J. M. Keijser

Learning Transferable and Discriminative Features by Differentiable Sparse Coding

Master thesis

Thesis advisor:
dr. T.A.L. Van Erven

Date master exam: July 20, 2017

Mathematisch Instituut, Universiteit Leiden

Abstract

Sparse Coding is an unsupervised method for learning a basis which can be used to represent the data using only a few basis vectors. In recent years, this approach has successfully been applied to signal processing and classification. However, the performance of Sparse Coding deteriorates when tested on data that follows a distribution different from the data on which it was trained. This has led to the development of models that make Sparse Coding more robust against differences between training and test data. Distinct efforts in supervised Sparse Coding have resulted in Sparse Coding models that are optimized for prediction problems. This thesis builds on both lines of research in order to develop a Sparse Coding model that learns from different datasets and simultaneously allows direct optimization for a classification task. Robustness against dataset differences is achieved by transforming dissimilar samples to a shared representation. Direct optimization of the model is achieved by backpropagation of the classification error, which is made possible by switching from the conventional non-differentiable L_1-regularization to a differentiable alternative. Effectiveness of the proposed model is demonstrated by applying it to classification of handwritten digits from different datasets.

Contents

Chapter 1

Introduction

Sparse Coding refers to modeling data as the linear combination of a few elements from a basis that is learned from the data. In recent years, this approach has successfully been applied to signal processing and classification [62]. However, the performance of Sparse Coding models can deteriorate when the basis is used to model a dataset that follows a distribution that is different from the data on which it is learned [42]. This has motivated extensions of Sparse Coding that make the method more robust against domain differences [57], [42], [41], [3]. These efforts are part of the larger field of transfer learning in which models are developed that can learn from different but related datasets [54]. Other extensions of Sparse Coding are designed to tune the dictionary for a specific task [4], [46], [49], [71], [45]. It has been shown that this leads to significant improvements compared to the original unsupervised approach.
This thesis builds on both lines of research in order to develop a Sparse Coding model that is robust against domain differences and simultaneously can be directly optimized for a classification task. In this introductory chapter I give a brief summary of the related sparse coding methods, since detailed descriptions are given in Chapter 3. Section 1.1 introduces the unsupervised Sparse Coding model. Section 1.2 is devoted to different proposals for transfer Sparse Coding. Section 1.3 gives an overview of Sparse Coding algorithms that can be optimized for a predictive task. Finally, Section 1.4 describes the contributions of this thesis and the thesis outline.

1.1 Sparse Coding

Sparse Coding is an unsupervised method for learning a basis which can be used to represent each input sample using only a few basis vectors. Formally, the goal is finding a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_K]$ that can be used to approximate the input signals $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ using coefficient vector $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$. Here $\mathbf{b}_j, \mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{s}_i \in \mathbb{R}^K$. The number of basis vectors is allowed to exceed the dimensionality of the data ($K > d$), leading to a so-called *overcomplete* dictio-

nary. Sparse Coding shrinks and selects the coefficients to a sparse solution (i.e. one that contains only a few non-zero entries). This is done by incorporating a sparsity-inducing regularizer ϕ in the loss function:

$$(\mathbf{B}^*, \mathbf{S}^*) = \arg\min_{\mathbf{B},\mathbf{S}} \sum_i \|\mathbf{x}_i - \mathbf{B}\mathbf{s}_i\|_2^2 + \lambda \sum_i \phi(\mathbf{s}_i). \tag{1.1}$$

Here λ is a tunable parameter that trades off the sparsity of $\mathbf{S}$ with the quality of the signal reconstruction. Having determined the sparse coefficients $\mathbf{S}^* = [\mathbf{s}_1^*, \ldots, \mathbf{s}_n^*]$, they can serve as a representation of $\mathbf{X}$. In contrast to the data itself, the coefficients are sparse, a desirable property for a feature representation (see Section 2.5). In this thesis, the sparse representation will be used to classify the input data. This means that the coefficients can serve as feature vectors based on which a classifier is trained to predict the class label associated with each sample. Importantly, learning the coefficients $\mathbf{S}$ does not depend on the identities or class labels, making Sparse Coding an *unsupervised* form of learning from data.

Originally developed as a model for the early visual cortex [52], Sparse Coding has been applied to problems in computer vision such as image classification [59], [46], [33], [12], [76], [74], image denoising [21], [48], [47], music genre classification [59], audio processing [28], [77] and text analysis [4]. The succes of Sparse Coding can be attributed to its ability to learn a succinct representation of the input data. For example, Raina et. al. [59] used Sparse Coding to learn useful features for classification from related unlabeled data. Another factor contributing to its success is the availability of efficient optimization methods [2], [49], [39].

Sparse Coding is one of various methods for learning representations from data that are useful for predictive tasks. Section 2.5 gives an overview of representation learning from the perspective of Sparse Coding and transfer learning.

1.2 Transfer Sparse Coding

Sparse Coding can be used to learn a succinct representation of the data. However, when the training and test data are sampled from different distributions they will be encoded using different visual codewords from the learned basis. If this representation is used by a downstream system such as a classifier, such a difference will negatively influence performance [42]. This problem is not unique to Sparse Coding since most statistical methods assume that training and test data are sampled from the same probability distribution. Since this assumption may not hold in practice, the field of transfer learning aims to develop methods that can transfer knowledge between datasets that are different but related. Here, I will give an overview of methods for transfer learning based on Sparse Coding. Chapter 2 provides an overview of the transfer learning literature.

Specifically, I will discuss Sparse Coding methods designed for transfer learning that can be used for classification tasks. These methods aim to leverage *source data* $D_S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_s}, y_{n_s})\}$ in order to predict the class labels of the *target data* $D_T = \{(\mathbf{x}_{n_s+1}, y_{n_s+1}), \ldots, (\mathbf{x}_{n_s+n_t}, y_{n_s+n_t})\}$. Here, $\mathbf{x}_i \in \mathbb{R}^d$ denotes a sample and $y_i \in \mathbb{R}$ denotes the corresponding class label. When D_S and D_T are sampled from the same probability distribution, conventional methods can be used to learn $P(y|\mathbf{x})$ based on D_S. When D_S and D_T are sampled from two different probability distributions, transfer learning methods can be used. These methods assume that the source and target data are sampled i.i.d. from the source and target probability distribution.

Long et. al. introduce Transfer Sparse Coding (TSC) [42] which explicitly reduces the distribution divergence between the source and target data by incorporating the distance between the sample means of the sparse coefficients into the SC loss function (1.1). This distance is also used by the proposed method and is given by

$$\left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{s}_i - \frac{1}{n_t} \sum_{i=n_s+1}^{n_s+n_t} \mathbf{s}_i \right\|_2^2. \tag{1.2}$$

A more elaborate treatment of TSC can be found in Section 3.2.

Shekhar et. al. [61] first project the source and target data to a shared subspace before learning a sparse representation. This is done by jointly learning a dictionary and a projection matrix $\mathbf{P} \in \mathbb{R}^{d' \times d}$ that maps the data to a lower dimensional representation ($d' \leq d$). The dictionary is made class-specific by incorporating label information in the optimization procedure using the method of Fisher Discriminative Dictionary Learning (FDDL) [71]. The SDDL loss is defined as

$$\min_{\mathbf{B},\mathbf{P},\mathbf{S}} \quad \|\mathbf{PX} - \mathbf{BS}\|_F^2 + \lambda \|\mathbf{X} - \mathbf{P}^T\mathbf{PX}\|_F^2 + \tag{1.3}$$

$$\mu \|\mathbf{PX} - \mathbf{BS}_{in}\|_F^2 + \nu \|\mathbf{BS}_{out}\|_F^2. \tag{1.4}$$

$$\text{s.t.} \quad \mathbf{PP}^T = \mathbf{I}, \;\; \|\mathbf{s}_i\|_0 \leq t \;\; \forall i = 1, \ldots, n_s + n_t \tag{1.5}$$

Here $\|\cdot\|_F^2$ denotes the Frobenius matrix norm, i.e. $\|\mathbf{A}\|_F^2 = \sum_{i,j} A_{ij}^2$ and λ, μ, ν are tunable parameters. The first term ensures a faithful reconstruction of the projected data $\mathbf{PX}$, the second term ensures that the original data $\mathbf{X}$ can be recovered from $\mathbf{PX}$. The third and fourth term increase the class-specificity of the basis vectors, each of which is assigned to one of the classes prior to optimization. This is done by defining the matrices $\mathbf{S}_{in}$ and $\mathbf{S}_{out}$ in the following way.

$$(\mathbf{S}^{in})_{l,k} \begin{cases} S_{l,k} & \text{if } \mathbf{b}_i, \mathbf{x}_k \text{ belong to same class} \\ 0 & \text{otherwise.} \end{cases}$$

and

$$(\mathbf{S}^{out})_{l,k} \begin{cases} S_{l,k} & \text{if } \mathbf{b}_i, \mathbf{x}_k \text{ belong to different classes} \\ 0 & \text{otherwise.} \end{cases}$$

The method introduced in this thesis also combines Sparse Coding with a data transformation. Therefore, SDDL will be described extensively in Section 3.4.

In summary, several methods have been proposed to make Sparse Coding more robust against differences between training and test data. Of these methods, SDDL is the only one that is able to incorporate class labels in the optimization procedure. However, SDDL uses a heuristic approach; the basis is endowed with a predefined class-specific structure rather than directly optimized for classification. The next section discusses approaches that do allow direct optimization of the basis for predictive performance.

1.3 Discriminative Sparse Coding

In addition to designing methods for transferable feature learning, a lot of effort has been directed at learning dictionaries optimized for predictive performance. Specifically, there now exist multiple approaches for learning discriminative dictionaries, i.e. dictionaries that result in sparse coefficients useful for classifying the input data. It has been shown that dictionaries optimized for compressive sensing [19], face recognition [73] and image classification [46], [49], [4], [45] outperform unsupervised dictionary learning. This thesis concentrates on the specific problem of classification.
Mairal et. al. [45] propose a framework for so-called task driven dictionary learning. Given a convex loss function $L_s(y, \mathbf{W}, \mathbf{s}^*)$ that depends on the class label y, classifier parameters $\mathbf{W}$ and sparse coefficients $\mathbf{s}^* = \mathbf{s}^*(\mathbf{x}, \mathbf{B})$, their approach alternates between finding the sparse coefficients with respect to the basis and updating $\mathbf{B}$ and $\mathbf{W}$. The sparse coefficients optimizing (1.1) are found using Least Angle Regression (LAR, [20]), updating $\mathbf{B}$ and $\mathbf{W}$ is done by projected stochastic gradient descent. This means that instead of averaging the gradient of L_s over the full dataset, it is estimated based on a random sample $(\mathbf{x}, y)$. After taking a single step in the descent direction, the new parameter is projected onto $\mathcal{W}$ or $\mathcal{B}$, the feasible sets for $\mathbf{W}$ and $\mathbf{B}$ respectively:

$$\begin{aligned} \mathbf{W} &:= \Pi_{\mathcal{W}} \left[\mathbf{W} - \rho \nabla_{\mathbf{W}} L_s(y, \mathbf{W}, \mathbf{s}^*) \right], \\ \mathbf{B} &:= \Pi_{\mathcal{B}} \left[\mathbf{B} - \rho \nabla_{\mathbf{B}} L_s(y, \mathbf{W}, \mathbf{s}^*) \right]. \end{aligned}$$

Here $\Pi_{\mathcal{W}}$ and $\Pi_{\mathcal{B}}$ are the orthogonal projections on the convex sets $\mathcal{W}$ and $\mathcal{B}$. Mairal et. al. derive an expression for $\nabla_{\mathbf{B}} L_s(y, \mathbf{W}, \mathbf{s}^*)$ that depends on $\nabla_{\mathbf{s}_{\mathcal{A}}} L_s(y, \mathbf{W}, \mathbf{s}^*)$, the gradient of the loss with respect to the non-zero entries of $\mathbf{s}$.

Bagnell et. al. [4] also propose a method for directly optimizing the basis with respect to a supervised loss function L_s. In contrast to Mairal et. al. [45], they use the unnormalized Kullback-Leibler divergence as sparsity regularization

$$\phi(\mathbf{s}\|\mathbf{p}) = \sum_k \left(s^{(k)} \log \frac{s^{(k)}}{p^{(k)}} - s^{(k)} + p^{(k)} \right) \tag{1.6}$$

in the Sparse Coding loss (1.1). Here $\mathbf{p} = (p, \dots, p) \in \mathbb{R}^K$ is a nonnegative uniform prior vector. Compared to the L_1 regularized approach, this offers the advantage of a smooth relationship between the input data and its representation. Moreover, a differentiable loss allows optimization of the basis with respect to the supervised loss by end-to-end training of the model. This means that the entire pipeline consisting of feature extraction by Sparse Coding followed by classification is directly optimized for classification. This is different from the conventional Sparse Coding approach where unsupervised feature extraction is done independently from the classification task the features are used for. Because of its usefulness for classification problems, the proposed model is based on KL-regularized Sparse Coding. The DSC model will be discussed in Section 3.3.

