.8}$$

The right hand term on Equation (5.8) promotes joint sparsity by combining an l_1 norm and an l_∞ norm on the coefficient matrix. The l_1 norm operates on a vector formed by the maximum absolute values of the coefficients of each prototype across problems, encouraging most of these values to be 0. On the other hand, the l_∞ norm on each row promotes non-sparsity among the coefficients of a prototype. As long as the maximum absolute value of a row is not affected, no penalty is incurred for increasing the value of a row's coefficient. As a consequence only a few prototypes will be involved in the approximation but the prototypes involved will contribute in solving as many problems as possible.

When using the hinge loss the optimization problem in Equation (5.8) can be formulated as a linear program:

[1] The number of non-zero rows is the number of rows for which at least one of its elements is different than 0.

$$\min_{W,\boldsymbol{\varepsilon},\mathbf{t}} \sum_{k=1}^{m} \lambda_k \sum_{i=1}^{n_k} \varepsilon_i^k + \sum_{j=1}^{q} t_j \tag{5.9}$$

such that for $j = 1 : q$ and $k = 1 : m$

$$-t_j \leq W_{jk} \leq t_j \tag{5.10}$$

and for $k = 1 : m$ and $i = 1 : n_k$

$$y_i^k \mathbf{w}_k^\top z(\mathbf{x}_i^k) \geq 1 - \varepsilon_i^k \tag{5.11}$$

$$\varepsilon_i^k \geq 0 \tag{5.12}$$

The constraints in Equation (5.10) bound the coefficients for the $j\text{-}th$ prototype across the m problems to lie in the range $[-t_j, t_j]$. The constraints in Equations (5.11) and (5.12) impose the standard slack variable constraints on the training samples of each problem.

5.2.3 Computing the relevant prototype representation

In the last step of our algorithm we take the optimal coefficient matrix W^* computed in Equation (5.9) of Step 2 and a threshold ρ and create a new representation based on kernel distances to a subset of relevant prototypes. We define the set of relevant prototypes to be:

$$\mathcal{R} = \{r : \max_k |W_{rk}^*| > \rho\}. \tag{5.13}$$

We construct a new projection matrix B by taking all the columns of θ corresponding to indices in $\mathcal{R}$, where θ is the matrix computed in the first step of our algorithm. B is then a $q \times h$ matrix, where $h = |\mathcal{R}|$. The new sparse prototype representation is given by:

$$v(\mathbf{x}) = B^\top \varphi(\mathbf{x}). \tag{5.14}$$

When given a new target problem we project every example in the training and test set using $v(\mathbf{x})$. We could potentially train any type of classifier on the new space; in our experiments we chose to train a linear SVM.

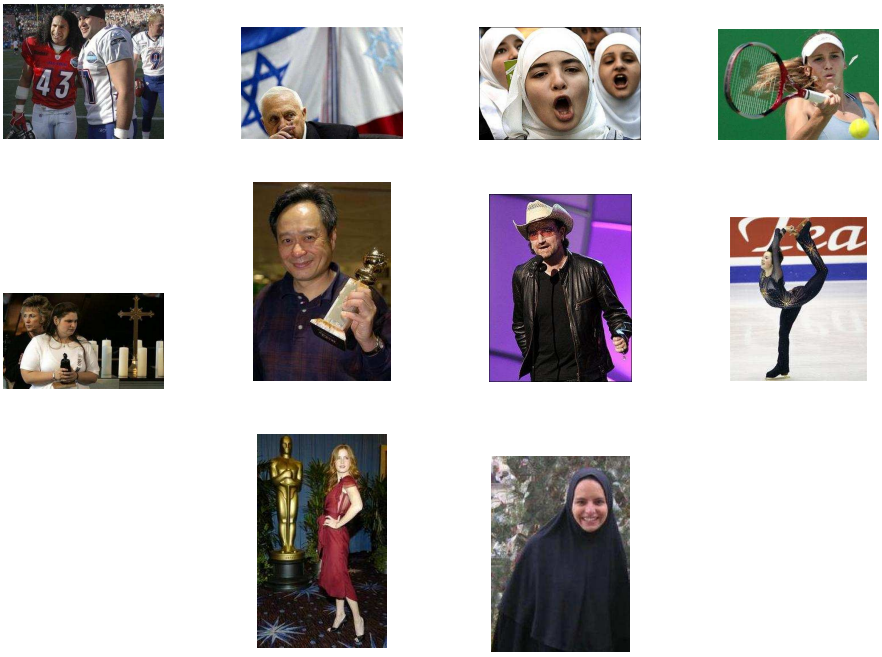

Figure 5-1: Example images. From top to bottom, left to right: SuperBowl, Sharon, Danish Cartoons, Australian Open, Trapped Coal Miners, Golden Globes, Grammys, Figure Skating, Academy Awards and Iraq.

5.3 Experiments

For these section we used the dataset Reuters.V1 described in Section 4.4. Around 40 percent of the images belonged to one of the ten most most frequent topics, which we used as the basis for our experiments: SuperBowl, Golden Globes, Danish Cartoons, Grammys, Australian Open, Ariel Sharon, Trapped Coal Miners, Figure Skating, Academy Awards and Iraq. Figure 5-1 shows some example images from each of these categories.

The experiments involved the binary prediction of whether an image belonged to a particular news topic. The data was partitioned into unlabeled, training, validation and testing sets: we reserved 2,000 images as a source of unlabeled data, 1,000 images as potential source of validation data and 5,000 images as a source of supervised training data. The remaining 2,382 images where used for testing. For each of the 10 most frequent topics we created multiple training sets of different sizes $\mathcal{T}_{y,n}$ for $n = 1, 5, 10, 15, \ldots, 50$

and $y = 1, 2, \ldots, 10$; where $\mathcal{T}_{y,n}$ denotes a training set for topic y which has n positive examples from topic y and $4n$ negative examples. The positive and negative examples were drawn at random from the pool of supervised training data of size 5,000. The total number of positive images in this pool for the ten top topics were: 341, 321, 178, 209, 196, 167, 170, 146, 137 and 125 respectively.

We consider a transfer learning setting where we have access to unlabeled data $\mathcal{U}$ and a collection of training sets $\mathcal{D}$ from m related tasks. Our goal is to learn a representation from $\mathcal{D}$ and $\mathcal{U}$ and use it to train a classifier for the *target* task.

In our experimental setting we took the 10 most frequent topics and held out one of them to be the target task; the other nine topics were used to learn the sparse prototype representation. We did this for each of the 10 topics in turn. The training set for related topic j in the collection of related training sets $\mathcal{D}$ was created by sampling all n_j positive examples of topic j and $2n_j$ negative examples from the pool of supervised training data.

We test all models using the 2,382 held out test images but we remove images belonging to topics in $\mathcal{D}$. We did this to ensure that the improvement in performance was not just the direct consequence of better recognition of the topics in $\mathcal{D}$; in practice we observed that this did not make a significant difference to the experimental results.

Our notion of relatedness assumes that there is a small subset of relevant prototypes such that all related problems can be well approximated using elements of that subset; the size of the relevant subset defines the strength of the relationship. In this experiment we deal with problems that are intuitively only weakly related and yet we can observe a significant improvement in performance when only few examples are available for training. In the future we plan to investigate ways of selecting sets of more strongly related problems.

5.3.1 Baseline Representation

For all the experiments we used an image representation based on a bag of words representation that combined color, texture and raw local image information. For every image in the dataset we sampled image patches on a fixed grid and computed three types of features for each image patch: color features based on HSV histograms, texture features consisting

of mean responses of Gabor filters at different scales and orientations and 'raw' features made by normalized pixel values. For each feature type we created a visual dictionary by performing vector quantization on the sampled image patches; each dictionary contained 2,000 visual words.

To compute the representation for a new image $\mathbf{x}_i$ we sample patches on a fixed grid to obtain a set of patches $p_i = \{x_{i1}, x_{i2}, \ldots, x_{ih}\}$ and then match each patch to its closest entry in the corresponding dictionary. The final baseline representation for an image is given by the 6,000 dimensional vector: $[[cw_1, \ldots, cw_{2000}], [tw_1, \ldots, tw_{2000}], [rw_1, \ldots, rw_{2000}]]$ where cw_i is the number of times that an image patch in p_i was matched to the i-th color word, tw_i the number of times that an image patch in p_i was matched to the i-th texture word and rw_i the number of times that an image patch in p_i was matched to the i-th raw word.

5.3.2 Raw Feature Baseline Model

The raw feature baseline model (RFB) consists of training a linear SVM classifier over the bag of words representation by choosing the optimal parameters to be:

$$\mathbf{w}^* = \min_{\mathbf{w}} \lambda \sum_i \mathsf{Loss}(f(\mathbf{x}_i), y_i) + \frac{1}{2}\|\mathbf{w}\|_2^2 \tag{5.15}$$

where $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ and Loss is the hinge loss described in Section 5.2.2.

We conducted preliminary experiments using the validation data from topic 1 and found 0.01 to be the parameter λ resulting in best equal error rate for all training sizes (where we tried values: $\{0.01, 0.1, 1, 10, 100\}$); we also noticed that for the validation set the baseline model was not very sensitive to this parameter. We set the constant λ for this and all other models to 0.01.

A baseline model was trained on all training sets $\mathcal{T}_{y,n}$ of the 10 most frequent topics and tested on the 2,382 test images. As explained in the previous Section, we removed from the testing set images that belonged to any of the other nine most frequent topics.

5.3.3 Low Rank Baseline Model

As a second baseline (LRB), we trained a linear SVM classifier but with the baseline feature vectors $\mathbf{x}$ in training and testing replaced by h-dimensional feature vectors: $g(\mathbf{x}) = L^\top \varphi(\mathbf{x})$. L is the matrix described in Section 5.2.1 created by taking the top h eigenvectors of $\mathcal{K}$, where $\mathcal{K}$ is the kernel matrix over unlabeled data points.

We present results for different values of $h = \{50, 100, 200\}$. For all experiments in this Section we used an RBF kernel over the bag of words representation: $k(\mathbf{x}_i, \mathbf{x}_j) = exp^{-\beta||x_i - x_j||_2^2}$. In a preliminary stage, we tried a range of values $\beta = \{0.003, 0.03, 0.3\}$ on the unlabeled data, 0.03 was the value that resulted in a non-degenerate kernel (i.e. neither a diagonal kernel nor a kernel made of ones). The value of β was then fixed to 0.03 for all experiments.

5.3.4 The Sparse Prototype Transfer Model

We ran experiments using the sparse prototype transfer learning (SPT) approach described in Section 5.2.3 . For each of the ten topics we train a linear SVM on feature vectors $v(\mathbf{x})$ obtained by running the sparse prototype transfer learning algorithm on the training sets of the remaining nine topics.

For a target held out topic j we use the 2,000 unlabeled points in $\mathcal{U}$ and a collection of training sets from related problems: $\mathcal{D} = \{\mathcal{T}_1, \ldots, \mathcal{T}_{j-1}, \mathcal{T}_{j+1}, \ldots\}$ to compute the sparse prototype representation $v(\mathbf{x})$ based on kernel distances to relevant prototypes. We report results for different values of the threshold ρ; in practice ρ could be validated using leave-one-out cross-validation.

5.3.5 Results

For all experiments we report the mean equal error rate and the standard error of the mean. Both measures were computed over nine runs of the experiments, where each run consisted of randomly selecting a training set from the pool of supervised training data.