In addition to the two methods for directly optimizing the Sparse Coding model for supervised learning, several (classes of) heuristic approaches for discriminative Sparse Coding have been proposed. Yang et. al. [71] designed a method for explicitly encouraging basis vectors to become class-specific. This is done by partitioning the dictionary in a set of class-specific dictionaries and penalizing the reconstruction of samples using dictionary elements that are assigned to the wrong class. The authors show that this improves the discriminative powers of the sparse coefficients. While assigning class-specific vectors seems like a natural heuristic, it is a heuristic nonetheless and therefore possibly inferior to directly optimizing the basis for the task. For example, the heuristic approach does not account for possible differences in the number of basis vectors that are needed to represent various classes.
Another heuristic approach consists of incorporating a discriminative term, i.e. one that depends on classification loss, in the Sparse Coding objective (1.1). From a practical point of view, the easiest way to do this is by learning a linear predictor $\mathbf{W}$ that minimizes $\|\mathbf{H} - \mathbf{WS}\|_F^2$ [73]. Here $\mathbf{S}$ are the sparse coefficients used as features for the predictor. The matrix $\mathbf{H} \in \mathbb{R}^{C \times n}$ is defined by $H_{ij} = 1$ if and only if sample j corresponds to class i and 0 otherwise. This approach offers the advantage that the loss can be rewritten as a new Sparse Coding problem by incorporating the discriminative term into the reconstruction objective:

$$\|\mathbf{X} - \mathbf{BS}\|_F^2 + \beta\|\mathbf{H} - \mathbf{WS}\|_F^2 = \|\tilde{\mathbf{X}} - \tilde{\mathbf{B}}\tilde{\mathbf{S}}\|_F^2,$$

where

$$\tilde{\mathbf{X}} = [\mathbf{X}; \sqrt{\beta}\mathbf{H}], \quad \tilde{\mathbf{B}} = [\mathbf{B}; \sqrt{\beta}\mathbf{W}].$$

The tuning parameter β determines the relative importance of the discriminative term. Discriminative terms other than linear regression can also be added

Property	SC [39]	TSC [42]	DSC [4]	SDDL[61]	**Proposed**
Invariance:					
minimizing differences	N	Y	N	N	**Y**
modeling differences	N	N	N	Y	**Y**
Selectivity:					
stable encoding	N	N	Y	N	**Y**
supervision	N	N	Y	Y	**Y**

Table 1.1: Comparison of the Sparse Coding methods from the literature and the method developed in this thesis based on their ability to learn features that are invariant with respect to domain differences and selective for the classification task. The methods TSC, DSC and SDDL each offer an improvement over traditional SC, but simultaneously suffer from specific drawbacks that are resolved by the proposed method.

to the SC loss [3] [41]. The resulting optimization problem cannot be rewritten as a new Sparse Coding problem. Solving them requires iterating between three steps: optimizing the basis, optimizing the coefficients and optimizing the classifier parameters.

In summary, including label information in the Sparse Coding procedure improves the usefulness of the sparse coefficients for classification purposes and there exist various methods for doing so. The most promising approaches directly optimize the basis for the classification task. However, these methods are not designed to perform well when the training and test data are sampled from different probability distributions. This begs the question how to combine methods for transfer Sparse Coding with those that are optimized for classification problems.

1.4 Contributions of this thesis

This thesis makes two main contributions. First, it introduces a method for transfer Sparse Coding that can be directly optimized for classification. Second, it derives an efficient procedure for solving the resulting optimization problem. Domain invariance of the model is increased by building on existing methods for transfer Sparse Coding outlined in Section 1.2. This way, the model is designed to combine the desirable properties of existing Sparse Coding models (Table 1.1).

Specifically, the proposed method transforms the data to a shared subspace using domain-specific linear projections $\mathbf{P}_S, \mathbf{P}_T$. The reconstruction loss is therefore defined by

$$\|\mathbf{P}_S\mathbf{X}_S - \mathbf{B}\mathbf{S}_S\|_F^2 + \|\mathbf{P}_T\mathbf{X}_T - \mathbf{B}\mathbf{S}_T\|_F^2,$$

where $\mathbf{X}_S$ $\mathbf{X}_T$ are the samples of the source and target domain respectively and

$\mathbf{S}_S, \mathbf{S}_T$ are the corresponding coefficients. A similar idea is incorporated in the SDDL model from Shekhar et. al. (equation (1.3)), but their transformation is not encouraged to actually minimize the domain differences. The proposed model achieves this by penalizing the mean of the sample differences using a regularization term analogous to (1.2):

$$\left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{P}_S \mathbf{x}_i - \frac{1}{n_t} \sum_{i=n_s+1}^{n_s+n_t} \mathbf{P}_T \mathbf{x}_i \right\|_2^2 .$$

Simultaneously with learning the transformation, the proposed model learns a basis that can be used to represent the transformed data. This is done using KL-regularized Sparse Coding introduced in the DSC model (Equation (1.6)). This offers the advantage over L_1-regularized Sparse Coding or Sparse Coding with a hard sparsity constraint as employed by Shekhar et. al., of a stable encoding of the data. Moreover, differentiability of the loss means that the basis and the projection can be optimized for the classification task by end-to-end training. Domain invariance of the sparse representations of both datasets is increased by penalizing the difference between the sample means of the sparse coefficients (Equation (1.2)). This way, the model reduces domain differences at different stages.

In addition to combining these complementary elements that lead to domain invariance and task selectivity, I derive an efficient method for supervised training of the introduced model. This method exploits tensor operations to eliminate the need for sequential optimization that slowed down previous algorithms. Optimization of all model parameters −coefficients, basis and transformation− is quite challenging due to non-convexity, an issue faced by all methods based on Sparse Coding. However, experimental results show that the proposed model can be optimized such that it outperforms existing methods for transfer and discriminative Sparse Coding.

The remainder of this thesis is organized as follows. Chapter 2 places approaches to transfer Sparse Coding in a broader context by discussing the larger field of transfer learning. Chapter 3 presents a detailed treatment of the methods that are directly related to the proposed model. The mathematical derivation and the efficient optimization of the proposed model are presented in Chapter 4. Chapter 5 is devoted to the experimental evaluation of the model and Chapter 6 draws conclusions.

Chapter 2

Transfer Learning

The Sparse Coding models discussed in this thesis are part of the broader fields of transfer learning and representation learning. This chapter presents both fields. First, Section 2.1 discusses the motivation for transfer learning. Section 2.2 introduces the definitions and notations that are used to formalize the problem of transfer learning. Next, I will present a structured overview of the transfer learning literature. Section 2.3 is devoted to different scenarios in which one may want to apply transfer learning. Section 2.4 presents different approaches to transfer learning. Finally, Section 2.5 provides an overview of representation learning from the perspective of Sparse Coding and transfer learning.

2.1 Motivation

Conventional methods for learning from data are based on the assumption that the data on which they are applied has the same properties as the data on which they are trained. This is true both of theory [34], [64] and applications [24]. Moreover, current state-of-the-art methods require training on vasts amounts of labeled data [37]. However, in many real-world applications such as medical imaging [63], this data might be prohibitively expensive or simply impossible to acquire. This gap between theory and practice has become one of the major obstacles in successful application of statistical learning algorithms [54]. Transfer learning aims to solve this problem by allowing the training and test data to be related but different [54], [65]. Specifically, the data may be sampled from different distributions, represented using different features or be associated with different tasks. The main idea is using labeled data from a related domain in order to improve the performance of the algorithm on the domain of interest.

Notation	Description	Notation	Description
$\mathcal{X}$	Feature space	$\mathcal{Y}$	Label space
$P(\mathbf{x})$	Marginal Distribution over $\mathcal{X}$	$P(y\|\mathbf{x})$	Conditional distribution
$\mathcal{D}$	Domain	$\mathcal{T}$	Learning task
$\mathcal{D}_S, \mathcal{D}_T$	Source and target domain	$\mathcal{T}_S, \mathcal{T}_T$	Source and target task

Table 2.1: Summary of commonly used notation

2.2 Definitions

In this section, I will introduce the problem of transfer learning following the widely used definitions from several survey articles [54], [65]. Commonly used notation is summarized in Table 2.1. A *domain* is defined by a feature space $\mathcal{X}$ with a marginal probability distribution $P(\mathbf{x})$. Given a domain $\mathcal{D} = \{\mathcal{X}, P(\mathbf{x})\}$, a *task* $\mathcal{T}$ consists of a finite label space $\mathcal{Y}$ and a conditional distribution function $P(y|\mathbf{x})$. The data $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ are assumed to be independent samples from $P(\mathbf{x}, y)$. A typical choice for the feature space is $\mathcal{X} = \mathbb{R}^d$, a typical choice for the label space is $\mathcal{Y} = \{-1, +1\}$ (binary classification).

In *transfer learning* (TL), we are dealing with two domains and their related tasks: a *source* domain $\mathcal{D}_S = \{\mathcal{X}_s, P_S(\mathbf{X})\}$ with $\mathcal{T}_S = \{\mathcal{Y}_S, P_S(\mathbf{y}|\mathbf{X})\}$ and a *target* domain $\mathcal{D}_T = \{\mathcal{X}_T, P_T(\mathbf{X})\}$ with $\mathcal{T}_T = \{\mathcal{Y}_T, P_T(\mathbf{y}|\mathbf{X})\}$. When the two domains and their tasks are equal, traditional statistical learning methods can be used to learn $P(y|\mathbf{x})$ based on the labeled samples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_1, y_n)\}$. When this assumption does not hold, i.e. $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_T \neq \mathcal{T}_S$, a model trained on source data might not generalize well to target data, or simply cannot be applied when the feature sets or label sets do not agree. Assuming the source and target domain and task are different but somehow related, transfer learning algorithms aim to use the knowledge from $\mathcal{D}_S, \mathcal{D}_T$ and $\mathcal{T}_S$ in order to improve the learning of $P_T(y|\mathbf{X})$.

2.3 Scenarios: when to transfer

From the definition of transfer learning as a difference between domains or tasks, it is possible to categorize TL approaches based on the specific differences between source and target (Table 2.2). In *inductive* transfer learning, the target task is different from the source task. The domains do not have to be the same, but it is often assumed they are. In this case, at least some labeled target samples are required to learn the target predictive distribution $P_T(y|\mathbf{x})$.
In *transductive* transfer learning, the source and target tasks are the same, but the domains are different. When the feature spaces are shared but the marginal distributions $P(\mathbf{x})$ are different, we are dealing with *Domain Adaptation* (DA) [55]. This thesis focuses on exactly this situation. The categorization of transfer learning settings depends on the specific relationship between source and target

Setting	Domains	Tasks
Traditional	$\mathcal{D}_S = \mathcal{D}_T$	$\mathcal{T}_S = \mathcal{T}_T$
Inductive TL	$\mathcal{D}_S = \mathcal{D}_T$	$\mathcal{T}_S \neq \mathcal{T}_T$
Transductive TL	$\mathcal{D}_S \neq \mathcal{D}_T$	$\mathcal{T}_S = \mathcal{T}_T$

Table 2.2: Transfer learning (TL) can be categorized into different settings [54]

Transfer based on:	Description
Instances	Re-weigh or re-sample source instances
Features	Transform representation of source (and target) samples
Parameters	Share or adapt parameters optimized for source domain
Relational knowledge	Transfer semantic relationships between domains

Table 2.3: According to [54], approaches to transfer learning can be divided into four main categories.

data. Given a specific difference between source and target data set, the next question is which approach to transfer learning should be used.

2.4 Approaches to transfer learning

The survey of [54] distinguishes the following approaches: transfer of instances, transfer of features, transfer of parameters and transfer of relational knowledge. See Table 2.2 for an overview. For each of these approaches, I will discuss the basic idea, the underlying assumptions and several exemplar models. The approach most relevant to this thesis is feature based transfer, discussed in Section 2.4.2.

2.4.1 Instance-based transfer

Instance-based transfer approaches assume that even though the labeled source samples cannot be used directly, they can be used after re-weighting or re-sampling. Instance-based models have been proposed for both the inductive and the transductive case. In the transductive case, it is assumed that the domains share a single conditional probability distribution $P(y|\mathbf{x})$, but different marginal distributions $P(\mathbf{x})$ [58]. This difference is often estimated by the ratio $P_T(\mathbf{x})/P_S(\mathbf{x})$ [54]. Instance-based methods for inductive transfer weight the labeled source samples based on their influence on the target error. Wu and Dietterich [69] incorporated a weighted sum of source and target loss of a Support Vector Machine (SVM). Jian et. al. [32] develop a Natural Language Processing (NLP) algorithm for removing misleading source instances $(\mathbf{x}^s, y^s)$ from the data based on the difference between $P_T(y^s|\mathbf{x}^s)$ and $P_S(y^s|\mathbf{x}^s)$. Finally, Dai et. al. extended the AdaBoost algorithm [23] to TrAdaBoost [13], which it-

eratively decreases the weights of the source samples that negatively influence the target error. Simultaneously, the weight of misclassified target samples is increased as is done in the original AdaBoost algorithm.
Instance-transfer has the advantage of being relatively simple since it does not require learning a feature transformation. Naturally, this also means the scope of such methods is quite limited. For example, it is unlikely there exists a set of instance weights that corrects for different illumination intensities as observed in visual applications. Rather, this problem is the result from low-level transformations of the data.