The equal error rate is 1-recall occurring when the decision threshold of the classifier is set so that the the proportion of false rejections will be equal to the proportion of false

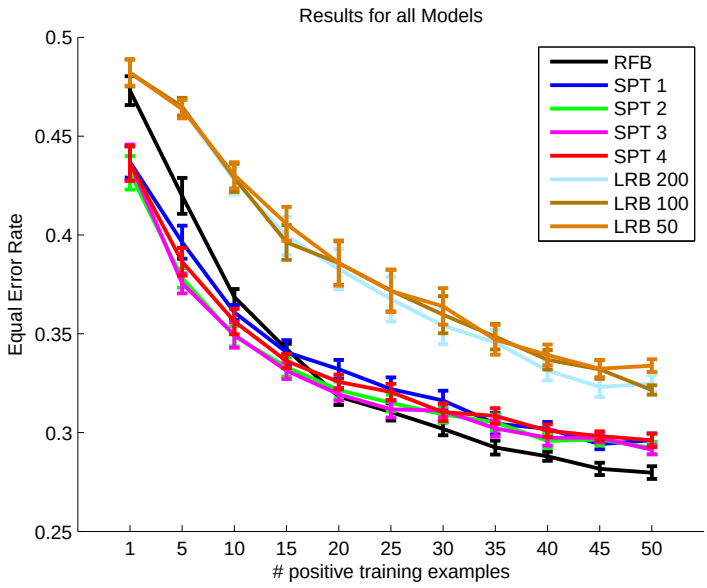

Figure 5-2: Mean Equal Error Rate over 10 topics for RFB, LRB (for $h = \{50, 100, 200\}$, see section 5.2.1 for details) and SPT (for $\rho = \{1, 2, 3, 4\}$, see section 5.2.3 for details).

acceptances. For example an equal error rate of 30 percent means that when the proportion of false rejections is equal to the proportion of false acceptances 70 percent of the positive examples are labeled correctly and 30 percent of the negative examples are misclassified as positive.

Figure 5-2 shows the mean equal error rate averaged over the ten most frequent topics for RFB, LRB and SPT models; as we can observe from this figure the low rank approach fails to produce a useful representation for all choices of h. In contrast, our sparse transfer approach produces a representation that is useful when training classifiers with small number of training examples (i.e. less than 10 positive examples); the improvement is most significant for $\rho \geq 3$.

For larger training sets the RFB baseline gives on average better performance than the SPT model. We speculate that when larger training sets are available the sparse representation needs to be combined with the raw feature representation. In the future we plan to investigate methods for fusing both representations. However, we would like to emphasize that there exist important applications for which only very few examples might be available

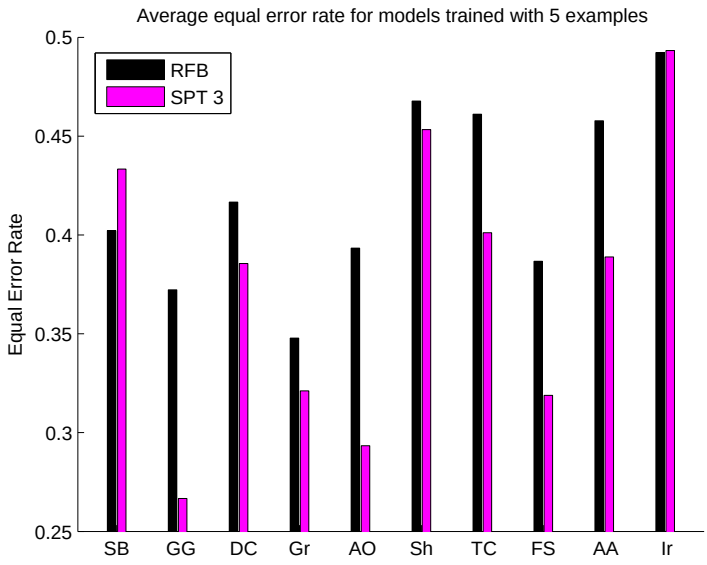

Figure 5-3: Mean Equal Error rate per topic for classifiers trained with five positive examples; for the RFB model and the SPT model for $\theta = 3$ (see Section 5.2.3 for details). SB: SuperBowl; GG: Golden Globes; DC: Danish Cartoons; Gr: Grammys; AO: Australian Open; Sh: Sharon; FS: Figure Skating; AA: Academy Awards; Ir: Iraq.

for training. For example when building a classifier for a user-defined category it would be unrealistic to expect the user to provide more than a handful of positive examples.

Figure 5-3 shows mean equal error rates for each topic when trained with 5 positive examples for the baseline model and the transfer learning model with $\rho = 3$. As we can see from these figures the sparse prototype transfer learning method significantly improves performance for 8 out of the 10 topics.

Figure 5-4 shows learning curves for the baseline and sparse transfer learning model for three different topics. The first topic, Golden Globes, is one of the topics that has the most improvement from transfer learning, exhibiting significantly better performance across all training sizes. The second topic, Academy Awards, shows a typical learning curve for the sparse prototype transfer learning algorithm; where we observe a significant improvement in performance when a few examples are available for training. Finally the third topic, Super Bowl, is the topic for which the sparse prototype transfer algorithm

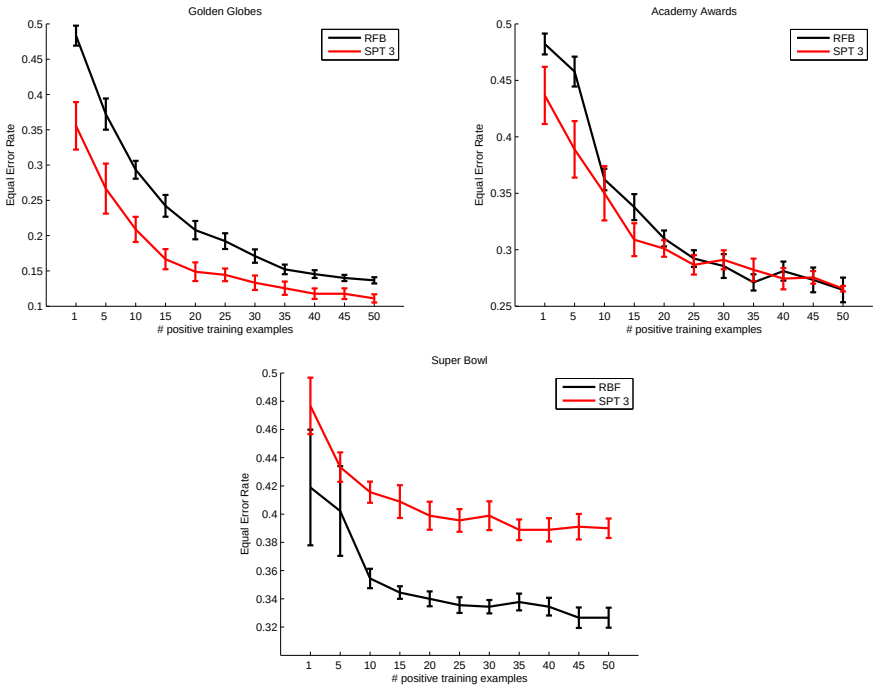

Figure 5-4: Learning curves for Golden Globes, Academy Awards and Super Bowl topics respectively for RFB and SPT model with $\theta = 3$, see Section 5.2.3 for details about parameter θ.

results in worst performance. We speculate that this topic might not be visually related to any of the other topics used for transfer. We have also noticed that this is one of the most visually heterogeneous topics since it contains images from the football field, a press conference and after-game celebrations.

5.4 Chapter Summary

To learn a new visual category from few examples, prior knowledge from unlabeled data as well as previous related categories may be useful. In this chapter wepresented a transfer learning algorithm which exploits available unlabeled data and an arbitrary kernel function by first forming a representation based on kernel distances to a large set of unlabeled data points. To transfer knowledge from previous related problems we observe that a category might be learnable using only a small subset of reference prototypes.

Related problems may share a significant number of relevant prototypes; we find such a concise representation by performing a joint loss minimization over the training sets of related problems with a shared regularization penalty that minimizes the total number of prototypes involved in the approximation.

This optimization problem can be formulated as a linear program that can be solved with an off-the-shelf package. We conduct experiments on a news-topic prediction task where the goal is to predict whether an image belongs to a particular news topic. Our results show that when only few examples are available for training a target topic, leveraging knowledge learnt from other topics can significantly improve performance.

The LP formulation presented in this chapter has two main limitations. The first limitations is that while it is feasible for small problems it does not scale (both in terms of time and memory) to problems with large number of variables[2]. The second limitation is that the linear program formulation can not be easily extended to handle other losses. Chapter 6 addresses these problems by developing a general and efficient algorithm for $l_{1,\infty}$ regularization.

[2]To give a concrete example, for a problem with 100,000 training samples and 6,000 dimensions the number of non-zero elements of the LP matrix would be around 10^9.

Chapter 6

An Efficient Projection for $l_{1,\infty}$ Regularization

The work presented in this chapter was published in Quattoni et al. (2009).

The LP formulation presented in chapter 5 had the limitation that it could not scale (both in terms of time and memory) to problems with large number of examples and dimensions. To give a concrete example, for a problem with: 100 tasks, 100,000 training samples and 6,000 dimensions we will have a total of $a = 700,000$ variables and $b = 1,300,000$ constraints, and an LP matrix with around 10^9 non-zero entries.

To illustrate the scalability of general LP solvers, Figure 6-1 shows time performance as a function of the number of non-zero entries of the LP matrix for a standard benchmark [1]. The results shown correspond to the best commercial solver for this benchmark. As we can see from this figure it is very difficult to use a general LP solver for large-scale problems. Another important limitation of general LP solvers is that their memory requirements are in the order of $O(min(a,b)^2)$ making them impractical for problems involving large number of variables and constraints (Shalev-Shwartz et al., 2007).

In addition, the LP formulation is specific to the hinge loss and cannot be easily extended to handle arbitrary loss functions. In this chapter we address these limitations and develop a general and efficient algorithm for $l_{1,\infty}$ regularization based on formulating the problem as a constrained convex optimization problem for which we derive a projected

[1] http://plato.asu.edu/bench.html

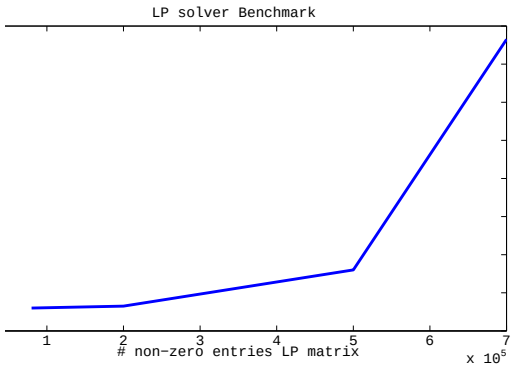

Figure 6-1: Time in cpu seconds as a function of the number of non-zero elements of the LP matrix, for the best commercial LP solver (CPLEX: http://www.cplex.com/) on a a set of standard benchmarks.

gradient method.

The remaining sections are organized as follows: Section 6.1 gives some motivation and frames our optimization algorithm in the larger context of projected gradient methods for minimizing large scale regularized objectives. Section 6.2 presents a general constrained convex optimization formulation for $l_{1,\infty}$ regularized models, Section 6.2.2 shows multitask learning as an instance of the general formulation, Section 6.2.3 reviews the projected gradient method which is the basis for our algorithm, Section 6.3 presents an algorithm to compute efficient projections to the $l_{1,\infty}$ ball, Section 6.4 reviews related optimization methods for $l_{1,\infty}$ regularization, Section 6.5 shows results of applying our algorithm to a synthetic dataset, Section 6.6 presents experiments on a *symmetric transfer* learning image annotation task and Section 6.7 presents results on a scene recognition dataset. Finally, Section 6.8 summarizes the results of this chapter.

6.1 Introduction

Learning algorithms based on l_1 regularized loss functions have had a relatively long history in machine learning, covering a wide range of applications such as sparse sensing (Donoho, 2004), l1-logistic regression (Ng, 2004), and structure learning of Markov networks (Lee

et al., 2006). A well known property of l_1 regularized models is their ability to recover sparse solutions. Because of this they are suitable for applications where discovering significant features is of value and where computing features is expensive. In addition, it has been shown that in some cases l_1 regularization can lead to sample complexity bounds that are logarithmic in the number of input dimensions, making it suitable for learning in high dimensional spaces (Ng, 2004).

Analogous to the use of the l_1 norm for single task sparse approximation, chapter 5 proposed the use of an $l_{1,\infty}$ norm for enforcing joint sparsity. Recall from chapter 5 that the $l_{1,\infty}$ norm is a matrix norm that penalizes the sum of maximum absolute values of each row.

As we have shown in the previous chapter, for a multitask application we can apply the $l_{1,\infty}$ regularization penalty to a parameter matrix $W = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$, where $\mathbf{w}_i$ is a parameter vector for the i-th task. In this case the $l_{1,\infty}$ regularizer is used to promote feature sharing across tasks and discover solutions where only a few features are non-zero in any of the m tasks (i.e. jointly sparse solutions). Other applications of $l_{1,\infty}$ regularization are simultaneous sparse signal approximation (Tropp, 2006a; Turlach et al., 2005) and structure learning of markov networks (Schmidt et al., 2008).

Finding jointly sparse solutions is important when we wish to control the run-time complexity of evaluating a set of classifiers. Consider a multitask problem with m tasks and a dual representation. At test time the cost of evaluating the m classifiers will be dominated by computing inner products between the test point and the support vectors of each task. Clearly, to control the run-time complexity we need to ensure that there will be a small number of shared support vectors.

In this chapter we present an efficient projected subgradient method for optimization of $l_{1,\infty}$ regularized convex objectives, which we formulate as a constrained convex optimization problem.

Projected gradient methods iterate between performing unconstrained gradient-based updates followed by projections to the feasible set, which in our case is an $l_{1,\infty}$ ball.

These methods have been shown to scale well and have been proposed for solving constrained optimization problems involving large numbers of variables. For example,

Shalev-Shwartz et al. (2007) developed a projected gradient method for l_2 regularization and Duchi et al. (2008) proposed an analogous algorithm for l_1. However, the problem of finding an efficient projected method for $l_{1,\infty}$ constraints remains open. The main challenge in developing a projected gradient algorithm for $l_{1,\infty}$ constraints resides on being able to efficiently compute Euclidean projections onto the $l_{1,\infty}$ ball. We show that this can be done in $O(n \log n)$ time and $O(n)$ memory, where n is the number of parameters of our model.

We apply our algorithm to a multitask image annotation problem where the goal is to predict keywords for a given image. We show that $l_{1,\infty}$ regularization performs significantly better than both independent l_2 and independent l_1 regularization. Furthermore, we show that $l_{1,\infty}$ is able to find jointly sparse solutions (i.e. parameters matrices with few non-zero rows).

6.2 A projected gradient method for $l_{1,\infty}$ regularization

In this section we start by describing a convex constrained optimization formulation of $l_{1,\infty}$ regularization, followed by a concrete application to multitask learning. We finish by introducing the projected gradient approach that we will use to solve the problem.

6.2.1 Constrained Convex Optimization Formulation

Assume we have a dataset $\mathcal{D} = (z_1, z_2, \ldots, z_n)$ with points z belonging to some set Z, and a $d \times m$ parameter matrix $W = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ where $\mathbf{w}_k \in R^d$ for $k = \{1, \ldots, m\}$ is the k-th column of W. For example in a multitask setting $\mathbf{w}_k$ would be the parameters for the k-th task. We also have a convex loss function $L(z, W)$ that measures the loss incurred by W on a training sample z. Let us now define the $l_{1,\infty}$ norm:

$$||W||_{1,\infty} = \sum_{j=1}^{d} \max_k |W_{j,k}| \tag{6.1}$$

When used as a regularization norm $l_{1,\infty}$ induces solutions where only a few rows will contain non-zero values. In fact Tropp (2006a) showed that under certain conditions the $l_{1,\infty}$ regularization norm is a convex relaxation of a pseudo-norm that counts the number of

non-zero rows of W.

One possibility for defining the $l_{1,\infty}$ regularization problem is to set it as a soft constraint:

$$\min_{W} \sum_{i=1}^{n} \mathsf{Loss}(z_i, W) + \lambda ||W||_{1,\infty} \tag{6.2}$$

Here λ is a parameter that captures the trade-off between error and sparsity. Another natural way of formulating the regularization problem is to set it as a constrained convex optimization:

$$\min_{W} \sum_{i=1}^{n} \mathsf{Loss}(z_i, W) \text{ s.t. } ||W||_{1,\infty} \leq C \tag{6.3}$$

In this case C is a bound on $||W||_{1,\infty}$ and serves a role analogous to that of λ in the previous formulation. In this chapter we concentrate on this latter formulation.

6.2.2 An Application: Multitask Learning

To give a concrete application let us describe the multitask joint regularization setting. The goal here is to train m jointly-sparse linear classifiers, one for each task. By jointly sparse we mean that we wish only a few features to be non-zero in any of the m problems. We can formulate this problem as an $l_{1,\infty}$ regularized objective.

Following the notation from the previous section, Z is the set of tuples: $(\mathbf{x}_i, y_i, l_i)$ for $i = 1 \ldots n$ where each $\mathbf{x}_i \in R^d$ is a feature vector, $l_i \in \{1, \ldots, m\}$ is a label specifying to which of the m tasks the example corresponds to, and $y_i \in \{+1, -1\}$ is the label for the example. Equivalently, using the notation from chapter 5 we could assume a collection of datasets $\mathcal{D} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k\}$ where $\mathcal{T}_k = \{(\mathbf{x}_1^k, y_1^k), (\mathbf{x}_2^k, y_2^k), \ldots, (\mathbf{x}_{n_k}^k)\}$ Assume that we wish to learn m linear classifiers of the form:

$$f_k(\mathbf{x}) = \mathbf{w}_k \cdot \mathbf{x} \tag{6.4}$$

and let $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]$ be a $d \times m$ matrix where $W_{j,k}$ corresponds to the j-th parameter of the k-th problem. So in the ideal case we would like a solution matrix W with a few non-zero rows (i.e. a few active features). In order to assess the classification

performance multiple classification losses could be used, in this chapter we used the hinge loss:

$$\mathsf{Loss}_{\mathrm{H}}(z, W) = \max(0, 1 - f_k(\mathbf{x})y) \tag{6.5}$$

In this formulation, the hinge loss encourages correct classification of the points and the $l_{1,\infty}$ norm is similar to l_1 regularization, but it encourages joint sparsity across the different tasks.

6.2.3 A Projected Gradient Method

Our algorithm for optimizing equation (6.3) is based on the projected subgradient method for minimizing a convex function $\mathrm{F}(W)$ subject to convex constraints of the form $W \in \Omega$, where Ω is a convex set (Bertsekas, 1999). In our case $\mathrm{F}(W)$ is some convex loss function, W is a parameter matrix and Ω is the set of all matrices with $||W||_{1,\infty} \leq C$.

A projected subgradient algorithm works by generating a sequence of solutions W^t via $W^{t+1} = \mathrm{P}_\Omega(W^t - \eta\nabla^t)$. Here ∇^t is a subgradient of F at W^t and $\mathrm{P}_\Omega(W)$ is is the Euclidean projection of W onto Ω, given by:

$$\min_{W' \in \Omega} ||W' - W||^2 = \min_{W' \in \Omega} \sum_{j,k} \left(W'_{j,k} - W_{j,k}\right)^2 \tag{6.6}$$

Finally, η is the learning rate that controls the amount by which the solution changes at each iteration.

Standard results in optimization literature (Bertsekas, 1999) show that when $\eta = \frac{\eta_0}{\sqrt{t}}$ and $\mathrm{F}(W)$ is a convex Lipschitz function the projected algorithm will converge to an ϵ-accurate solution in $O(1/\epsilon^2)$ iterations.

For the hinge loss case described in the previous section, computing the subgradient of the objective of equation (6.3) is straightforward. The subgradient for the parameters of each task can be computed independently of the other tasks, and for the k-th task it is given by summing examples of the task whose margin is less than one:

$$\nabla_k^t = \sum_{i \,:\, l_i = k,\ f_k(\mathbf{x}_i) y_i < 1} y_i \mathbf{x}_i \tag{6.7}$$

In the next section we show how to compute the projection onto the $l_{1,\infty}$ ball efficiently.

6.3 Efficient Projection onto the $l_{1,\infty}$ Ball

We start this section by using the Lagrangian of the projection to characterize the optimal solution. This will allow us to map the projection to a simpler problem for which we can develop an efficient algorithm, that we present in the second part of the section.

6.3.1 Characterization of the solution

We now describe the projection of a matrix A to the $l_{1,\infty}$ ball. For now, we assume that all entries in A are non-negative, later we will show that this assumption imposes no limitation. The projection can be formulated as finding a matrix B that solves the following convex optimization problem:

$$\mathbf{P_{1,\infty}}: \quad \min_{B,\boldsymbol{\mu}} \quad \frac{1}{2}\sum_{i,j}(B_{i,j}-A_{i,j})^2 \tag{6.8}$$

$$\text{s.t.} \quad \forall i,j \;\; B_{i,j} \leq \mu_i \tag{6.9}$$

$$\sum_i \mu_i = C \tag{6.10}$$

$$\forall i,j \;\; B_{i,j} \geq 0 \tag{6.11}$$

$$\forall i \;\; \mu_i \geq 0 \tag{6.12}$$

In the above problem, the objective (6.8) corresponds to the Euclidean distance between A and B, whereas the constraints specify that B is in the boundary of the $l_{1,\infty}$ ball of radius C. To do so, there are variables $\boldsymbol{\mu}$ that stand for the the maximum coefficients of B for each row i, as imposed by constraints (6.9), and that sum to the radius of the ball, as imposed by constraint (6.10). Constraints (6.11) and (6.12) stand for non-negativity of the new coefficients and maximum values.

We now present the Lagrangian of problem $\mathrm{P}_{1,\infty}$ and three lemmas that will be used to

derive an efficient algorithm. The Lagrangian is:

$$\begin{aligned}\mathcal{L}(B, \boldsymbol{\mu}, \boldsymbol{\alpha}, \theta, \boldsymbol{\beta}, \boldsymbol{\gamma}) &= \frac{1}{2}\sum_{i,j}(B_{i,j} - A_{i,j})^2 \\ &+ \sum_{i,j}\alpha_{i,j}(B_{i,j} - \mu_i) + \theta\Big(\sum_i \mu_i - C\Big) \\ &- \sum_{i,j}\beta_{i,j}B_{i,j} - \sum_i \gamma_i\mu_i\end{aligned}$$

Lemma 1 *At the optimal solution of* $\mathrm{P}_{1,\infty}$ *there exists a constant* $\theta \geq 0$ *such that for every* i*: either (a)* $\mu_i > 0$ *and* $\sum_j(A_{i,j} - B_{i,j}) = \theta$*; or (b)* $\mu_i = 0$ *and* $\sum_j A_{i,j} \leq \theta$*.*

Proof: Differentiating $\mathcal{L}$ with respect to $B_{i,j}$ gives the optimality condition $\frac{\partial \mathcal{L}}{\partial B_{i,j}} = B_{i,j} - A_{i,j} + \alpha_{i,j} - \beta_{i,j} = 0$. Differentiating $\mathcal{L}$ with respect to μ_i gives the optimality condition $\frac{\partial \mathcal{L}}{\partial \mu_i} = \theta - \sum_j \alpha_{i,j} - \gamma_i = 0$.