2.4.2 Feature-based transfer

These approaches approaches aim to learn "good" features that simultaneously minimize the domain difference while being useful for the target task of interest. That is, they should be both *invariant* with respect to domain differences and *selective* for properties associated with class differences. Ben-David et. al. provide a theory of representations for domain adaptation [6], later extended to the case of multiple domains [50], [5]. A central result is an upper bound on the target generalization error of a classifier trained on labeled source data. The upper bound is determined by the empirical classification error on the source data and the difference between the representations of the two domains, measured by training a binary classifier to distinguish between source and target samples. The theory from Ben-David and colleagues therefore shows that a good representation is one which simultaneously achieves a low training error and low domain difference. The goal of learning a representation that trades off low error rate and high domain similarity is very intuitive, but it is also very general. It therefore leaves a lot of choice when designing an algorithm that is supposed to achieve this goal. Indeed, a plethora of feature-transfer methods have been proposed, some of which have been reviewed in Section 1.2.
Feature transfer can be applied to both the inductive and transductive case. Additionally, we can distinguish between methods that find a shared representation of both domains (symmetric transfer) and methods that find a mapping from one domain to the other (asymmetric transfer).

Recently, feature based transfer received a lot of attention because of current advances in learning representations from the data [7]. Relevant ideas from the field of representation learning will be discussed in Section 2.5.

2.4.3 Parameter-based transfer

The *Parameter-based* approach to transfer learning assumes that models trained on related datasets share a subset of their parameters [54] or the parameters of the classifier trained on the source domain can be adapted resulting in classifier that predicts the target labels. The latter approach is taken by the Adaptive SVM from Yang et. al. [70] that iteratively adjusts the decision boundary of the source classifier $f^s(\mathbf{x})$ using a sequence of functions Δf that gradually adjust

f^s based on classification error of the labeled target data:

$$f^t(\mathbf{x}) = f^s(\mathbf{x}) + \Delta f(\mathbf{x}).$$

Recently, this idea was combined with a deep neural network (DNN) for feature transfer [44] to handle a difference both in marginal and conditional distribution functions. While early methods used a two-stage procedure of (i) reducing the domain mismatch and (ii) classification, more recent methods for parameter-based transfer integrate these two steps. For example, The Domain Transfer SVM [18] simultaneously reduces the domain difference using a kernel-based distance function between pairs of samples and learns the decision function for classification of the target data. This idea was generalized in [17] by learning a convex combination of multiple kernels that minimize the domain divergence. This idea too, has been exploited in combination with recent methods for deep feature transfer [43]. Note that at this point, the distinction between feature and parameter based transfer starts to blur. Using multiple kernel learning to find similar representations for both domains might be considered as an instance of feature-based transfer. Conversely, transfer learning using artificial neural networks (NNs) often involves sharing layers and therefore parameters between domains.

2.4.4 Relational-knowledge transfer

Relational-knowledge transfer assumes that semantic relationships are presevered between the domains. For example, the relationship between a professor and a PhD student might be comparable to that between a CEO and an employee, suggesting the possibility of transfer from a text corpus on universities to one concerning business. By its very nature, relational-knowledge transfer seems tightly bound to the field of Natural Language Processing (NLP) with applications in statistical relational learning [14], [51] and sentiment analysis [40].

2.5 Learning representations

Data representation is a crucial factor in determining the success or failure of any machine learning system [15]. Ideally, the features correlate well with the learning task to the point that a linear predictor can be trained on top of the feature representation. Unfortunately, raw data is often in a form that is unsuitable for the task at hand. For example, the pixel values of an image are high-dimensional and do not correlate well with 'high-level' information relevant to tasks such as object recognition. While this shows the importance of representation learning, it is not clear *how* to learn a good representation. The reason is that in contrast to other aspects of machine learning such as optimization, there are no clear criteria for judging the quality of a learned representation. Nevertheless, various desirables for representations have been proposed, some of which are relevant for this thesis. They are usually formulated

in terms of 'factors' that generate the data. In case of Sparse Coding, these factors refer to the basis vectors.

- Smoothness [7]: similar inputs are mapped to similar representations, so $\phi(x) \approx \phi(y)$ when $x \approx y$. This way, the representation ϕ helps the predictor to generalize in small neighborhoods of the samples. This goal is rather subtle in the context of transfer learning since we might want the representation to be invariant with respect to domain specific properties, but sensitive to differences relevant to the predictive task.

- Sparsity [52]: for each observation, only a small number of all possible factors that generate the observation are relevant. This assumption is known to hold for visual data (images admit a sparse representation with respect to a wavelet basis). It is also convenient from a computational point of view, since a sparse representation makes learning associations easy [53] and increases memory capacity compared to a dense representation [68], [22].

- Shared factors across tasks [7], [30], suggesting the possibility for inductive transfer (Section 2.3). Ideally, the basis vectors from Sparse Coding can be used to represent samples from both domains.

- Disentangled representation [7], [67]. Different factors of variation combine in a complicated way to generate the data. In particular, nuisance and intrinsic variability can be entangled, diminishing the usefulness of the representation. Therefore, a representation which disentangles these factors might prove to be useful for transfer learning.

- Hierarchy of factors: high level factors result from combining low level factors. This property is relevant for transfer learning, since domain differences might occur at different levels of the hierarchy. The model introduced in this thesis exploits this property to a certain extent by reducing domain differences using a transformation followed by Sparse Coding.

nt These goals for a representation can be in opposition; simultaneously satisfying them involves trade-offs. Sparsity requires that only the most relevant factors are used to explain the data [52], inhibiting the activity of other factors. But this limits smoothness since a small change in the data might tip the balance of making the second best factor the single best, changing the representation. The method proposed in this thesis trades off sparsity with smoothness by using a differentiable regularization of the coefficients.
Regardless of the listed goals, a representation should first and foremost be useful for the predictive task. This explains why recent advances in representation learning are driven by end-to-end training of differentiable architectures that combine feature learning and classification. The method developed in this thesis exploits exactly this strategy by building on previous work outlined in Section 3.3.

Chapter 3

Sparse Coding Methods

The primary aim of this chapter is to introduce the four Sparse Coding methods that inspired the proposed model. Of particular interest will be the different ways in which the models aim to transfer between domains and in which they can be optimized for predictive performance. Additionally, I will discuss the algorithms used to optimize the model parameters. This issue matters because the Sparse Coding objective is non-convex in all parameters -basis and coefficients- simultaneously, making optimization quite challenging.
Section 3.1 introduces the unsupervised Sparse Coding (SC) model from [52]. Subsequently, I will discuss three different models which improve SC in complementary ways. Section 3.2 describes Transfer Sparse Coding (TSC) [42], a way of making SC more robust to domain differences. Section 3.3 discusses Differentiable Sparse Coding (DSC) [4]. In contrast to conventional Sparse Coding models, DSC uses a smooth sparsity penalty which offers the potential to make the learned features more discriminative. Section 3.4 discusses Shared Domain Adaptive Dictionary Learing (SDDL) from [61]. This combines Sparse Coding with a transformation of the data, potentially allowing for greater domain differences compared to TSC. I will conclude this chapter by summarizing the strengths and weaknesses of the four methods in Section 3.5.

The following notation will be used to formalize Sparse Coding. Bold upper case letters denote matrices, bold lower case denotes vectors. For example, the matrix of samples is given by $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$. The $\mathbf{x}_i$ will interchangeably be referred to as samples and images since Sparse Coding has been particularly successful in computer vision problems. Matrix entries are indicated using subscripts, so $\mathbf{X} = (X_{d,k})_{d,k}$, vector entries are indicated using superscripts, $\mathbf{x}_i = (\mathbf{x}_i^{(k)})_k$. Greek letters correspond to tuning parameters, specifically λ and μ refer to the sparsity and transfer parameter respectively.

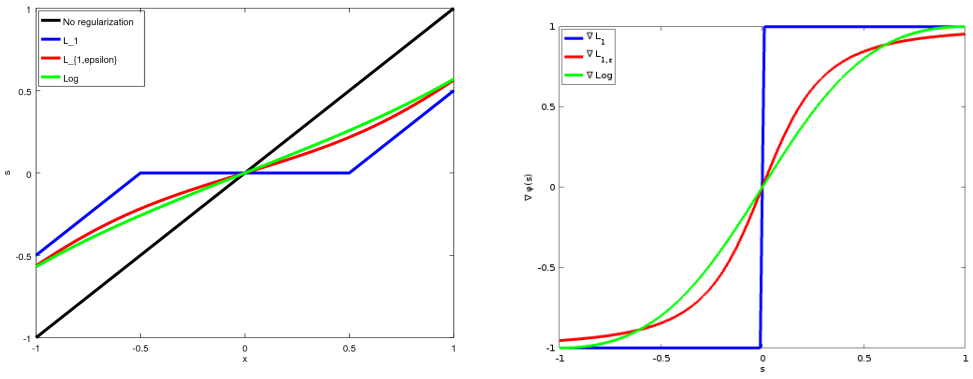

Figure 3.1: L_1-regularization leads to the sparsest coefficients, alternative penalties provide a smooth approximation. Left: optimal coefficients given input values x encoded using the identity as a basis. The L_1-norm thresholds the coefficients such that for small x, the optimal coefficient is 0. The $L_{1,\varepsilon}$ and log-penalty shrink s in a smooth way without thresholding. Right: the gradients of the regularization functions differentially suppress coefficients near zero during gradient-based optimization.

3.1 Sparse Coding (SC)

The goal of Sparse Coding is to represent the input vectors $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ as a linear combination of a small number of vectors from a dictionary or basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_K]$. The coefficients with respect to the basis are given by $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$, with $\mathbf{s}_i \in \mathbb{R}^K$. The number of basis vectors is allowed to exceed the of the input data ($K > d$, if $\mathbf{x} \in \mathbb{R}^d$). This *over-completeness* leads to greater flexibility in using $\mathbf{B}$ to encode the data [52]. An input image $\mathbf{x}$ is therefore approximated as $\mathbf{x} \approx \sum_k \mathbf{b}_k s_k$, where $\mathbf{s}$ is a (sparse) vector of coefficients. The reconstruction error is measured by $\|\mathbf{x} - \sum_k \mathbf{b}_k s^{(k)}\|_2^2$. The coefficients $\mathbf{S}$ are *sparsified* by adding an appropriate penalty function ϕ to the reconstruction error. Frequently used choices for ϕ are [39]:

$$\phi(s) = \begin{cases} s & L_1 \text{ penalty;} \\ \sqrt{\varepsilon + s^2} & L_{1,\varepsilon} \text{ penalty} \\ \log(1 + s^2) & \text{log-penalty.} \end{cases} \tag{3.1}$$

The L_1-penalty leads to the most sparse coefficients, the $L_{1,\varepsilon}$ and log-penalty are smooth approximations. See Figure 3.1.

Combining the goals of a faithful and sparse representation leads to the

Sparse Coding objective

$$\min_{\mathbf{b}_1,\ldots,\mathbf{b}_k;\mathbf{s}} \ \|\mathbf{x} - \sum_k \mathbf{b}_k s^{(k)}\|_2^2 + \lambda \sum_k \phi(s^k), \tag{3.2}$$

$$\text{s.t.} \ \|\mathbf{b}_k\|_2^2 \leq c \quad \forall k = 1, \ldots, K. \tag{3.3}$$

Here $\lambda > 0$ is a tunable parameter trading off a low reconstruction error and sparsity. The maximum column norm of B can be tuned using c, which is typically set to 1. The norm constraints on the basis vectors are needed since there always exists a scaling of $\mathbf{B}$ and $\mathbf{s}$ that leaves the approximation $\sum_k \mathbf{b}_k s^{(k)}$ unchanged but makes $\mathbf{s}$ arbitrarily small. This would trivially minimize the sparsity penalty. Given an input matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, the objective (3.2) can be formulated for all samples simultaneously:

$$\min_{\mathbf{B},\mathbf{S}} \ \|\mathbf{X} - \mathbf{BS}\|_F^2 + \lambda \sum_{i=1}^{n} \sum_{k=1}^{K} \phi(S_{ik}), \tag{3.4}$$

$$\text{s.t.} \quad \sum_{1 \leq d \leq D} B_{dk}^2 \leq c \ \ \forall k. \tag{3.5}$$

After finding the optimal $\mathbf{B}^*$ and $\mathbf{S}^* = [\mathbf{s^*}_1, \ldots, \mathbf{s^*}_n]$, the coefficients can be used as feature vectors for training a classifier. Therefore, the usefulnesses of Sparse Coding as feature learning method crucially depends on the discriminative information contained in $\mathbf{S}$. There are no guarantees on the discriminative information contained in $\mathbf{S}$ since $\mathbf{S}$ is learned using a two-stage method, consisting of *unsupervised* feature learning followed by *supervised* training of a classifier (Figure 3.1). In principle, any classifier can be trained on the sparse coefficients $\mathbf{S}$ to predict class labels $\mathbf{y} = [y_1, \ldots, y_n]$ corresponding to the input $\mathbf{X}$. For the experiments (Chapter 5), I have chosen L_2-regularized multinomial logistic regression.