We now assume $\mu_i > 0$ to prove (a). The complementary slackness conditions imply that whenever $\mu_i > 0$ then $\gamma_i = 0$ and therefore $\theta = \sum_j \alpha_{i,j}$. If we assume that $B_{i,j} > 0$, by complementary slackness then $\beta_{i,j} = 0$, and therefore $\alpha_{i,j} = A_{i,j} - B_{i,j}$. If $B_{i,j} = 0$ then $B_{i,j} - \mu_i \neq 0$ and so $\alpha_{i,j} = 0$ due to complementary slackness; we then observe that $A_{i,j} = -\beta_{i,j}$, but since $A_{i,j} \geq 0$ and $\beta_{i,j} \geq 0$ it must be that $A_{i,j} = 0$; so we can express also $\alpha_{i,j} = A_{i,j} - B_{i,j}$. Thus, $\theta = \sum_j(A_{i,j} - B_{i,j})$ which proves (a).

When $\mu_i = 0$, $B_{i,j} = 0$ because of (6.9) and (6.11). Then, using the optimality conditions $\frac{\partial \mathcal{L}}{\partial B_{i,j}}$ we get that $\alpha_{i,j} = A_{i,j} + \beta_{i,j}$. Plugging this into $\frac{\partial \mathcal{L}}{\partial \mu_i}$ we get $\theta = \sum_j(A_{i,j} + \beta_{i,j}) + \gamma_i$. By definition $\beta_{i,j} \geq 0$ and $\gamma_i \geq 0$, which proves (b). $\square$

Lemma 1 means that when projecting A, for every row whose sum is greater than θ, the sum of the new values in the row will be reduced by a constant θ. The rows whose sum is less than θ will become zero.

The next lemma reveals how to obtain the coefficients of B given the optimal maximums $\boldsymbol{\mu}$.

Lemma 2 *Let* $\boldsymbol{\mu}$ *be the optimal maximums of problem* $\mathrm{P}_{1,\infty}$*. The optimal matrix* B *of* $\mathrm{P}_{1,\infty}$

satisfies that:

$$A_{i,j} \geq \mu_i \implies B_{i,j} = \mu_i \tag{6.13}$$

$$A_{i,j} \leq \mu_i \implies B_{i,j} = A_{i,j} \tag{6.14}$$

$$\mu_i = 0 \implies B_{i,j} = 0 \tag{6.15}$$

Proof: If $\mu_i = 0$ the lemma follows directly from (6.9) and (6.11). The rest of the proof assumes that $\mu_i > 0$.

To prove (6.13), assume that $A_{i,j} \geq \mu_i$ but $B_{i,j} \neq \mu_i$. We consider two cases. When $B_{i,j} > 0$, by (6.9), if $B_{i,j} \neq \mu_i$ then $B_{i,j} < \mu_i$, which means $\alpha_{i,j} = 0$ due to complementary slackness. This together with $\beta_{i,j} = 0$ imply that $B_{i,j} = A_{i,j}$, and therefore $A_{i,j} < \mu_i$, which contradicts the assumption. When $B_{i,j} = 0$ then $\alpha_{i,j} = 0$ and $A_{i,j} = 0$ (see proof of Lemma 1), which contradicts the assumption.

To prove (6.14), assume that $A_{i,j} \leq \mu_i$ but $B_{i,j} \neq A_{i,j}$. We again consider two cases. If $B_{i,j} > 0$, $\beta_{i,j} = 0$; given that $B_{i,j} \neq A_{i,j}$, then $\alpha_{i,j} > 0$, and so $B_{i,j} = \mu_i$ due to complementary slackness. But since $\alpha_{i,j} > 0$, $A_{i,j} > B_{i,j} = \mu_i$, which contradicts the assumption. If $B_{i,j} = 0$ then $A_{i,j} = 0$ (see proof of Lemma 1), which contradicts the assumption $\square$

With these results, the problem of projecting into the $l_{1,\infty}$ ball can be reduced to the following problem, which finds the optimal maximums $\boldsymbol{\mu}$:

$$\mathbf{M}_{\mathbf{1},\infty}: \quad \text{find} \quad \boldsymbol{\mu}\ ,\ \theta \tag{6.16}$$

$$\text{s.t.} \quad \sum_i \mu_i = C \tag{6.17}$$

$$\sum_{j:A_{i,j} \geq \mu_i} (A_{i,j} - \mu_i) = \theta\ ,\ \forall i \text{ s.t. } \mu_i > 0 \tag{6.18}$$

$$\sum_j A_{i,j} \leq \theta\ ,\ \forall i \ \text{ s.t. } \ \mu_i = 0 \tag{6.19}$$

$$\forall i \ \ \mu_i \geq 0 \ \ ; \ \ \theta \geq 0 \tag{6.20}$$

With $\boldsymbol{\mu}$ we can create a matrix B using Lemma 2. Intuitively, the new formulation reduces finding the projection to the $l_{1,\infty}$ ball to finding a new vector of maximum absolute values that will be used to truncate the original matrix. The constraints express that the cumulative mass removed from a row is kept constant across all rows, except for those

rows whose coefficients become zero. A final lemma establishes that there is a unique solution to $\mathrm{M}_{1,\infty}$. Therefore, the original projection $\mathrm{P}_{1,\infty}$ reduces to finding the solution of $\mathrm{M}_{1,\infty}$.

Lemma 3 *For a matrix A and a constant $C < ||A||_{1,\infty}$, there is a unique solution $\boldsymbol{\mu}^*, \theta^*$ to the problem $\mathrm{M}_{1,\infty}$.*

Proof: For any $\theta \geq 0$ there is a unique $\boldsymbol{\mu}$ that satisfies (6.18), (6.19) and (6.20). To see this, consider $\theta \geq \sum_j A_{i,j}$. In this case we must have $\mu_i = 0$, by equation (6.19). For $\theta < \sum_j A_{i,j}$ we have $\mu_i = \mathrm{f}_i(\theta)$ where f_i is the inverse of the function

$$\mathrm{g}_i(\mu) = \sum_{j:A_{i,j} \geq \mu} (A_{i,j} - \mu)$$

$\mathrm{g}_i(\mu)$ is a strictly decreasing function in the interval $[0, \max_j A_{i,j}]$ with $\mathrm{g}_i(0) = \sum_j A_{i,j}$ and $\mathrm{g}_i(\max_j A_{i,j}) = 0$. Therefore it is clear that $\mathrm{f}_i(\theta)$ is also well defined on the interval $[0, \sum_j A_{i,j}]$.

Next, define

$$\mathrm{N}(\theta) = \sum_i \mathrm{h}_i(\theta)$$

where $\mathrm{h}_i(\theta) = 0$ if $\theta > \sum_j A_{i,j}$ and $\mathrm{h}_i(\theta) = \mathrm{f}_i(\theta)$ otherwise. $\mathrm{N}(\theta)$ is strictly increasing in the interval $[0, \max_i \sum_j A_{i,j}]$. Hence there is a unique θ^* that satisfies $\mathrm{N}(\theta^*) = C$; and there is a unique $\boldsymbol{\mu}^*$ such that $\mu_i^* = \mathrm{h}_i(\theta^*)$ for each i. $\square$

So far we have assumed that the input matrix A is non-negative. For the general case, it is easy to prove that the optimal projection never changes the sign of a coefficient (Duchi et al., 2008). Thus, given the coefficient matrix W used by our learning algorithm, we can run the $l_{1,\infty}$ projection on A, where A is a matrix made of the absolute values of W, and then recover the original signs after truncating each coefficient.

6.3.2 An efficient projection algorithm

In this section we describe an efficient algorithm to solve problem $\mathrm{M}_{1,\infty}$. Given a $d \times m$ matrix A and a ball of radius C, the goal is to find a constant θ and a vector $\boldsymbol{\mu}$ of maximums for the new projected matrix, such that $C = \sum_i \mu_i$.

As we have shown in the proof of Lemma 3, $\boldsymbol{\mu}$ and θ can be recovered using functions $\mathrm{N}(\theta)$ and $\mathrm{h}_i(\theta)$. Each function $\mathrm{h}_i(\theta)$ is piecewise linear with $m + 1$ intervals. Furthermore

$\mathrm{N}(\theta)$, the sum of functions $\mathrm{h}_i(\theta)$, is also piecewise linear with $dm + 1$ intervals. Section 6.3.3 describes the intervals and slopes of h_i.

Our algorithm builds these functions piece by piece, until it finds a constant θ that satisfies the conditions of problem $\mathrm{M}_{1,\infty}$; it then recovers $\boldsymbol{\mu}$. The cost of the algorithm is dominated by sorting and merging the x-coordinates of the h_i functions, which form the intervals of N. Therefore the complexity is $O(dm \log dm)$ time and $O(dm)$ in memory, where dm is the number of parameters in A. As a final note, the algorithm only needs to consider non-zero parameters of A. Thus, in this complexity cost, dm can be interpreted as the number of non-zero parameters. This property is particulary attractive for learning methods that maintain sparse coefficients.

6.3.3 Computation of h_i

In this section we describe the intervals and slope of piecewise linear functions h_i. Let $\mathbf{s}_i$ be a vector of the coefficients of row i in A sorted in decreasing order, with an added 0 at position $m + 1$, $s_{i,1} \geq s_{i,2} \geq \ldots \geq s_{i,m} \geq s_{i,m+1} = 0$. Then, let us define points $r_{i,k} = \sum_{j:A_{i,j} \geq s_{i,k}} (A_{i,j} - s_{i,k}) = \sum_{j=1}^{k} (s_{i,j} - s_{i,k}) = \sum_{j=1}^{k-1} s_{i,j} - (k-1)s_{i,k}$, for $1 \leq k \leq m+1$. Each point $r_{i,k}$ corresponds to the reduction in magnitude for row i that is obtained if we set the new maximum to $s_{i,k}$. Clearly $\mathrm{h}_i(r_{i,k}) = s_{i,k}$. Furthermore it is easy to see that h_i is piecewise linear with intervals $[r_{i,k}, r_{i,k+1}]$ for $1 \leq k \leq m$ and slope:

$$\frac{s_{i,k+1} - s_{i,k}}{\sum_{j=1}^{k} s_{i,j} - k s_{i,k+1} - \sum_{j=1}^{k-1} s_{i,j} + (k-1)s_{i,k}} = -\frac{1}{k}$$

After point $r_{i,m+1}$ the function is constant and its value is zero. Note that this comes from equation (6.19) that establishes that $\mu_i = 0$ for $\theta > r_{i,m+1} = \sum_{j=1}^{m} A_{i,j}$.

6.4 Related Work

A number of optimization methods have been proposed for $l_{1,\infty}$ regularization. In particular, Turlach et al. (2005) developed an interior point algorithm for optimizing a twice differentiable objective regularized with an $l_{1,\infty}$ norm. One of the limitations of this ap-

proach is that it requires the exact computation of the Hessian of the objective function. This might be computationally expensive for some applications both in terms of memory and time.

Schmidt et al. (2008) propose a projected gradient method for $l_{1,\infty}$ regularized models that differs in the projection strategy. In our case, the objective corresponds directly to the loss function, and we project to the $l_{1,\infty}$ ball. In contrast, their method minimizes the following regularized objective:

$$\sum_{i=1}^{n} \mathsf{Loss}(z_i, W) + \lambda \sum_{j=1}^{d} \mu_j$$
$$\text{s.t.} \quad |W_{j,k}| \leq \mu_j$$

This formulation introduces auxiliary variables μ_j that stand for the maximum absolute value of each group (i.e., the variables $W_{j,k}$ for any k). The gradient step updates all the variables without taking into account the constraints: the main variables are updated according to the gradient of the loss function, whereas the auxiliary variables are lowered by the constant λ.

Then, a projection step makes a second update that enforces the constraints. In this case, this step can be decomposed into d independent l_∞ projections, each enforcing that the auxiliary variables μ_j are in fact the maximum absolute values of each group.

Standard results in optimization theory (see 2.4) show that the convergence of a projected gradient method is in the order of:

$$\epsilon \leq O\Big(\frac{D^2 + G^2 \log(T+1)}{2\sqrt{T}}\Big)$$

where T is the number of iterations; D is an upper bound on the maximum distance between the initial solution and the optimal solution; and G is an upper bound on the norm of the gradient of the objective at any feasible point.

In our formulation, G depends only on the gradient of the loss function. For example, for the hinge loss it can be shown that G can be bounded by the size of an l_2 ball containing all the training examples (Shalev-Shwartz et al., 2007). In the method by (Schmidt et al.,

2008), G has an additional term which corresponds to the gradient of $\lambda \sum_j \mu_j$. So the convergence of their method will have an additional dependence on $d\lambda^2$.

As we have shown in chapter 5, for the special case of a linear objective the regularization problem can be expressed as a linear program. While this is feasible for small problems it does not scale to problems with large number of variables.

Our $l_{1,\infty}$ projection algorithm is related to the l_1 projection of Duchi et al. in that theirs is a special case of our algorithm for $m = 1$. The derivation of the general case for $l_{1,\infty}$ regularization is significantly more involved as it requires reducing a set of l_∞ regularization problems tied together through a common l_1 norm to a problem that can be solved efficiently.

6.5 Synthetic Experiments

For the synthetic experiments we considered a multitask setting where we compared the $l_{1,\infty}$ projection with both independent l_2 projections for each task and independent l_1 projections. In all cases we used a projected subgradient method, thus the only difference is in the projection step. For all the experiments we used the sum of average hinge losses per task as our objective function. For these experiments as well as the experiments in the following section the learning rate was set to $\eta_0/\sqrt{t}$, where η_0 was chosen to minimize the objective on the training data (we tried values 0.1, 1, 10 and 100). All models were run for 200 iterations.

To create data for these experiments we first generated parameters $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m]$ for all tasks, where each entry was sampled from a normal distribution with 0 mean and unit variance. To generate jointly sparse vectors we randomly selected 10% of the features to be the global set of relevant features V. Then for each task we randomly selected a subset $v \subseteq V$ of relevant features for the task. The size of v was sampled uniformly at random from $\{|V|/2, \ldots, |V|\}$. All parameters outside v were zeroed.

Each of the dimensions of the training points $\mathbf{x}_i^k$ for each task was also generated from a normal distribution with 0 mean and unit variance. All vectors were then normalized to have unit norm. The corresponding labels y_i^k were set to $\mathrm{sign}(\mathbf{w}_k \cdot \mathbf{x}_k^i)$. The test data was

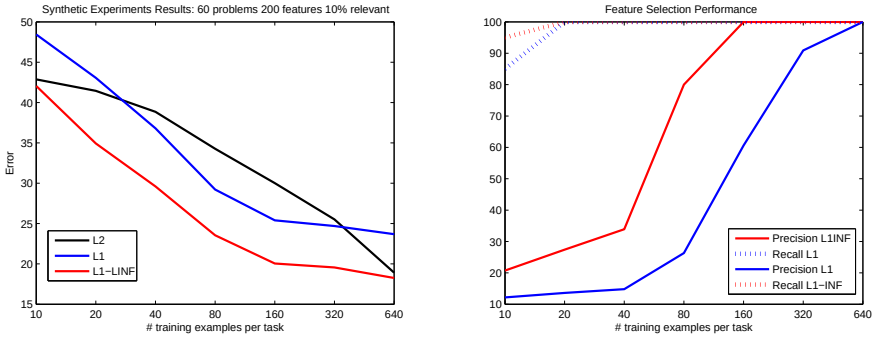

Figure 6-2: Synthetic experiments. Top: test error. Bottom: feature selection performance.

generated in the same fashion. The number of dimensions for these experiments was set to 200 and the number of problems to 60.

We evaluated three types of projections: $l_{1,\infty}$, independent l_2 and independent l_1. For each projection the ball constraints C were set to be the true norm of the corresponding parameters. That is for the $l_{1,\infty}$ norm we set $C = ||W||_{1,\infty}$. For the independent l_1 and l_2 norms we set $C_k = ||\mathbf{w}_k||_1$ and $C_k = ||\mathbf{w}_k||_2$, resp., where C_k is the regularization parameter for task k.

We trained models with different number of training examples ranging from 10 to 640 examples per task and evaluated the classification error of the resulting classifier on the test data.

Figure 6-2 shows the results of these experiments. As we would expect, given the amount of feature sharing between tasks, the $l_{1,\infty}$ projection results in better generalization than both independent l_1 and independent l_2 projections. Since we know the relevant feature set V, we can evaluate how well the l_1 and $l_{1,\infty}$ projections recovered these features. For each model we take the coefficient matrix W learned and select all the features corresponding to non-zero coefficients for at least one task. The bottom plot shows precision and recall of feature selection for each model, as we increase the number of training examples per task. As we can see both the l_1 model and the $l_{1,\infty}$ can easily recognize that a feature is in the relevant set: the recall for both models is high even with very few training examples. The main difference between the two models is in the precision at recovering relevant features: the $l_{1,\infty}$ model returns significantly sparser solutions, and thus has higher

precision.

6.6 Image Annotation Experiments

In these experiments we use our algorithm in a multitask learning application. We compare the performance of independent l_2, independent l_1, and joint $l_{1,\infty}$ regularization. In all cases we used the sum of hinge losses as our objective function. To train the l_1 regularized models we used the projected method of Duchi et al. (2008) (which is a special case of our projection when $m = 1$). To train the l_2 regularized models we used the standard SVM-Light software [2].

For these section we used the dataset Reuters.V2 described in Section 4.4. Images on the Reuters website have associated captions. We selected the 40 most frequent content words as our target prediction tasks (a content word is defined as not being in a list of *stop* words). That is, each task involved the binary prediction of whether a content word was an appropriate annotation for an image. Examples of words include: awards, president, actress, actor, match, team, people.

We partitioned the data into three sets: 10,382 images for training, 5,000 images as validation data, and 5,000 images for testing. For each of the 40 most frequent content words we created multiple training sets of different sizes, $n = \{40, 80, 160, 320, 640\}$: each training set contained n positive examples and $2n$ negative examples. All examples for each task were randomly sampled from the pool of supervised training data.

For all experiments we used as an image representation the vocabulary tree respresentation (Nister and Stewenius, 2006), with 11,000 features in our case. As a preprocessing step we performed SVD to obtain a new basis of the image space where features are uncorrelated. In preliminary experiments we observed that for both the l_1 and $l_{1,\infty}$ models this transformation eased the task of finding sparse solutions.

[2] http://svmlight.joachims.org/

6.6.1 Evaluation and Significance Testing

[3] To compare the performance of different classifiers we use the AUC criterion, which is commonly used in evaluation on retrieval tasks. For a single task, assuming a labeled test set (x_i, y_i) for $i = 1 \ldots n$, the AUC measure for a function f can be expressed (e.g., see (Agarwal et al., 2005)) as

$$\frac{1}{n^+} \sum_{i:y_i=+1} \sum_{j:y_j=-1} \frac{I[f(x_i) > f(x_j)]}{n^-} \tag{6.21}$$

where n^+ is the number of positive examples in the test set, n^- is the number of negative examples, and $I[\pi]$ is the indicator function which is 1 if π is true, 0 otherwise. The AUC measure can be interpreted (Agarwal et al., 2005) as an estimate of the following expectation, which is the probability that the function f correctly ranks a randomly drawn positive example over a randomly drawn negative item:

$$AUC(f) = \mathbf{E}_{X^+ \sim \mathcal{D}^+, X^- \sim D^-} \left[I[f(X^+) > f(X^-)] \right]$$

Here $\mathcal{D}^+$ is the distribution over positively labeled examples, and $\mathcal{D}^-$ is the distribution over negative examples.

This interpretation allows to develop a simple significance test, based on the sign test, to compare the performance of two classifiers f and g (more specifically, to develop a significance test for the hypothesis that $AUC(f) > AUC(g)$). Assuming that $n^+ < n^-$ (which is the case in all of our experiments), and given a test set, we create pairs of examples (x_i^+, x_i^-) for $i = 1 \ldots n^+$. Here each x_i^+ is a positive test example, and x_i^- is an arbitrary negative test example; each positive and negative example is a distinct item from the test set. Given the n^+ pairs, we can calculate the following counts:

$$\begin{aligned} s^+ &= \sum_i I[(f(x_i^+) > f(x_i^-)) \wedge (g(x_i^+) < g(x_i^-))] \\ s^- &= \sum_i I[(f(x_i^+) < f(x_i^-)) \wedge (g(x_i^+) > g(x_i^-))] \end{aligned}$$

These counts are used to calculate significance under the sign test.

[3] We would like to thank Shivani Agarwal for useful discussions on this subject.

# samples	l_2 p-value	l_1 p-value
4800	0.0042	0.00001
9600	0.0002	0.009
19200	0.00001	0.00001
38400	0.35	0.36
63408	0.001	0.46

Table 6.1: Significance tests for the Image Annotation task, comparing $l_{1,\infty}$ with l_2 and l_1.

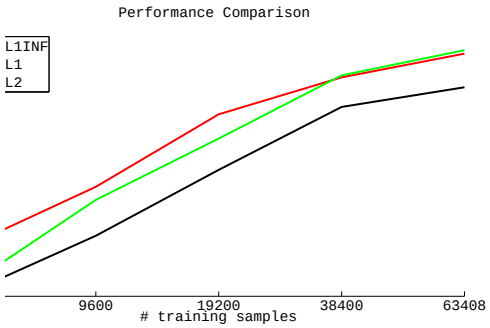

Figure 6-3: Model Comparison in the Image Annotation task: $l_{1,\infty}$ (red), l_1 (green), l_2 (black).

In our experiments, the set-up is slightly more complicated, in that we have a multitask setting where we are simultaneously measuring the performance on several tasks rather than a single task. The test set in our case consists of examples (x_i, y_i, l_i) for $i = 1 \ldots n$ where l_i specifies the task for the i'th example. We replace the AUC measure in Eq. 6.21 with the following measure:

$$\frac{1}{n^+} \sum_l \sum_{i:l_i=l,y_i=+1} \sum_{j:l_i=l,y_j=-1} \frac{I[f_l(x_i) > f_l(x_j)]}{n_l^-}$$

where n^+ is the total number of positive examples in the test set, n_l^- is the number of negative examples for the l'th task, and f_l is the classifier for the l'th task. It is straightforward to develop a variant of the sign test for this setting; for brevity the details are omitted.