3.1.1 Optimization

For the remainder of this section, the L_1-norm will be used as sparsity penalty ϕ. Given this choice, the objective (3.4) is convex in each of the optimization parameters $\mathbf{B}$ and $\mathbf{S}$ separately, but it is not convex in both parameters simultaneously. Most approaches for optimizing the objective therefore alternate between optimizing $\mathbf{B}$ while holding $\mathbf{S}$ fixed and optimizing with respect to $\mathbf{S}$ while holding $\mathbf{B}$ fixed. Learning $\mathbf{B}$ given $\mathbf{S}$ is a least squares problem with quadratic constraints. This can be solved using any convex optimization algorithm, gradient descent or block-coordinate gradient descent with iterative projections or solving the dual problem. Learning $\mathbf{S}$ given $\mathbf{B}$ is an L_1-regularized least squares problem. This can be solved using e.g. an interior point method [11] or an adapted version of least angle regression [20] I have used the algorithm proposed by Lee et. al. [39] because it can easily be adapted to solve the Transfer Sparse Coding problem (Section 3.2).

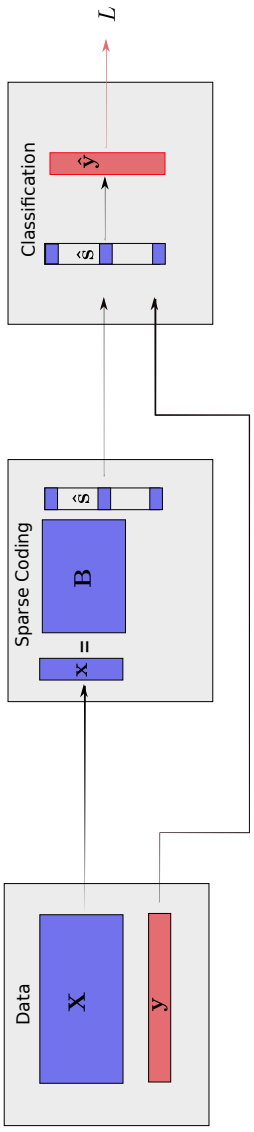

Figure 3.2: Sparse Coding learns a basis $\mathbf{B}$ and coefficients $\mathbf{S}$ that reconstruct the input data $\mathbf{X}$. The optimal coefficients $\hat{\mathbf{S}}$ can subsequently be used to train a classifier. This is a two-stage process; classification does not influence feature learning.

Learning the coefficients

The problem of optimizing $\mathbf{S}$ given $\mathbf{B}$ is given by

$$\min_{\mathbf{S}} \|\mathbf{X} - \mathbf{BS}\|_F^2 + \lambda \sum_{i,k} \phi(S_{ik}).$$

and can be solved for each coefficient vector separately since given $\mathbf{B}$, the optimal values of different coefficients are independent. Given a single input image $\mathbf{x}$, the objective with respect to the corresponding coefficient vector $\mathbf{s}$ is equal to

$$\min_{\mathbf{s}} \ f(\mathbf{s}), \tag{3.6}$$

where f is the SC loss

$$f(\mathbf{s}) = \|\mathbf{x} - \sum_k \mathbf{b}_k s^{(k)}\|_2^2 + \lambda \sum_k |s^{(k)}|. \tag{3.7}$$

The crucial observation motivating the FSS algorithm is that if we knew the signs of the optimal $\mathbf{s}^*$, the problem would reduce to a quadratic programming problem which can be solved analytically. Therefore, the FSS algorithm iterates between searching for the signs of $\mathbf{s}$ and solving the resulting QP given the current signs. It can be shown that the algorithm converges to an optimal solution in a finite number of steps [39].

Determining the optimal sign vector given the current coefficients is the most involved part of the algorithm since this concerns optimizing a non-differentiable function. Let

$$g(\mathbf{s}) = \|\mathbf{x} - \mathbf{Bs}\|_2^2$$

be the differentiable part of the loss function. Since the loss is convex, a necessary and sufficient condition for $\mathbf{s}$ to be a global optimum of f is that the zero vector is contained in the subgradient, the set of subderivatives of f. Let $\nabla^{(k)}$ denote the subgradient with respect to the k-th entry of $\mathbf{s}$. The variable $\theta \in \mathbb{R}^K$ will be used to keep track of the optimal signs of $\mathbf{s}$.
For $s^{(k)} \neq 0$, $|s^{(k)}|$ is differentiable and hence its only subderivative is the derivative: $\nabla^{(k)}|s^{(k)}| = \text{sign}(s^{(k)})$. For $s^{(k)} = 0$, $\nabla^{(k)} g(\mathbf{s}) = [-1, 1]$. Hence, the optimality conditions for the full loss f are given by

$$\begin{cases} \nabla^{(k)} g(\mathbf{s}) + \lambda \text{sign}(s^{(k)}) = 0 & \text{if } s^{(k)} \neq 0; \\ \nabla^{(k)} g(\mathbf{s}) \in [-\lambda, \lambda] & \text{if } s^{(k)} = 0. \end{cases}$$

Note that in both situations, 0 is indeed an element of the subdifferentiable. In case the optimality conditions do not hold, the signs of $\mathbf{s}$ have to be updated in order to improve the objective function f. It makes intuitive sense how this should be done. If $s^{(k)} \neq 0$ but $\nabla^{(k)} g(\mathbf{s}) + \lambda \text{sign}(s^{k)}) \neq 0$, the loss will decrease by letting $s^{(k)} = 0$. Therefore consider how to update $s^{(k)}$ in case $s^{(k)} = 0$ but the optimality condition does not hold. If $\nabla^{(k)} g(\mathbf{s}) > \lambda$, it must be true that $\nabla^{(k)} f(\mathbf{s}) > 0$ since $\nabla^{(k)}|s^{(k)}| < 1$. This means that decreasing $s^{(k)}$ will lower the

value of f, so let $\theta^{(k)} = -1$. Analogously, if $\nabla^{(k)} g(\mathbf{s}) < -\lambda$, taking $\theta^{(k)} = 1$ and therefore increasing $\mathbf{s}$ will improve the objective value. In the activation step, the algorithm greedily selects the entry which potentially improves the objective the most. This entry will become active, i.e. non-zero. This determines how to select the optimal signs $\mathbf{s}$. Having determined θ, it is given which entries of $\mathbf{s}$ are 0. What is left to do is determining the specific values of the non-zero entries. Hence given θ, the SC optimization problem reduces to a problem involving only the non-zero entries of s. Given θ, (3.6) is equivalent to $\min_{\mathbf{s}_\mathcal{A}} f(\mathbf{s}_\mathcal{A})$, where $\mathcal{A}$ is the *active set* of nonzero entries in $\mathbf{s}$:

$$\min_{\mathbf{s}_\mathcal{A}} \|\mathbf{x}_\mathcal{A} - \mathbf{B}_\mathcal{A}\mathbf{s}_\mathcal{A}\|_2^2 + \lambda\theta_\mathcal{A}^T\mathbf{s}_\mathcal{A} \tag{3.8}$$

Here $\mathbf{B}_\mathcal{A}$ denotes the matrix consisting of the columns k of $\mathbf{B}$ such that $k \in \mathcal{A}$ and $\mathbf{x}_\mathcal{A}$ denotes the vector with entries of $\mathbf{x}$ indexed by $\mathcal{A}$. This problem has a global minimum that can be found by solving $\partial f/\partial \mathbf{s}_\mathcal{A} = 0$. Subsequently, we can update the nonzero entries of $\mathbf{s}$. The zero entries remain unchanged in this step.

Learning the basis

Optimizing with respect to the basis $\mathbf{B}$ while treating $\mathbf{S}$ as fixed reduces the optimization problem (3.4) to

$$\min_{\mathbf{B}} \ \|\mathbf{X} - \mathbf{B}\mathbf{S}\|_F^2 \tag{3.9}$$

$$\text{s.t.} \ \sum B_{ij}^2 \leq c \ \ \forall j. \tag{3.10}$$

I will now discuss the algorithm proposed by Lee et. al. [39] that finds the optimal basis by solving the Lagrange dual. For dual variables $\lambda_j \geq 0$, the Lagrangian of (3.9) is given by

$$L(\mathbf{B}, \lambda) = \text{Tr}((\mathbf{X} - \mathbf{B}\mathbf{S})^T(\mathbf{X} - \mathbf{B}\mathbf{S})) + \sum_k \lambda_k \sum_i (B_{ik}^2 - c).$$

At the optimal basis $\hat{\mathbf{B}}$, the gradient should vanish:

$$-\mathbf{X}\mathbf{S}^T + \mathbf{S}\mathbf{S}\hat{\mathbf{B}} + \mathbf{S}\mathbf{S}^T\hat{\mathbf{B}} + \Lambda\hat{\mathbf{B}} = 0,$$

so $\hat{\mathbf{B}} = (\mathbf{S}\mathbf{S}^T + \Lambda)^{-1}(\mathbf{X}\mathbf{S}^T)$, where $\Lambda = \text{diag}(\lambda_k)$. Substituting $\hat{\mathbf{B}}$ gives the Lagrange dual

$$D(\Lambda) = \min_{\mathbf{B}} L(\mathbf{B}, \lambda) = \text{Tr}(\mathbf{X}^T\mathbf{X} - \mathbf{X}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T + \Lambda)^{-1})(\mathbf{X}\mathbf{S}^T)^T - c\Lambda).$$

Optimizing with respect to Λ can be done using a second order method like Newton's method. The required gradients are given by

$$\frac{\partial D}{\partial \lambda_i} = \|\mathbf{X}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T + \Lambda)^{-1}\mathbf{e}_i\|^2 - c,$$

$$\frac{\partial D}{\partial \lambda_i \partial \lambda_j} = -2\left((\mathbf{S}\mathbf{S}^T + \Lambda)^{-1}(\mathbf{X}\mathbf{S}^T)^T\mathbf{X}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T + \Lambda)^{-1}\right)_{i,j}\left((\mathbf{S}\mathbf{S}^T + \Lambda)^{-1}\right)_{i,j}.$$

Given the optimal dual variables Λ, the optimal basis is given by

$$\mathbf{B}^T = (\mathbf{S}\mathbf{S}^T + \Lambda)^{-1}(\mathbf{X}\mathbf{S}^T)^T.$$

The described approach has the desirable property that the number of dual variables depends on the number of basis vectors, but not on the number of samples. In addition, the algorithm does not depend on the regularization of $\mathbf{S}$ since only B is optimized. It can therefore be used in combination with an arbitrary regularization function of $\mathbf{S}$, such as the transfer penalty used by TSC.

3.2 Transfer Sparse Coding (TSC)

The Sparse Coding method described achieves satisfying results in different machine learning tasks, it is not designed for transfer learning problems. When faced with images following different distributions, the sparse representation of these images will be different as well. Therefore, a classifier trained on the representations from one data set will not generalize well to representations from a different dataset. This section discusses a solution to this challenge proposed by Long et. al. [42].
Their method is designed to perform transductive transfer learning (Chapter 2), in which the marginal distribution $P(\mathbf{x})$ differs between the domains, but the conditional distribution $P(y|\mathbf{x})$ is shared. Assume we are given labeled source data $D_S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_s}, y_{n_s})\}$ and unlabeled target data $D_T = \{\mathbf{x}_{n_{s+1}}, \ldots, \mathbf{x}_{n_s+n_t}\}$ where the images from both datasets follow different probability distributions. The naive approach would be to learn a single basis that is used to represent the source and target samples together with corresponding coefficient matrices $\mathbf{S}_s$ and $\mathbf{S}_t$.

However, the latent representation by the sparse coefficients will follow a different distribution as well, rendering it suboptimal as image descriptor classification purposes. Long et. al. tackle this problem by explicitly reducing the domain differences between the source and target coefficients. This is done by extending the SC objective with the difference between the means of $\mathbf{S}_s$ and $\mathbf{S}_t$:

$$\left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{s}_i - \frac{1}{n_t} \sum_{i=n_s+1}^{n_s+n_t} \mathbf{s}_i \right\|_2^2 = \mathrm{Tr}(\mathbf{S}\mathbf{M}\mathbf{S}^T). \tag{3.11}$$

Here $\mathbf{M} = (M_{ij})_{1 \leq i,j \leq n}$ is the matrix with entries given by

$$M_{ij} = \begin{cases} 1/n_s^2 & \text{if } 1 \leq i, j \leq n_s; \\ 1/n_t^2 & \text{if } n_s + 1 \leq i, j \leq n_s + n_t; \\ -1/(n_s n_t) & \text{otherwise} \end{cases} \tag{3.12}$$

The Transfer Sparse Coding (TSC) objective is therefore equal to

$$L(\mathbf{X}, \mathbf{S}, \mathbf{B}) = \|\mathbf{X} - \mathbf{B}\mathbf{S}\|_2^2 + \lambda\|\mathbf{S}\|_1 + \mu\mathrm{Tr}(\mathbf{S}\mathbf{M}\mathbf{S}^T), \tag{3.13}$$

with μ a tunable parameter determining the importance of the transfer penalty. Long et. al. [42] show that adding this regularization improves the performance of SC on different domain adaptation problems. The experimental results from Chapter 5 show that this improvement can be very modest.