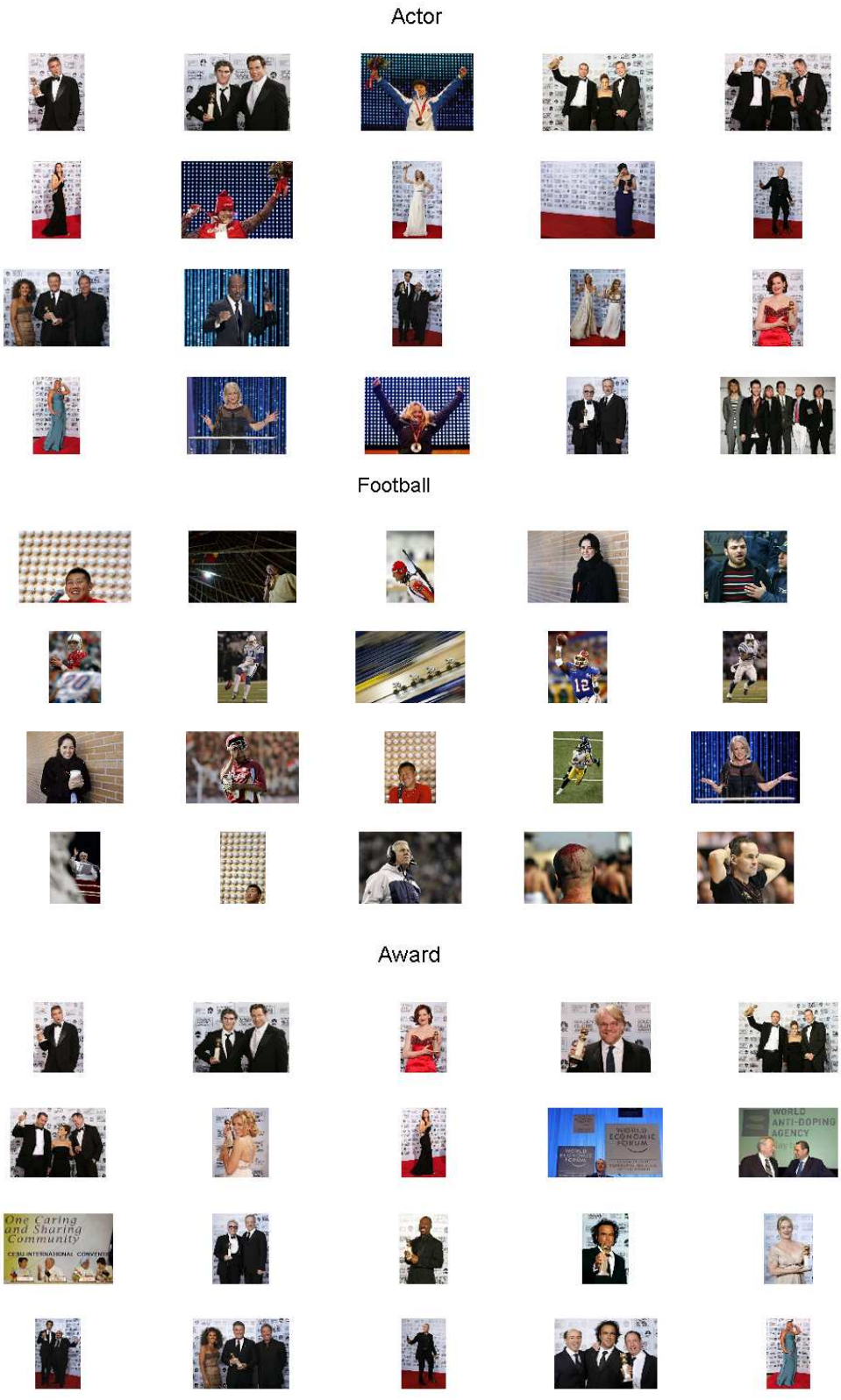

Figure 6-4: $l_{1,\infty}$ model top ranked images for words: actor,football and award

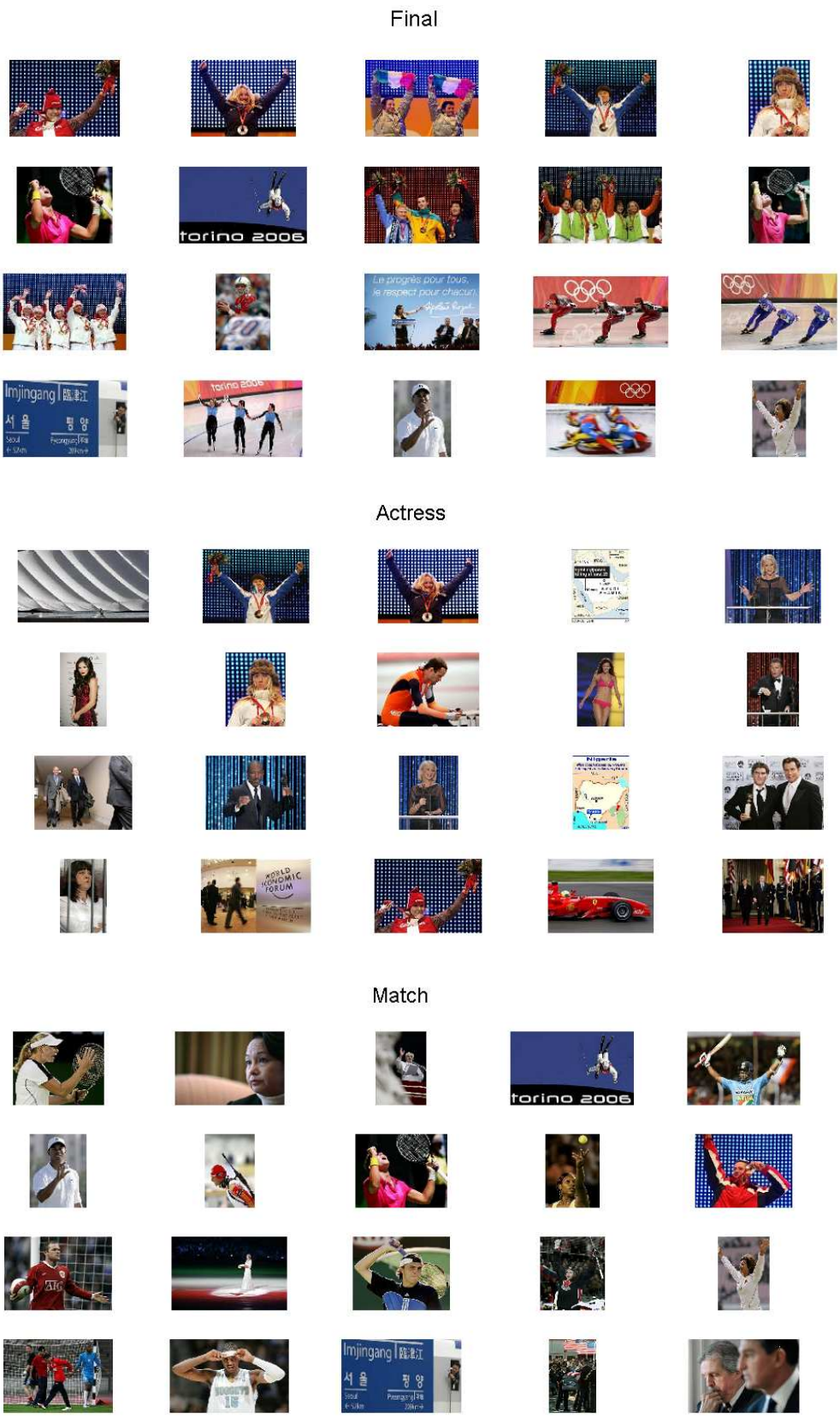

Figure 6-5: $l_{1,\infty}$ model top ranked images for words: final,actress and match

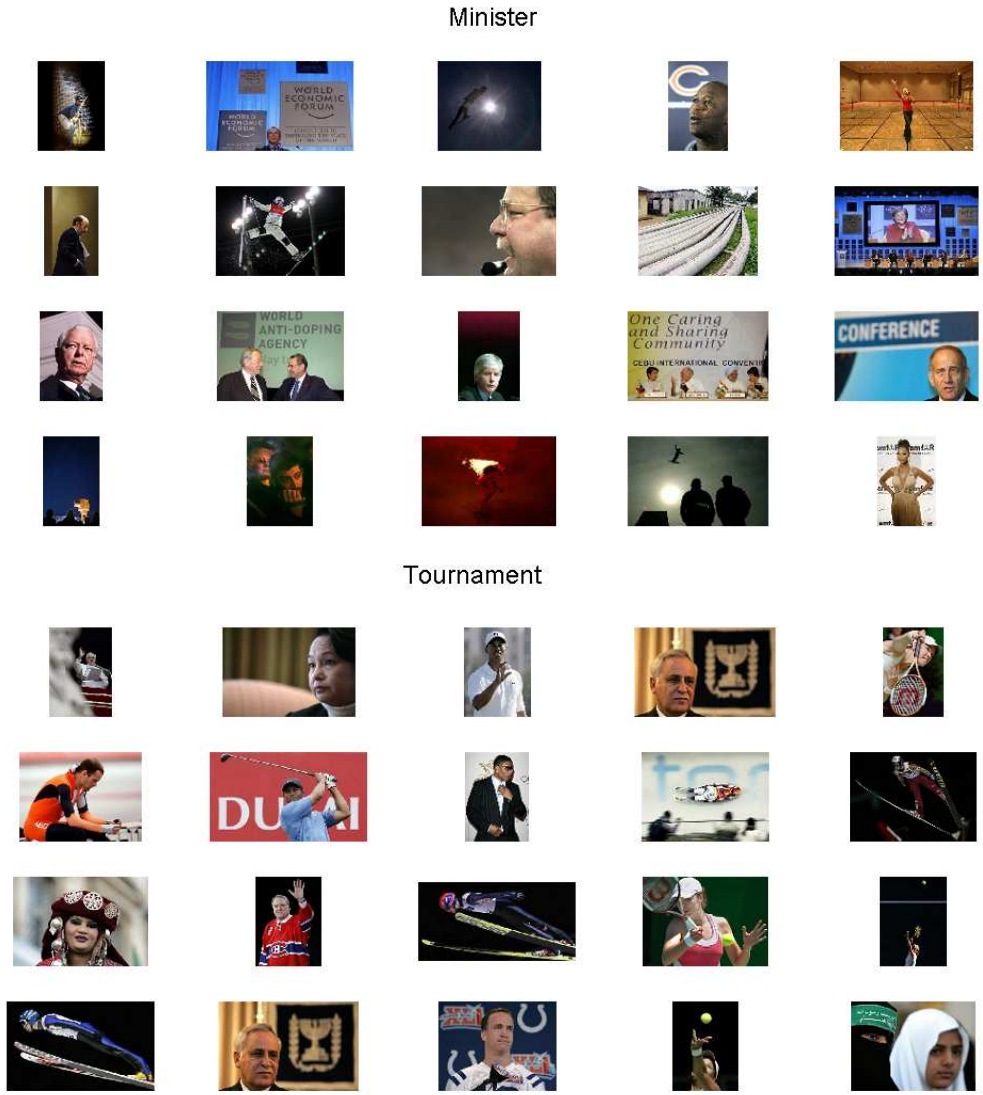

Figure 6-6: $l_{1,\infty}$ model top ranked images for words: minister and tournament

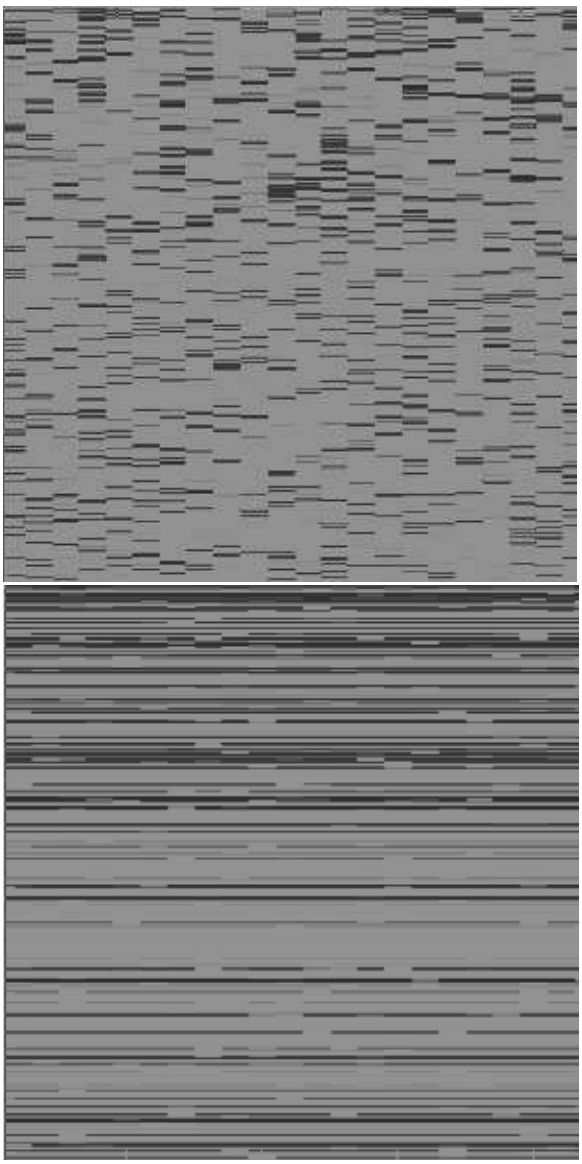

Figure 6-7: Parameter matrices of the image annotation task trained with 4,800 training samples for l_1 (left) and $l_{1,\infty}$ (right). Each row corresponds to the parameters of a feature across tasks. Gray corresponds to 0 (inactive parameter), whereas black levels indicate the absolute value of the parameters.

6.6.2 Results

To validate the constant C for each model we assume that we have 10 words for which we have validation data. We chose the C that maximized the AUC (we tried values of

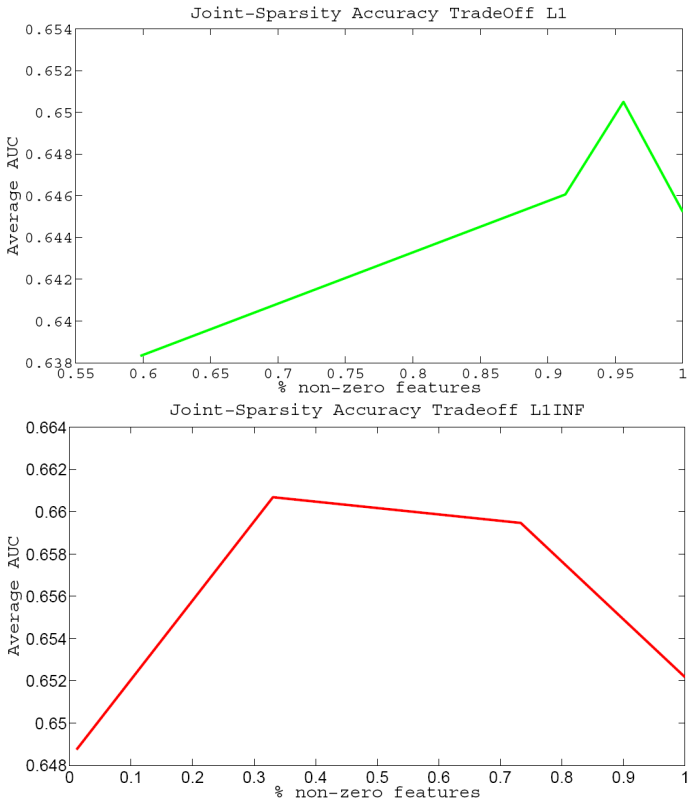

Figure 6-8: AUC measure with respect joint sparsity in the Image Annotation task, for l_1 (top) and $l_{1,\infty}$ (bottom).

$C = \{0.1, 1, 10, 100\}$).

Figure 6-3 shows results on the Reuters dataset for l_2, l_1 and $l_{1,\infty}$ regularization as a function of the total number training samples. Table 6.1 shows the corresponding significance tests for the difference between $l_{1,\infty}$ and the other two models. As we can see the $l_{1,\infty}$ regularization performs significantly better than both $l1$ and $l2$ for training sizes of less than $20,000$ samples, for larger training sizes all models seem to perform similarly. Figures 6-4, 6-5 and 6-6 show some retrieval results for the $l_{1,\infty}$ model.

Figure 6-7 shows the absolute values of the parameters matrices for each model trained with 4,800 examples, where each column corresponds to the parameters for one task and each row to a feature. As we can see $l_{1,\infty}$ regularization produces a joint sparsity pattern, where many tasks use the same features (i.e. rows).

Figure 6-8 shows the accuracy of the learned model as a function of the total number

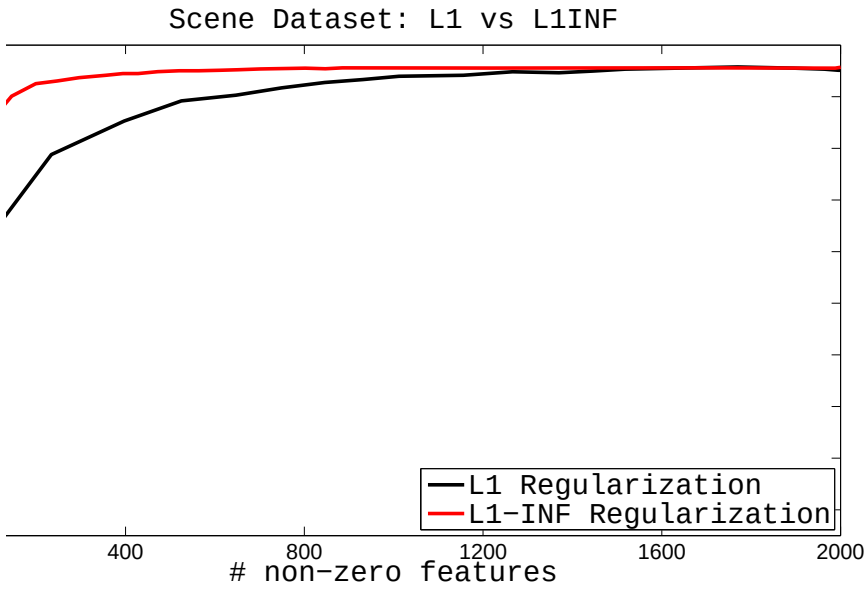

Figure 6-9: Average AUC as a function of the total number of features used by all models. To create this figure we ran both the l_1 and $l_{1,\infty}$ models with different values of C. This resulted in models with different numbers of non-zero features.

of non-zero features (i.e. the number of features that were used by any of the tasks), where we obtained solutions of different sparsity by controlling the C parameter. Notice that the $l_{1,\infty}$ model is able to find a solution of 66% average AUC using only 30% of the features while the l_1 model needs to use 95% of the features to achieve a performance of 65%.

6.7 Scene Recognition Experiments

We also conducted some experiments on an indoor Scene recognition dataset containing 67 indoor scene categories. We used 50 images per class for training, 30 images as a source of unlabeled data to build a representation and the remaining 20 images for testing.

We used the unlabeled data: $\mathcal{U} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\}$ to create a very simple representation based on similarities to the unlabeled images. As a basic image representation $g(\mathbf{x})$ we choose the Gist descriptor (Oliva and Torralba, 2001). Using the unlabeled images we

compute the following representation:

$$v(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), k(\mathbf{x}, \mathbf{x}_2), \ldots, k(\mathbf{x}, \mathbf{x}_q)] \tag{6.22}$$

where $k(\mathbf{x}_i, \mathbf{x}_j) = \exp \frac{-||g(\mathbf{x}_i) - g(\mathbf{x}_j)||_2^2}{2\sigma^2}$. We will refer to the unlabeled images as templates.

In the experiments we trained one-vs-all classifiers for each category using the hinge loss. We compare l_1 regularization with $l_{1,\infty}$ joint regularization. As a performance metric we report the average AUC computed over all classes.

Figure 6-9 shows average performance for l_1 and $l_{1,\infty}$ regularized models as a function of the total number of non-zero features used by any of the 67 classifiers. To create this figure we ran both the l_1 and $l_{1,\infty}$ models with different values of C. As we would expect when no sparsity restriction is imposed both models perform equivalently. However, for sparse models the $l_{1,\infty}$ regularization results in significantly better performance. In particular, the $l_{1,\infty}$ regularized model obtains $\approx 80\%$ AUC using $\approx 10\%$ of the features while the l_1 models needs $\approx 30\%$ of the features to achieve similar performance.

Figures 6-10 and 6-11 show some examples of features (i.e. templates) chosen by a joint model that has a total of ≈ 100 non-zero parameters. We observe that the features chosen by the jointly sparse model seem to have a generic nature. For example, the first and third features in Figure 6-10 are filter type features that seem to discriminate between vertical and horizontal structures.

The feature before the last one in Figure 6-11 appears to recognize indoor scenes that have shelves. For this feature, Figure 6-12 shows the parameter values across the 67 categories. Most of the features chosen by the $l_{1,\infty}$ model seem to have a similar pattern in the sense that they partition evenly the space of scenes.

6.8 Chapter Summary

In recent years the $l_{1,\infty}$ norm has been proposed for joint regularization. In essence, this type of regularization aims at extending the l_1 framework for learning sparse models to a setting where the goal is to learn a set of jointly sparse models.

In this chapter we have presented a simple and effective projected gradient method

Figure 6-10: This figure shows some examples of features (i.e. templates) chosen by a joint model with a total of 100 nonzero parameters. Each row corresponds to one feature. The first column shows the template (i.e. unlabeled image). The following six columns show the images in the training set most similar to the template. The last three columns show the images less similar to the template.

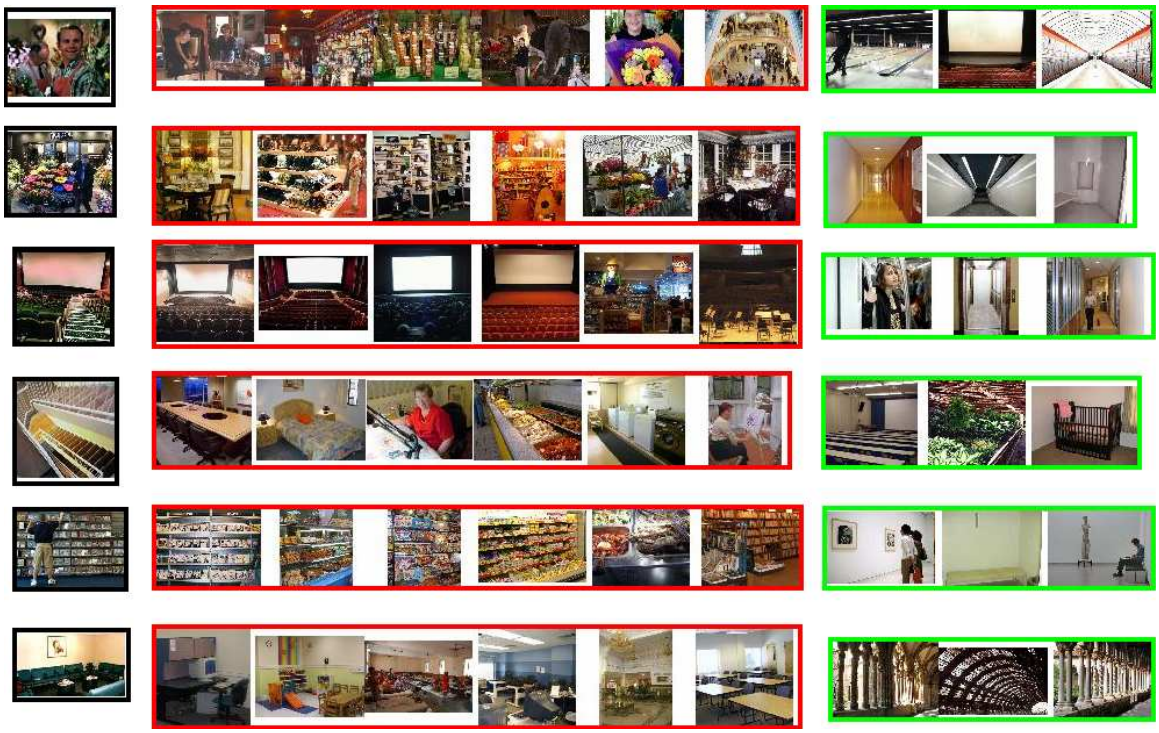

Figure 6-11: This figure shows some examples of features (i.e. templates) chosen by a joint model with a total of 100 nonzero parameters. Each row corresponds to one feature. The first column shows the template (i.e. unlabeled image). The following six columns show the images in the training set most similar to the template. The last three columns show the images less similar to the template.