3.2.1 Optimization

Analogous to the procedure for SC, the optimization of the objective consists of two iterative steps. Since the transfer regularization (3.11) does not depend on $\mathbf{B}$, optimization of the basis can be done using the dual algorithm. Optimization of the sparse codes can be done using an adapted version of the FSS algorithm from Section 3.1.1. For a single sample $\mathbf{x}_i$, the objective (3.13) equals

$$f(s_i) = g(s_i) + \lambda \sum_k |s_i^{(k)}|, \tag{3.14}$$

where the differentiable function g now is defined as

$$g(\mathbf{s}_i) = \|\mathbf{x}_i - \mathbf{B}\mathbf{s}_i\|_2^2 + \mu \sum_{j=1}^{n} M_{ij} \mathbf{s}_i^T \mathbf{s}_j. \tag{3.15}$$

Given its gradient

$$\frac{\partial g}{\partial \mathbf{s}_i} = -2\mathbf{B}^T(\mathbf{x}_i - \mathbf{B}\mathbf{s}_i) + 2\mu M_{ii}\mathbf{s}_i + \mu \sum_{j=1}^{n} M_{ij}\mathbf{s}_i,$$

Algorithm 2 (p. 34) learns transferable sparse codes by minimizing f.

3.3 Differentiable Sparse Coding (DSC)

Conventional Sparse Coding methods are able to learn useful features by combining an unsupervised reconstruction objective with a sparsity penalty. However, the usual L_1 regularization leads to a unstable relationship between the data and the coefficients. This means that small variations in the input data can lead to large variations in the latent representation. Bagnell and Bradley therefore proposed Differentiable Sparse Coding (DSC) [4] that uses a smooth regularization function instead of the L_1-penalty. This differentiable regularization function is the unnormalized Kullback-Leibler divergence, previously been used to solve inverse problems in different applied sciences [36]. KL-regularization is known to be particularly useful for comparing sparse vectors [72]. See Figure 3.3 for a comparison of the KL-divergence with other regularizers.

In DSC, KL regularization penalizes differences from a nonnegative prior vector $\mathbf{p}$. Since the sparse coefficients $\mathbf{s}$ are not required to sum to 1, the unnormalized KL-divergence is used. This is defined by

$$KL(\mathbf{s}\|\mathbf{p}) = \sum_k \left(s^{(k)} \log \frac{s^{(k)}}{p^{(k)}} - s^{(k)} + p^{(k)} \right). \tag{3.16}$$

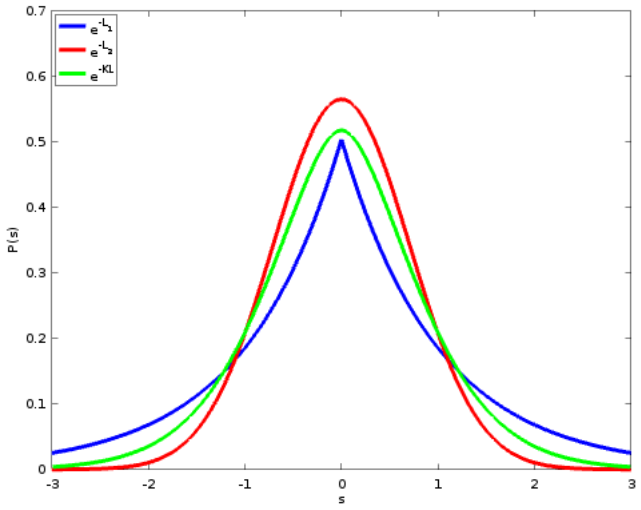

Figure 3.3: From a probabilistic perspective, a regularizer $\phi(s)$ is equivalent to a certain prior distribution $P(s)$ over the coefficients. This figure shows that the prior corresponding to the unnormalized KL-divergence offers a smooth compromise between a Normal prior and a Laplacian prior corresponding to L_2 and L_1 regularization.

Note that in principle any vector $\mathbf{p}$ could be used in order to impose a certain structure to the coefficients. However, Bagnell et. al. propose to use a uniform prior vector $\mathbf{p} = (p, \ldots, p)$ whose L_1 magnitude Kp equals the expected L_1-norm of the coefficients. As p approaches 0, the regularization becomes more similar to the usual L_1 prior. Note that KL-regularization does not encourage sparse solutions in the usual sense of the word. Rather, it leads to "pseudo-sparse" coefficients from which most entries do not deviate from a fixed value p very much. This corresponds to the goal of learning feature vectors from which most elements can be replaced by a given value without a significant decrease in the classification performance. Combining reconstruction error with KL-regularization results in the DSC loss function

$$L(\mathbf{s}, \mathbf{B} \| \mathbf{p}) = \frac{1}{2} \|\mathbf{x} - \mathbf{B}\mathbf{s}\|_2^2 + \lambda \sum_k \left(s^{(k)} \log \frac{s^{(k)}}{p} - s^{(k)} + p \right). \tag{3.17}$$

3.3.1 Optimization: unsupervised learning

During optimization, the basis is first optimized according to the unsupervised loss (3.17), subsequently, it can be fine-tuned to become more discriminative (Figure 3.4). This section is devoted to the unsupervised learning procedure, summarized in Algorithm 4 on p. 36. The next section presents the supervised

learning procedure. Analogous with other methods for Sparse Coding, the approach for optimizing the DSC loss (3.17) optimizes $\mathbf{B}$ and $\mathbf{S}$ by iteratively optimizing the basis given the coefficients and vice versa.
Learning the coefficient matrix $\mathbf{S}$ can be done for each $\mathbf{s}_i$ separately, since the optimal values for different samples do not depend on each other given the dictionary. Optimization of the coefficients is done using the Exponentiated Gradient Descent (EGD) algorithm, since this is known to perform efficient feature selection [35] as is needed when dealing with over-complete dictionaries. The EGD update rule for the coefficients $\mathbf{s}$ reads

$$\mathbf{s} = \mathbf{s} \circ \exp\left(-\alpha \frac{\partial L}{\partial \mathbf{s}}\right),$$

where $\circ$ denotes the entrywise matrix product. The learning rate α is determined using backtracking line search [10] (Algorithm 3, p. 35). In case L is the DSC loss (3.17), the gradient is equal to

$$\frac{\partial L}{\partial \mathbf{s}} = \mathbf{B}^T(\mathbf{B}\mathbf{s} - \mathbf{x}) + \lambda \log \frac{\mathbf{s}}{p}. \tag{3.18}$$

Optimization of the basis is done using Stochastic Gradient Descent (SGD). This means that rather than computing the gradient over all training samples, the gradient

$$\frac{\partial L}{\partial \mathbf{B}} = (\mathbf{B}\mathbf{s} - \mathbf{x})\mathbf{s}^T.$$

for a random sample $\mathbf{x}$ is computed. Subsequently, $\mathbf{B}$ is updated by taking a step in the negative descent direction:

$$\mathbf{B} = \mathbf{B} - \eta \frac{\partial L}{\partial \mathbf{B}}.$$

In practice, mini-batches $[\mathbf{x}_1, \ldots, \mathbf{x}_{n_b}]$ of $n_b > 1$ samples will be used instead of a single sample. This is a classical heuristic in stochastic gradient descent algorithms.

3.3.2 Optimization: Supervised learning

The original Sparse Coding model is unsupervised since it is aimed at modeling the statistics of the input and does not depend on class labels (see Figure 3.1). However, the current state-of-the-art in feature learning heavily depends on exploiting large labeled training sets for end-to-end training. Therefore, tuning the features based on label information is of paramount importance to the usefulness of a feature learning method. In contrast to the Sparse Coding models discussed in previous sections, Differentiable Sparse Coding can be used for supervised fine-tuning of the basis using backpropagation.

In Sparse Coding, samples are represented in terms of the basis, meaning the usefulness of the sparse coefficients as feature vectors is determined by the discriminative properties of the basis. Concretely, the specific combination of basis vectors used to reconstruct a sample should be informative about the class label of this image. Unfortunately, directly optimizing the basis to minimize the classification loss is not possible, since the loss is determined by the coefficients but not by the basis. However, the values of $\hat{\mathbf{S}}$ are determined by the basis, establishing an implicit relationship between the supervised loss and the $\mathbf{B}$. This relationship can be made explicit using the dependency of the both $\mathbf{B}$ and the loss L_{sup} on the coefficients $\mathbf{S}$.

Concretely, suppose that a classifier is trained to predict the class labels $\mathbf{y}$ based on the sparse coefficients $\mathbf{S}$. Let $L_{sup}(\hat{\mathbf{s}}, y, \mathbf{W})$ denote the loss incurred when predicting the class label y using the features $\mathbf{s}$ and classifier parameters $\mathbf{W}$. As usual, the gradients $\partial L_{sup}/\partial \mathbf{W}$ are used to find a value of $\mathbf{W}$ that minimizes L_{sup} over all training samples. The coefficients are used as features and are kept fixed at this stage. Instead, we would like to adjust $\mathbf{B}$ in order to minimize L_{sup}. Since L_{sup} does not depend on $\mathbf{B}$ explicitly, we cannot compute $\partial L_{sup}/\partial \mathbf{B}$ directly. However, this can be done using chain rule of calculus

$$\frac{\partial L_{sup}}{\partial \mathbf{B}} = \frac{\partial L_{sup}}{\partial \hat{\mathbf{s}}} \cdot \frac{\partial \hat{\mathbf{s}}}{\partial \mathbf{B}} \tag{3.19}$$

After classification, we can directly compute the first gradient on the right hand side. What is left to do is computing the second gradient $\partial \hat{\mathbf{s}}/\partial \mathbf{B}$. This is done using implicit differentiation of the unsupervised loss function.

At the coefficients $\hat{\mathbf{s}}$ that minimize the unsupervised Sparse Coding loss (3.17), the gradient of this loss L vanishes, that is $\partial L/\partial \hat{\mathbf{s}} = 0$. This is equivalent to

$$\frac{\partial L_{rec}}{\partial \hat{\mathbf{s}}} = -\lambda \frac{\partial KL}{\partial \hat{\mathbf{s}}},$$

where KL is the unnormalized KL-regularization defined by (3.16). or by (3.18):

$$(\mathbf{B}\hat{\mathbf{s}} - \mathbf{x})\hat{\mathbf{s}}^T = -\lambda \log \frac{\hat{\mathbf{s}}}{p}$$

In order to obtain $\partial \hat{\mathbf{s}}/\partial \mathbf{B}$, we again take the derivative, this time with respect to the (d,k)th entry of $\mathbf{B}$:

$$\frac{\partial}{\partial B_{dk}} \frac{\partial L}{\partial \hat{\mathbf{s}}} = -\lambda \frac{\partial KL}{\partial B_{dk}} \frac{\partial L}{\partial \mathbf{s}}.$$

This equality implicitly defines the relationship between the optimal coefficients $\hat{\mathbf{s}}$ and the basis. The partial derivatives are given by

$$\begin{aligned}
\frac{\partial}{\partial B_{dk}} \frac{\partial L}{\partial \hat{\mathbf{s}}} &= \mathbf{J}^{kd} \left(\mathbf{B}\hat{\mathbf{s}} - \mathbf{x}\right) - \mathbf{B}^T \mathbf{J}^{dk} \hat{\mathbf{s}} \\
\frac{\partial KL}{\partial B_{dk}} \frac{\partial L}{\partial \hat{\mathbf{s}}} &= -\operatorname{diag}\left(\frac{\lambda}{\hat{\mathbf{s}}}\right) \frac{\partial \hat{\mathbf{s}}}{\partial \mathbf{B}}.
\end{aligned}$$

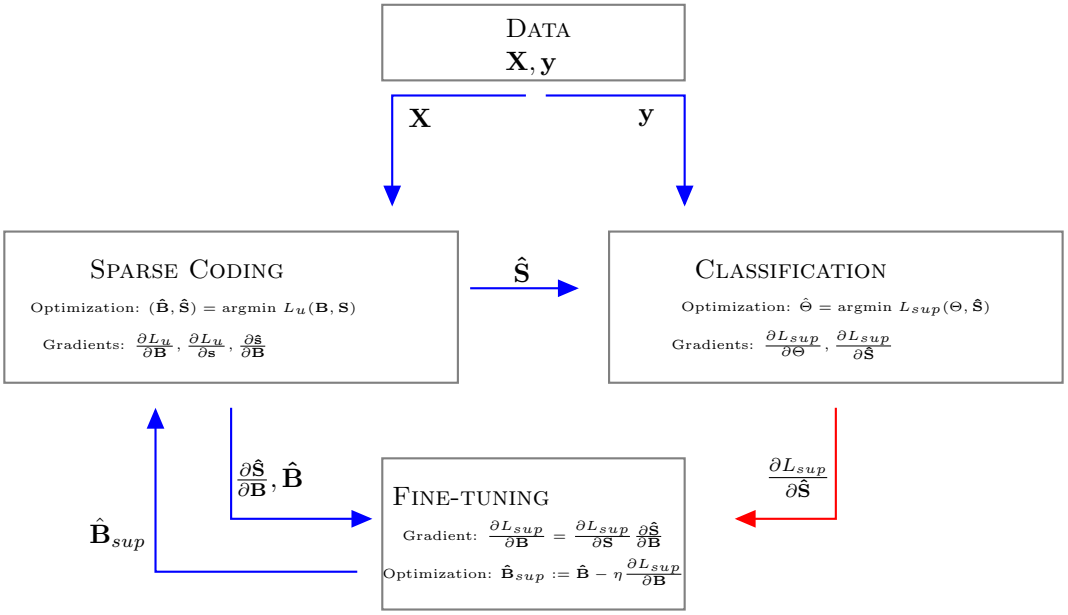

Figure 3.4: Different stages of the DSC model. Black arrows indicate forward flow of data, red arrows indicate (backward flow of) a supervised error signal. The Sparse Coding and Transformation stage both receive $\partial L/\partial \mathbf{S}$ in order to update the basis $\mathbf{B}$ and transformation matrix $\mathbf{P}$, respectively. Progressive increase in uniformity of colors of both domain indicates reduction in domain differences, achieved by transfer penalties (dotted vertical lines).