Figure 6-12: The figure on the top left shows the parameter weights across the 67 categories for the feature before the last one in figure 6-11. The figure on the top right shows the categories with positive parameter weights (red) and negative parameter weights (green). The figure in the bottom shows example images of positive and negative categories.

for training joint models with $l_{1,\infty}$ constraints. The main challenge in developing such a method resides on being able to compute efficient projections to the $l_{1,\infty}$ ball. We present an algorithm that works in $O(n \log n)$ time and $O(n)$ memory where n is the number of parameters. This matches the computational cost of the most efficient algorithm (Duchi et al., 2008) for computing l_1 projections.

We have applied our algorithm to a multitask image annotation problem. Our results show that $l_{1,\infty}$ leads to better performance than both l_2 and l_1 regularization. Furthermore $l_{1,\infty}$ is effective in discovering jointly sparse solutions.

One advantage of our approach is that it can be easily extended to work on an online convex optimization setting (Zinkevich, 2003). In this setting the gradients would be computed with respect to one or few examples. Future work should explore this possibility.

Chapter 7

Conclusion

7.1 Thesis Contributions

Ideally, we would like to build image classifiers that can exploit a large number of complex features. Working with such rich representations comes at an added cost: Training might require large amounts of supervised data but only a few labeled examples might be available for a given task. To address this problem we proposed transfer algorithms that: 1) Leverage unlabeled examples annotated with meta-data and 2) Leverage supervised training data from related tasks. In particular, this thesis makes the following contributions:

- **Learning Image Representations using Unlabeled Data annotated with Meta-Data:**

  We presented a method based on *structure learning* (Ando and Zhang, 2005) that leverages unlabeled data annotated with meta-data. Our method is able to learn an image representation that is low-dimensional, but nevertheless captures the information required to discriminate between image categories.

  Our results show that using the meta-data is essential to derive efficient image representations, i.e. a model that ignores the meta-data and learns a visual representation by performing PCA on the unlabeled images alone did not improve performance.

- **Jointly Regularized Model:**

  We developed a transfer algorithm that can exploit supervised training data from re-

lated tasks. Our algorithm is based on the observation that to train a classifier for a given image classification task we might only need to consider a small subset of relevant features and that related tasks might share a significant number of such features. We can find the optimal subset of shared features by performing a joint loss minimization over the training sets of related tasks with a shared regularization penalty that minimizes the total number of features involved in the approximation.

In our first set of experiments we used the joint model in an *asymmetric transfer* setting. Our results demonstrate that when only few examples are available for training a *target* task, leveraging knowledge learnt from other tasks using our joint learning framework can significantly improve performance.

- **An Efficient Algorithm for Joint Regularization:**

  While the LP formulation is feasible for small data-sets, it becomes untractable for optimizing problems involving thousands of dimensions and examples. To address this limitation we presented a general, simple and effective projected gradient method for training $l_{1,\infty}$ regularized models with guaranteed convergence rates of $O(\frac{1}{\epsilon^2})$.

  The proposed algorithm has $O(n \log n)$ time and memory complexity, with n being the number of parameters of the joint model. This cost is comparable to the cost of training m independent sparse classifiers. Therefore, our work provides a tool that makes implementing a joint sparsity regularization penalty as easy and almost as efficient as implementing the standard l_1 and l_2 penalties.

  We tested our algorithm in a *symmetric transfer* image annotation problem. Our results show that when training data per task is scarce $l_{1,\infty}$ leads to better performance than both l_2 and l_1 regularization.

  Furthermore, $l_{1,\infty}$ is effective in discovering jointly sparse solutions. This is important for settings where: 1) Computing features is expensive and 2) We need to evaluate a large number of classifiers. In such cases finding jointly sparse solutions is essential to control the run-time complexity of evaluating multiple classifiers.

7.2 Discussion

7.2.1 Representations and Sparsity

For the experiments on the Reuters dataset presented in chapter 6 we performed SVD on the raw feature representation. Performing SVD results in a new feature representation with the following properties: 1) Features are uncorrelated and 2) Every feature is now a linear combination of the raw features. We have observed that for some feature representations performing SVD to obtain a new representation with properties 1 and 2 eases the task of finding sparse solutions. Whether whitening the data or not is necessary depends greatly on the feature representation itself. Ideally, the raw feature representation would be rich and complex enough so as to make the SVD step unnecessary. For example no such step was needed for the scene recognition experiments presented in 6.7.

Note however that performing SVD has consequences on the computational cost of the method at test time. For a given test point all the raw features will need to be computed to project the point to the sparse SVD representation. Finding a sparse solution will lead to computational savings in the SVD representation both in terms of memory and time since less eigenvectors need to be stored an less inner products need to be computed.

7.2.2 Differences between feature sharing and learning hidden representations approach

The *learning hidden representations* approach of Ando and Zhang (2005) is appealing because it can exploit latent structures in the data. However, such a power comes at a relatively high computational cost: when used for *symmetric transfer* the corresponding joint training algorithm must perform an alternating optimization where each iteration involves training classifiers for each task.

The *feature sharing* approach might not be able to uncover latent structures but as we have shown in chapter 6 it can be implemented very efficiently, i.e. we can derive joint learning algorithms whose computational cost is in essence the same as independent training.

This approach is appealing for applications were we can easily generate a large number of candidate features to choose from. This is the case in many image classification problems, for example in chapter 5 we generate a large set of candidate features using a kernel function and unlabeled examples.

Furthermore, finding jointly sparse models is useful when obtaining a subset of relevant features is in itself a goal. For example, consider a representation based on distances to unlabeled images and an active learning setting where we improve our representation by refining the annotation of the most relevant unlabeled images (i.e. annotating objects contained in the image).

We would like to note that as shown in chapter 4, in the case of *asymmetric transfer* a single iteration of the alternating minimization algorithm might suffice to compute a useful representation. The representation learnt with this approach is orthogonal to the representation learnt with joint regularization. Therefore, future work should explore combining both representations to further improve performance.

7.3 Future Work

Some possible avenues of future work are:

- **Task Clustering:**

  One of the limitations of our transfer algorithm is that it assumes the existence of a subset of relevant features shared by all tasks. However, in practice for a given set of problems there might be clusters of tasks that share relevant features among them. Discovering such clusters might allow us to share information more efficiently across tasks.

- **Online Optimization:**

  One advantage of the projected gradient method is that it can be easily extended to work on an online convex optimization setting Zinkevich (2003). In this setting the gradients would be computed with respect to one or few examples, which might improve the overall efficiency of the algorithm.

- **Feature Representations and Feature Sharing:**

  If there exists a subset of features that can be shared across tasks our algorithm will find it. However, if such subset does not exist we will not be able to share information across tasks. Therefore, given a vision application it would be interesting to investigate which types of feature representations are optimal to promote sharing.

- **Generalization Properties of $l_{1,\infty}$ Regularized Linear Models:**

  The generalization properties of l_1 regularized linear models have been well studied (Zhang, 2002) but little is known about the generalization properties of $l_{1,\infty}$ regularized linear models (Tropp, 2006a; Negahban and Wainwrigh, 2008)

Bibliography

S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth. Generalization bounds for the area under the roc curve. *JMLR*, 6:393–425, 2005.

Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *Proceedings of ICML*, 2007.

R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Proceedings of NIPS*, 2006.

B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

M. Balcan, A. Blum, and S. Vempala. Kernels as features: On kernels, margins, and low-dimensional mappings. In *Machine. Learning Journal, 65(1):79 94*, 2004.

S. Baluja. Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In *In Neural and Information Processing Systems (NIPS)*, 1998.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

S. Boyd and A. Mutapcic. Subgradient methods. In *Lecture Notes for EE364b, Standform University*, 2007.

L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of ICML*, 2004.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20(2):273–297, 2005.

D. Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. Technical report, Techical report, Statistics Dpt, Standford University, 2004.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of International Conference on Machine Learning*, 2008.

T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proceedings of KDD*, 2004.

L. Fei-Fei, P. Perona, and R. Fergus. One-shot learning of object categories. *Pattern Analysis and Machine Intelligence*, 28(4), 2006.

M. Fink. Object classification from a single example utilizing class relevance metrics. In *Proceedings of NIPS*, 2004.

J.H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 1998.

A. Holub, M. Welling, and P. Perona. Combining generative models and fisher kernels for object recognition. In *Proc. ICCV 2005*, 2005.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *In Advances in Neural Information Processing Systems 11*, 1998.

L. Jacob, F. Bach, and J.P. Vert. Clustered multi-task learning: A convex formulation. In *Proceedings of NIPS*, 2008.

T. Jebara. Multi-task feature and kernel selection for svms. In *Proceedings of ICML*, 2004.

S. Kakade, K. Sridharan, and A. Tewari. On the complexity of linear prediction: Risk bounds, margin bounds and regularization. In *NIPS*, 2008.

M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

S. I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of markov networks using l1-regularization. In *NIPS*, 2006.

J. Mutch and D. Lowe. Multiclass object recognition with sparse, localized features. In *Proceedings of CVPR*, 2006.

S. Negahban and M. Wainwrigh. Phase transitions for high-dimensional joint support recovery. In *Proceedings of NIPS*, 2008.

Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *ICML*, 2004.

K. Nigam, A. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2):103–134, 2000.

D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

G. Obozinski, B. Taskar, and M. Jordan. Multi-task feature selection. In *Technical Report*, 2006.

G. Obozinski, M. Wainwright, and M. Jordan. High-dimensional union support recovery in multivariate regression. In *Proceedings of NIPS*, 2008.

A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal in Computer Vision*, 42:145–175, 2001.

A. Quattoni, M. Collins, and T. Darrell. Learning visual representations using images with captions. In *Proc. CVPR 2007*, 2007.

A. Quattoni, M. Collins, and T. Darrell. Transfer learning for image classification with sparse prototype representations. In *Proc. CVPR 2008*, 2008.

A. Quattoni, X. Carreras, M. Collins, and T. Darrell. An efficient projection for l1,infinity regularization. In *Proc. ICML 2009*, 2009.

R. Raina, A. Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *Proceedings of the 23rd International Conference on Machine learning*, pages 713–720, 2006.

M. Schmidt, K. Murphy, G. Fung, and R. Rosale. Structure learning in random fields for heart motion abnormality detection. In *CVPR*, 2008.

M. Seeger. Learning with labeled and unlabeled data. Technical report, Institute for Adaptive and Neural Computation, Univ. of Edinburgh, 2001.

S. Shalev-Shwartz and N. Srebro. Svm optimization: Inverse dependence on training set size. In *Internation Conference of Machine Learning*, 2008.

S. Shalev-Shwartz, Y. Singer, and A. Y. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of ICML*, 2004.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of International Conference on Machine Learning*, 2007.

J. Sivic, B. Russell, A. Efros, Zisserman, and W. Freeman. Discovering object categories in image collections. *Proc. Int'l Conf. Computer Vision, Beijing, 2005*, 2005.

N. Srebro and T. Jaakkola. Weighted low-rank approximations. *MACHINE LEARNING INTERNATIONAL CONFERENCE*, 20:720–727, 2003.

S. Thrun. Is learning the n-th thing any easier than learning the first? In *In Advances in Neural Information Processing Systems*, 1996.

A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *Pattern Analysis and Machine Intelligence*, In press, 2006.

J. Tropp. Algorithms for simultaneous sparse approximation, part ii: convex relaxation. In *Signal Process. 86 (3) 589-602*, 2006a.

J. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 50(10):2231–2242, 2004.

J. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 52(3):1030–1051, 2006b.

B. Turlach, W. Venables, and S. Wright. Simultaneous variable selection. *Technometrics*, 47(3):349–363, 2005.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proceedings of ICCV*, 2007.

T. Zhang. Covering number bounds of certain regularized linear function classes. *The Journal of Machine Learning Research*, 2:527–550, 2002.

P. Zhao, G. Rocha., and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. *Technical Report 703, Statistics Department, UC Berkeley*, 2007.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of ICML*, 2003.