Here $\mathbf{J}^{kd}$ denotes the $K \times D$ matrix with only zeros, except for the (k,d)-th entry which is equals 1. Collecting terms and solving for $\partial\hat{\mathbf{s}}/\partial B_{dk}$ gives

$$\left(\mathbf{B}^T\mathbf{B} + \operatorname{diag}(\frac{\lambda}{\hat{\mathbf{s}}})\right)\frac{\partial\hat{\mathbf{s}}}{\partial\mathbf{B}_{dk}} = -\mathbf{J}^{kd}(\mathbf{B}\hat{\mathbf{s}} - \mathbf{x}) - \mathbf{B}^T\mathbf{J}^{dk}\hat{\mathbf{s}}, \tag{3.20}$$

$$\frac{\partial\hat{\mathbf{s}}}{\partial B_{dk}} = -\left(\mathbf{B}^T\mathbf{B} + \operatorname{diag}(\frac{\lambda}{\hat{\mathbf{s}}})\right)^{-1} \times \tag{3.21}$$

$$\left(\mathbf{B}^T\mathbf{J}^{kd}\hat{\mathbf{s}} + \mathbf{J}^{dk}(\mathbf{B}\hat{\mathbf{s}} - \mathbf{x})\right). \tag{3.22}$$

With this expression, (3.19) can be used to compute $\partial L_{sup}/\partial B_{dk}$. Subsequently, this is used to update the basis by SGD:

$$\mathbf{B} := \mathbf{B} - \alpha\frac{\partial L_{sup}}{\partial\mathbf{B}}.$$

Note that computation of $\partial\hat{\mathbf{S}}/\partial\mathbf{B}$ requires $\partial\hat{\mathbf{s}}/\partial B_{dk}$ for all samples and basis entries (d,k). In Chapter 4, I will derive an explicit expression for $\partial\hat{\mathbf{S}}/\partial\mathbf{B}$ that facilitates a much more efficient approach.

3.4 Shared Domain Adaptive Dictionary Learning (SDDL)

I will now describe the final Sparse Coding model which projects data from different domains to a shared subspace. This approach is hypothesized to complement Transfer Sparse Coding, which explicitly reduces but does not model differences between domains. There are multiple models that combine dimension reduction and Sparse Coding (see e.g. [26], [29]), but I will focus on the method proposed by Shekhar et. al., since this is designed to perform transfer learning.

The SDDL method uses source data $\mathcal{D}_S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_s}, y_{n_s})\}$ and target data $\mathcal{D}_S = \{(\mathbf{x}_{n_s+1}, y_{n_s+1}), \ldots, (\mathbf{x}_{n_s+n_t}, y_{n_s+n_t})\}$. For ease of notation I assume that both source and target data are d-dimensional. In general the dimensionality is allowed to be different for both datasets since the projections are domain-specific. In contrast to TSC (Section 3.2), it is assumed that class labels are available for not just the source, but also the target data. Based on the combined source and target datasets, SDDL learns a basis $\mathbf{B} \in \mathbb{R}^{d\times K}$ along with sparse coefficients $\mathbf{S}_s = [\mathbf{s}_1, \ldots, \mathbf{s}_{n_s}]$ and $\mathbf{S}_t = [\mathbf{s}_{n_s+1}, \ldots, \mathbf{s}_{n_s+n_t}]$. Importantly, the method also learns projection matrices $\mathbf{P}_s$ and $\mathbf{P}_t$ that map the source and target data to a common space. This shared representation is aimed to contain less domain-specific features. Therefore, it can be chosen to have lower dimension than the original data. After projecting the data upon the subspace, it is reconstructed using the dictionary $\mathbf{B}$. The reconstruction loss is therefore defined as

$$L_{rec}(\mathbf{X}_s, \mathbf{X}_s, \mathbf{S}_s, \mathbf{S}_t\mathbf{P}_s, \mathbf{P}_t, \mathbf{B}) = \|\mathbf{P}_s\mathbf{X}_s - \mathbf{B}\mathbf{S}_s\|_F^2 + \|\mathbf{P}_t\mathbf{X}_t - \mathbf{B}\mathbf{S}_t\|_F^2$$

In this framework, the projections and coefficients are domain specific, but the dictionary is shared. After transforming the data, it should be possible to recover at least part of the original image. This is achieved by including the following regularization term, that requires the projections to be close to orthogonal in the direction of the data:

$$L_{retain}(\mathbf{X}_s, \mathbf{X}_t, \mathbf{P}_s, \mathbf{P}_s) = \|\mathbf{X}_s - \mathbf{P}_s^T \mathbf{P}_s \mathbf{X}_s\|_F^2 + \|\mathbf{X}_t - \mathbf{P}_t^T \mathbf{P}_t \mathbf{X}_t\|_F^2.$$

The combined objective can be written more compactly as

$$\begin{aligned} L(\mathbf{X}, \mathbf{P}, \mathbf{B}, \mathbf{S}) &= L_{rec}(\mathbf{X}, \mathbf{P}, \mathbf{B}, \mathbf{S}) + \lambda_r L_{retain}(\mathbf{X}, \mathbf{P}) \\ &= \|\mathbf{P}\mathbf{X} - \mathbf{B}\mathbf{S}\|_F^2 + \lambda_r \|\mathbf{X} - \mathbf{P}^T \mathbf{P}\mathbf{X}\|_F^2, \end{aligned}$$

where $\mathbf{P} = [\mathbf{P}_s, \mathbf{P}_t], \mathbf{S} = [\mathbf{S}_s, \mathbf{S}_t]$ and $\mathbf{X}$ is the block diagonal matrix

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_s & 0 \\ 0 & \mathbf{X}_t \end{pmatrix}.$$

SDDL has two important additional properties Shekhar et. al. adapted from the literature. First, the projections are required to be orthonormal. This prevents collapsing some of the dimensions of the subspace. Moreover, this constraint makes it possible to use a previously developed approach for efficient optimization of orthogonal matrices [66]. Secondly, the discriminative power of the basis is increased using the method of Fisher Discriminative Dictionary Learning [71]. The basis is partitioned into a collection of class-specific smaller bases $\mathbf{B} = [\mathbf{B}_1, \ldots, \mathbf{B}_C]$ where C denotes the number of classes. The following term encourages reconstruction of samples using the basis elements corresponding to its label and penalizes using out-of-class elements:

$$L_{class}(\mathbf{X}, \mathbf{P}, \mathbf{B}, \mathbf{S}) = \mu \|\mathbf{P}\mathbf{X} - \mathbf{B}\mathbf{S}_{in}\|_F^2 + \nu \|\mathbf{B}\mathbf{S}_{out}\|_F^2.$$

The matrices $\mathbf{S}_{in}$ and $\mathbf{S}_{out}$ are defined by

$$(\mathbf{S}^{in})_{l,k} \begin{cases} S_{l,k} & \text{if } \mathbf{b}_i, \mathbf{x}_k \text{ belong to same class} \\ 0 & \text{otherwise.} \end{cases}$$

and

$$(\mathbf{S}^{out})_{l,k} \begin{cases} S_{l,k} & \text{if } \mathbf{b}_i, \mathbf{x}_k \text{ belong to different classes} \\ 0 & \text{otherwise.} \end{cases}$$

The SDDL optimization problem is given by

$$\min_{\mathbf{B},\mathbf{P},\mathbf{S}} \quad L_{rec}(\mathbf{X}, \mathbf{P}, \mathbf{B}) + L_{class}(\mathbf{X}, \mathbf{P}, \mathbf{B}, \mathbf{S}) + \lambda L_{retain}(\mathbf{X}, \mathbf{P}, \mathbf{S}) \tag{3.23}$$

$$\text{s.t.} \quad \mathbf{P}\mathbf{P}^T = \mathbf{I}, \ \ \|\mathbf{s}_i\|_0 \leq t \ \ \forall i = 1, \ldots, n_s + n_t \tag{3.24}$$

The parameters μ, ν (contained in L_{class}), λ and t are tunable. Here, sparsity of the coefficients is enforced using a hard constraint instead of the L_1-penalty.

These two approaches to Sparse Coding are related since the the minimal L_1 solution is also the sparsest solution for most underdetermined systems [16]. Solving (3.23) is done using an iterative approach. In comparison to the optimization over a basis and coefficient matrix, a third step is added in which the projection is optimized while holding $\mathbf{B}$ and $\mathbf{S}$ fixed. This is done by first proving that the optimal projection is if of a particular form. This form is determined by the eigendecomposition of $\mathbf{X}^T\mathbf{X}$ and the solution of an optimization problem with orthogonality constraints. The latter problem is non-convex but can be solved using a method for optimization on the Stiefel manifold of orthonormal matrices [66]. Optimizing the basis is done using an algorithm for discriminative dictionary learning, proposed in [71]. Finding the optimal coefficients given $\mathbf{P}$ and $\mathbf{B}$ can be solved using Orthogonal Matching Pursuit (OMP) [56], a greedy algorithm that iteratively builds up t-element coefficients.

3.5 Comparing the models

Having discussed the four models, I will now compare their respective advantages and disadvantages. I will argue that TSC, DSC and SDDL each posses properties that are highly complementary. These can be interpreted from the goal to learn features that are both invariant and selective. On one hand, a model should enable transfer by learning features that are *invariant* with respect to domain differences. On the other hand, the features should be *selective* for task relevant properties of the data so they are useful for classification. This is summarized in Table 3.1 (p. 38).

Sparse Coding is able to learn features that are useful for image classification and other machine learning tasks. However, its performance deteriorates when the learned dictionary is used to encode images from different domains. This is because the sparse representation of input sampled from different probability distributions will follow different distributions as well. That is, the representation is not invariant with respect to domain differences. Transfer Sparse Coding (TSC) therefore improves upon SC by explicitly minimizing differences in the representation of different domains, but fails to transform samples from different domains such that they can be encoded using a single dictionary. This means that it represents data with different statistics using similar sparse coefficients with respect to the same basis. Shared Domain-adaptive Dictionary Learning (SDDL) alleviates this problem by projecting the data to a shared subspace. However, the obtained representations are not explicitly regularized towards domain-invariance as is done in TSC.

Leaving the challenges from transfer learning aside, we can compare the discussed methods by their ability to learn discriminative features. The methods SC, TSC and SDDL suffer from a non-smooth way of inducing sparsity. This leads to an unstable relationship between the data and the latent representation, thereby diminishing the usefulness of this representation for image classification.

Moreover, the conventional Sparse Coding model is unsupervised, meaning the learned features are not optimized for the downstream task. SDDL incorporates class labels but it does not directly optimize the basis using the supervised information. Instead, a heuristic approach in used in which dictionary atoms are forced to specialize to a given class. This might become exceedingly problematic with a large number of classes. These challenges are overcome by Differentiable Sparse Coding (DSC), which learns a stable encoding of the data and offers the possibility to directly optimize the basis for the task at hand. However, DSC lacks any mechanism to cope with domain differences.

In summary, the three methods SC, DSC and SDDL posses complementary properties (Table 3.1). Therefore, it should be possible to combine them in way that eliminates their weak points while retaining the favorable properties, resulting in a method capable of learning invariant and selective features. The next chapter will introduce the proposed method that is designed to do exactly this.

Input: Data matrix $\mathbf{X}$, basis $\mathbf{B}$, sparsity regularization parameter λ
Output: Optimal coefficient matrix $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$
for $i \in [1, \ldots, n]$ **do**
  begin initialize
    $\mathbf{s}_i := \mathbf{0}$, active set $\mathcal{A} = \varnothing$ and $\theta := \mathbf{0}$ where $\theta^{(k)} \in \{-1, 0, 1\}$ denotes $\text{sign}(s_i^{(k)})$.
  end
  begin Activate
    From zero coefficients of $\mathbf{s}_i$, select $k := \text{argmax}_k |\nabla^{(i)} g(\mathbf{s}_i)|$.
    Activate $s_i^{(k)}$ only if it locally improves the objective $f(\mathbf{s}_i)$:
    If $\nabla^{(k)} g(\mathbf{s}_i) > \lambda$, then set $\theta^{(k)} = -1$;
    If $\nabla^{(k)} g(\mathbf{s}_i) < -\lambda$, then set $\theta^{(k)} = 1$.
    In both cases: $\mathcal{A} := \mathcal{A} \cup \{k\}$.
  end
  begin Feature-sign search
    Compute the solution to the unconstrained QP (3.8) given by the solution to $\partial f(\mathbf{s}_{i,\mathcal{A}})/\partial \mathbf{s}_{i,\mathcal{A}} = \mathbf{0}$:

$$\mathbf{s}_{i,\mathcal{A}}^{new} := \left(\mathbf{B}_{\mathcal{A}}^T \mathbf{B}_{\mathcal{A}}\right)^{-1} \left(\mathbf{B}_{\mathcal{A}}^T \mathbf{x}_{i,\mathcal{A}} - \lambda \theta_{\mathcal{A}}/2\right).$$

    Check the objective value at $\mathbf{s}_{i,\mathcal{A}}^{new}$ and all other points between $\hat{\mathbf{s}}_\mathbf{i}$ and $\mathbf{s}_{i,\mathcal{A}}^{new}$ where any coefficient sign changes. Update $\mathbf{s}_{i,\mathcal{A}}$ to the point with the lowest objective value, remove zero coefficients of $\mathbf{s}_{i,\mathcal{A}}^{new}$ from $\mathcal{A}$ and set $\theta := \text{sign}(\mathbf{s}_i)$.
  end
  begin Check optimality conditions
    1. Optimality condition for nonzero coefficients:
       $\nabla^{(k)} g(\mathbf{s}_i) + \lambda \text{sign}(s_i^{(k)} = 0 \; \forall k : s_i^{(k)} \neq 0$. If $\exists k$ for which condition does not hold go to "Feature-sign search".
    2. Optimality condition for zero coefficients: $|\nabla^{(k)} g(\mathbf{s}_i)| \leq \lambda \; \forall k : \mathbf{s}_i^{(k)} = 0$. If $\exists k$ for which condition does not hold go to "Activate".
  end
end

Algorithm 1: Feature-sign search algorithm, adapted from Lee et. al. [39]

Input: Data matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, basis $\mathbf{B} \in \mathbb{R}^{d \times K}$, transfer matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$, sparsity/transfer parameters $\lambda, \mu \in \mathbb{R}$

Output: Optimal coefficient matrix $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$

for $i \in [1, \ldots, n]$ **do**

begin Initialize

$\mathbf{s}_i := \mathbf{0}, \theta := \mathbf{0}$, and active set $\mathcal{A} = \varnothing$, where $\theta^{(k)} \in \{-1, 0, 1\}$ denotes $\text{sign}(s_i^{(k)})$.

end

begin Activate

From zero coefficients of $\mathbf{s}_i$, select $k := \text{argmax}_k |\nabla^{(i)} g(\mathbf{s}_i)|$, where g is defined in (3.15). Activate $s_i^{(k)}$ only if it locally improves the objective $f(\mathbf{s}_i)$.

If $\nabla^{(k)} g(\mathbf{s}_i) > \lambda$, then set $\theta^{(k)} = -1$;

If $\nabla^{(k)} g(\mathbf{s}_i) < -\lambda$, then set $\theta^{(k)} = 1$.

In both cases: $\mathcal{A} := \mathcal{A} \cup \{k\}$.

end

begin Feature-sign search

Update active coefficients:

$$\mathbf{s}_{i,\mathcal{A}}^{new} := \left(\mathbf{B}_{\mathcal{A}}^T \mathbf{B}_{\mathcal{A}} + \mu M_{ii} \mathbf{I}\right)^{-1} \left(\mathbf{B}_{\mathcal{A}}^T \mathbf{x}_{i,\mathcal{A}} - \lambda/2\theta_{\mathcal{A}} + \sum_{j \neq i} \mu M_{ij} \mathbf{s}_j\right),$$

so that

$$\mathbf{s}_{i,\mathcal{A}}^{new} = \text{argmin}_{\mathbf{s}_{i,\mathcal{A}}} \|\mathbf{x}_{i,\mathcal{A}} - \mathbf{B}_{\mathcal{A}} \mathbf{s}_{i,\mathcal{A}}\|_2^2 + \mu \sum_{j=1}^{n} M_{ij} \mathbf{s}_{i,\mathcal{A}}^T \mathbf{s}_{j,\mathcal{A}} + \lambda \sum_{k \in \mathcal{A}} \theta^{(k)} s_i^{(k)}.$$

Check the objective value at $\mathbf{s}_{i,\mathcal{A}}^{new}$ and the other points between $\hat{\mathbf{s}}_i$ and $\mathbf{s}_{i,\mathcal{A}}^{new}$ where any coefficient sign changes. Update $\mathbf{s}_{i,\mathcal{A}}$ to the point with the lowest objective value, remove zero coefficients of $\mathbf{s}_{i,\mathcal{A}}^{new}$ from $\mathcal{A}$ and set $\theta := \text{sign}(\mathbf{s}_i)$.

end

begin Check optimality conditions

1. Optimality condition for nonzero coefficients: $\nabla^{(k)} g(\mathbf{s}_i) + \lambda \text{sign}(s_i^{(k)}) = 0 \; \forall k : s_i^{(k)} \neq 0$. If $\exists k$ for which condition does not hold go to "Feature-sign search".

2. Optimality condition for zero coefficients: $|\nabla^{(k)} g(\mathbf{s}_i)| \leq \lambda \; \forall k : \mathbf{s}_i^{(k)} = 0$. If $\exists k$ for which condition does not hold go to "Activate".

end

end

Algorithm 2: Feature-sign search algorithm for TSC, adapted from Long et. al. [42]

Input: Loss function $f : \mathbb{R}^d \to \mathbb{R}$, gradient ∇f, current value of optimization parameter $\theta_0 \in \mathbb{R}^d$, initial guess α, decay factor $\beta \in (0.1, 0.8)$.
Output: Optimal learning rate α.
begin Initialization
  Largest possible parameter update:

$$\theta_{new} := \theta_0 \circ \exp\left(-\alpha \nabla f\right).$$

end
begin Decrease updates until heuristic met
  while $f(\theta_{new}) > f(\theta_0) - \alpha/2 \|\nabla f(\theta_0)\|_2^2$ **do**
    Decrease learning rate: $\alpha := \alpha \cdot \beta$, try smaller update:

$$\theta_{new} := \theta_0 \circ \exp\left(-\alpha \nabla f(\theta_0)\right).$$

  end
end

Algorithm 3: Backtracking line search, adapted from [10].

Input: Data matrix $\mathbf{X} \in \mathbb{R}^{D \times n}$, i $\lambda \in \mathbb{R}$ (sparsity regularization parameter), $p \in \mathbb{R}$ (prior), number of iterations $iter$, minibatch size n_b.

Output: Optimal coefficient matrix $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$, basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_K]$.

begin optimization

  for $i \in [1, \ldots, iter]$ **do**

    Sample a mini-batch $\mathbf{X}_b = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_b}]$ from X. Let $\mathbf{S}_b$ denote the corresponding coefficients.

    for $b \in [1, \ldots, B]$ **do**

      Repeatedly update mini-batch coefficients using EGD:

      for $i_s \in [1, \ldots, egdIter]$ **do**

        Determine learning rate α using backtracking line search (Algorithm 3), update:

$$\mathbf{S}_b := \mathbf{S}_b \circ \exp\left(-\alpha \frac{\partial L}{\partial \mathbf{S}_b}\right)$$

      end

      so that (approximately) $\mathbf{S}_b = \operatorname{argmin}_{\mathbf{s}_b} f(\mathbf{S}_b)$.

      Set learning rate $\eta \propto 1/\sqrt{i}$, update basis using SGD rule:

$$\mathbf{B} := \mathbf{B} - \eta \frac{\partial L}{\partial \mathbf{B}}.$$

    end

  end

end

Algorithm 4: DSC algorithm using EGD for optimizing the coefficients and SGD for optimizing the basis.

	SC [39]	TSC [42]	DSC [4]	SDDL[61]
Invariance:				
minimizing differences	N	Y	N	N
modeling differences	N	N	N	Y
Selectivity:				
Stable encoding	N	N	Y	N
Supervision	N	N	Y	Y

Table 3.1: Comparison of the Sparse Coding methods discussed in this chapter, based on their ability to learn features that are invariant with respect to domain differences and selective for classification task. The methods TSC, DSC and SDDL each offer an improvement over SC, but simultaneously suffer from specific drawbacks.

Chapter 4

Proposed Model

This chapter introduces on the proposed method that is inspired by the algorithms discussed in Chapter 3. As we have seen, each of these algorithms has complementary strengths and weaknesses. The proposed model is therefore designed to combine the strengths of Transfer Sparse Coding (Section 3.2), Differentiable Sparse Coding (Section 3.3) and Shared Domain-Adaptive Dictionary Learning (Section 3.4). Transfer learning is done using a two-fold approach. Data from different domains are transformed to a shared representation. Here, Sparse Coding with a transfer-penalty is performed to minimize domain differences. The discriminative power of the features is enhanced by using a differentiable sparsity penalty. This offers the advantage of a smooth relationship between the input data and the latent representation. More importantly, such a penalty makes the loss function differentiable. Differentiability is exploited to make the model more useful for prediction by end-to-end training.
Section 4.1 introduces the model components and shows how they are combined. This requires formulating several constraints on the basis and projection as a soft-constraint to facilitate backpropagation. Section 4.2 concerns unsupervised learning of the model. Section 4.3 derives a method for tuning the model to become more discriminative. The method optimizes over full matrices and all samples simultaneously.

4.1 The model

The proposed model reduces domain differences in two stages (Figure 4.1) by learning a projection matrix $\mathbf{P}$, a basis $\mathbf{B}$ and sparse coefficient matrix $\mathbf{S}$. The data available to the model is divided into a source and a target dataset. The source data is given by $\{\mathbf{X}_s, \mathbf{y}_s\}$, where $\mathbf{X}_s = [\mathbf{x}_1, \ldots, x_{n_s}]$ is the input samples for which class labels $\mathbf{y}_s = [y_1, \ldots, y_{n_s}]$ are available. The target data $\mathbf{X}_t = [\mathbf{x}_{n_s+1}, \ldots, \mathbf{x}_{n_s+n_t}]$ is sampled from a different probability distribution. Moreover, I assume that no class labels are available for the target data (transductive transfer, Chapter 2).

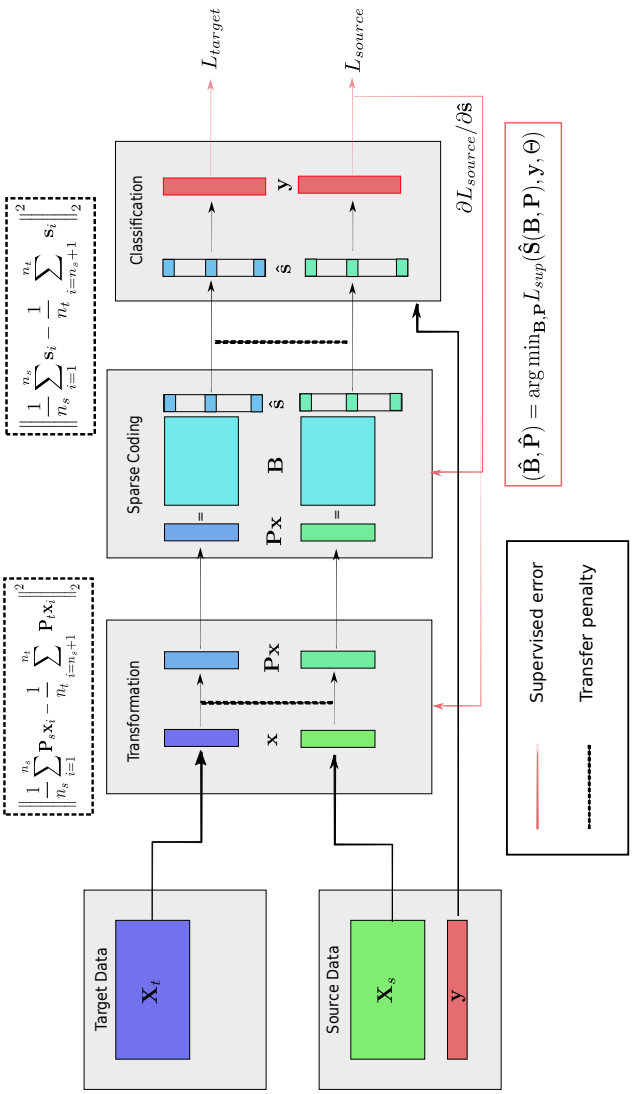

Figure 4.1: Domain differences are reduced in two stages. First by transforming the data to a shared subspace using domain-specific projections, then by Sparse Coding the data using a shared dictionary. Subsequently a classifier is trained on the sparse coefficients of the source data. Since the target coefficients follow a similar distribution, the classifier can be expected to generalize from the source to the target domain at test time.

The proposed model is motivated by the aim of achieving simultaneously the following goals. Since most parts are adapted from the methods discussed in Chapter 3, they will briefly be explained here.

- The basis $\mathbf{B}$ and matrix $\mathbf{S}$ should faithfully reconstruct the transformed data $\mathbf{PX}$, such that

  $$L_{rec}(\mathbf{X}, \mathbf{P}, \mathbf{B}, \mathbf{S}) = \frac{1}{2}\|\mathbf{PX} - \mathbf{BS}\|_F^2.$$

  is small. Recall the notation introduced in Section 3.5 for combining the domain-specific projection matrices $\mathbf{P} = [\mathbf{P}_s, \mathbf{P}_t]$ that each map the samples from one domain to a shared subspace. This corresponds to the reconstruction loss as measured in the SDDL framework (Section 3.4).

- Mapping the projected data back to the original space should result in an image close to the original data. This means

  $$L_{retain}(\mathbf{X}, \mathbf{P}) = \frac{1}{2}\|\mathbf{X} - \mathbf{P}^T\mathbf{PX}\|_F^2.$$

  should be small. Here $\|\cdot\|_F^2$ denotes the Frobenius matrix norm. This term is equal to the 'signal fidelity penalty' in the SDDL model (Section 3.4).

- While SDDL uses a hard orthogonality constraint, we wish to design a differentiable and unconstrained loss function since this allows supervised tuning of the model by backpropagation (Section 4.3). Therefore the projection matrix is required to be *close* to orthogonal using the following soft constraint:

  $$L_{ortho}(\mathbf{P}) = \frac{1}{4}\|\mathbf{I} - \mathbf{PP}^T\|_F^2.$$

  This prevents the projection matrix from collapsing dimensions of the shared subspace. In contrast to SDDL, including this (soft) constraint is not strictly necessary for optimizing the parameters.

- In the SDDL framework, there is no guarantee that transforming the data will minimize the differences between data from different domains. Therefore, in the proposed model differences in the shared subspace are explicitly penalized by matching the first moments of the transformed data:

  $$L_{sim}(\mathbf{P}, \mathbf{X}) = \frac{1}{2}\mathrm{Tr}((\mathbf{PX})\mathbf{M}(\mathbf{PX})^T).$$

  Here $\mathbf{M} = (M_{ij})_{1 \le i,j \le n}$ is defined in Equation (3.12).

- Using an analogous regularization for the coefficients should increase the similarity between the sparse representations. This amounts to including the following regularization that penalizes the difference in first moments between the sparse coefficients of the source and target domain:

  $$L_{sim}(\mathbf{S}) = \frac{1}{2}\mathrm{Tr}(\mathbf{SMS}^T),$$

This is equal to the transfer-penalty employed by Transfer Sparse Coding (Section 3.2).

- The transformation should preserve the local geometry of the data. This is achieved by incorporating Graph-Laplacian regularization [75]. Let $\mathbf{W} \in \mathbb{R}^{n \times n}$ be the k-nearest neighbor matrix of the data:

$$W_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i, \mathbf{x}_j \text{ are } k\text{-nearest neighbours;} \\ 0 & \text{otherwise.} \end{cases}$$

The degree of sample i in the graph matrix is $g_i = \sum_j W_{ij}$, the sum of the weights of the edges connected to it. Let $\mathbf{G}$ be the diagonal matrix with diagonal elements g_i, then the graph Laplacian is defined by

$$\mathbf{L} = \mathbf{G} - \mathbf{W}.$$

Including the the regularizer

$$L_{graph}(\mathbf{PX}) = \frac{1}{2} \sum_{i,j} W_{ij} \|\mathbf{P}\mathbf{x}_i - \mathbf{P}\mathbf{x}_j\|_2^2 = \frac{1}{2} \mathrm{Tr}((\mathbf{PX})\mathbf{L}(\mathbf{PX})^T)$$

in the objective penalizes similar samples being transformed to dissimilar representations.

- The coefficients should be sparse, in the sense that most entries should be close to a single value p. Following Differentiable Sparse Coding (Section 3.3), 'sparsity' is measured using the unnormalized KL-divergence:

$$L_{reg}(\mathbf{S} \| p) = \sum_{\mathbf{s}_1, \ldots, \mathbf{s}_n} \sum_k \left(s_i^{(k)} \log \frac{s_i^{(k)}}{p} - s_i^{(k)} + p \right).$$

In contrast to L_1 regularization, this results in a smooth encoding of the data.

- A soft constraint on the L_2-norms of the basis vectors is imposed. This is done in order to prevent the optimization procedure from scaling $\mathbf{B}$ and $\mathbf{S}$ such that the sparsity penalty becomes arbitrarily small, but the reconstruction $\mathbf{BS} \approx \mathbf{X}$ remains unchanged:

$$\begin{aligned} L_{col}(\mathbf{B}) &= \sum_k \left(\|\mathbf{b}_k\|_2^2 - 1 \right)^2 \\ &= \|\mathrm{diag}\left(\mathbf{B}^T\mathbf{B} - \mathbf{1}_{D \times 1} \right)\|_F^2 \end{aligned}$$

This is in contrast to other approaches for dictionary learning ([39], [45], [52]), which optimize non-differentiable loss functions. In principle any of these methods could be used for unsupervised learning $\mathbf{B}$ after incorporation of the data transformation $\mathbf{P}$. Supervised tuning of $\mathbf{B}$, however, requires a differentiable loss that can be optimized using online learning.

Combining the various parts, we arrive at the following loss function

$$L(\mathbf{X},\mathbf{P},\mathbf{B},\mathbf{S}) = L_{data}(\mathbf{X},\mathbf{P},\mathbf{B},\mathbf{S}) + \lambda_B L_{col}(\mathbf{B}) + \tag{4.1}$$
$$\lambda_s L_{sparsity}(\mathbf{S}\|p) + \mu L_{transfer}(\mathbf{S}) + \gamma_P L_{graph}(\mathbf{PX}) \tag{4.2}$$
$$\lambda_r L_{retain}(\mathbf{X},\mathbf{P}) + \lambda_o L_{ortho}(\mathbf{P}) + \mu_P L_{transfer}(\mathbf{PX}), \tag{4.3}$$

where $\lambda_B, \lambda_s, \mu, \gamma_P, \lambda_r, \lambda_o$ and μ_P are tunable parameters determining the relative importance of the different terms. In practice, λ_B and λ_o will be chosen large enough such that the the corresponding soft constraints become negligible (see experimental results in Chapter 5). Since the other tuning parameters should be chosen by cross-validation.

4.2 Unsupervised learning

Unsupervised learning is done by finding the parameters $\mathbf{S}, \mathbf{B}, \mathbf{P}$ that minimize (4.1). Like other Sparse Coding methods such as those discussed in Chapter 3, the loss is not convex in all parameters simultaneously. Convergence to the global minimizer $(\mathbf{S}^*, \mathbf{B}^*, \mathbf{P}^*)$ is therefore not guaranteed. Both EGD and SGD use the gradients of the loss to determine the optimal descent direction. The gradients are computed over a randomly sampled mini-batch of size $n_b \geq 1$. The gradient with respect to the coefficients $\mathbf{S}$ of a mini-batch is given by:

$$\frac{\partial L}{\partial \mathbf{S}} = \frac{\partial L_{data}}{\partial \mathbf{S}} + \lambda_s \frac{\partial L_{sparsity}}{\partial \mathbf{S}} + \mu \frac{\partial L_{transfer}}{\partial \mathbf{S}} \tag{4.4}$$
$$= \mathbf{B}^T(\mathbf{BS} - \mathbf{PX}) + \lambda_s \log\left(\frac{\mathbf{S}}{p}\right) + \mu(\mathbf{SM}^T + \mathbf{SM}). \tag{4.5}$$

Here $\log \cdot / p$ refers to the entry-wise operation, so

$$\left\{\log \frac{\mathbf{S}}{p}\right\}_{i,j} = \log \frac{S_{i,j}}{p}.$$

The gradient with respect to the basis is equal to

$$\frac{\partial L}{\partial \mathbf{B}} = \frac{\partial L_{data}}{\partial \mathbf{B}} + \lambda_n \frac{\partial L_{col}}{\partial \mathbf{B}} \tag{4.6}$$
$$= (\mathbf{BS} - \mathbf{PX})\mathbf{S}^T + \lambda_B \mathbf{B} \circ \operatorname{diag}(\mathbf{B}^T\mathbf{B} - \mathbf{1}_{D\times 1}), \tag{4.7}$$

where $\operatorname{diag}(\mathbf{B}^T\mathbf{B} - \mathbf{1}_{D\times 1})$ refers to the diagonal matrix with non-zero entries corresponding to the diagonal entries of $\mathbf{B}^T\mathbf{B} - \mathbf{1}_{D\times 1}$. The gradient with respect

to the projection matrix equals

$$\frac{\partial L}{\partial \mathbf{P}} = \frac{\partial L_{data}}{\partial \mathbf{P}} + \lambda_r \frac{\partial L_{retain}}{\partial \mathbf{P}} \tag{4.8}$$

$$+ \lambda_o \frac{\partial L_{ortho}}{\partial \mathbf{P}} + \mu_P \frac{\partial L_{sim}}{\partial \mathbf{P}} \tag{4.9}$$

$$+ \mu_P \frac{\partial L_{transfer}}{\partial \mathbf{P}} + \gamma_P \frac{\partial L_{graph}}{\partial \mathbf{P}} \tag{4.10}$$

$$= (\mathbf{PX} - \mathbf{BS})\mathbf{S}^T + \tag{4.11}$$

$$\lambda_r \left(-2\mathbf{PXX}^T + (\mathbf{PP}^T)\mathbf{P}(\mathbf{XX}^T) + \mathbf{P}(\mathbf{XX}^T)(\mathbf{P}^T\mathbf{P})\right) \tag{4.12}$$

$$+ \lambda_o(\mathbf{PP}^T - \mathbf{I})\mathbf{P}^T + \mu_P \left(\mathbf{PM}^T + \mathbf{PM}\right) \tag{4.13}$$

$$+ \mu_P(\mathbf{PXM}^T\mathbf{X}^T + \mathbf{PXMX}^T) + \gamma_P(\mathbf{PXL}^T\mathbf{X}^T + \mathbf{PXLX}^T). \tag{4.14}$$

Algorithm 5 uses these gradients to find the optimal parameters. The optimal learning rate for updating the coefficients is determined using backtracking line search. (Algorithm 3). The basis and projection are optimized using a batch algorithm Limited-Memory BFGS (L-BFGS [60]) since this led to faster convergence compared to the online setting (SGD, used by [4] for DSC).

4.3 Supervised Learning

After unsupervised learning of the model, the coefficients $\hat{\mathbf{S}} = [\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_n]$ serve as feature input to a classifier. The classifier is then trained to predict the class labels based on $\hat{\mathbf{S}}$. Mathematically, this amounts to finding the classifier parameters $\mathbf{W}$ that minimize the supervised loss $L_{sup}(\hat{\mathbf{s}}_i, y_i, \mathbf{W})$ of the classifier. Typically, the prediction will not be perfect and the gradient $\partial L_{sup}/\partial \mathbf{W}$ can be used to adjust $\mathbf{W}$ in the direction that decreases L_{sup}. Additionally, we would like to make the model more discriminative by adjusting the basis and the projection. Since classification is done based on $\hat{\mathbf{S}}$, but not on $\mathbf{B}$ and $\mathbf{P}$, the supervised gradients with respect to these parameters cannot be computed directly. Instead, this is done by backpropagating $\partial L/\partial \hat{\mathbf{S}}$ to the feature learning stage:

$$\frac{\partial L_{sup}}{\partial \mathbf{B}} = \frac{\partial L_{sup}}{\partial \mathbf{s}} \cdot \frac{\partial \mathbf{s}}{\partial \mathbf{B}},$$
$$\frac{\partial L_{sup}}{\partial \mathbf{P}} = \frac{\partial L_{sup}}{\partial \mathbf{s}} \cdot \frac{\partial \mathbf{s}}{\partial \mathbf{P}}.$$

The gradient $\partial L_{sup}/\partial \hat{\mathbf{S}}$ is available directly after classification, simply by differentiating the classifier loss. The goal of this section is the more involved computation of the gradients $\partial \hat{\mathbf{S}}/\partial \mathbf{B}$ and $\partial \hat{\mathbf{S}}/\partial \mathbf{P}$. For the DSC model, Section 3.3 describes the derivation of $\partial \hat{\mathbf{s}}_i/\partial B_{dk}$ of a *single* sample $\hat{\mathbf{s}}_i$ with respect to a *single* basis entry B_{dk}. As mentioned before, optimization using these sample and entry wise gradients is inefficient since it requires looping over both the samples i and the matrix entries (d, k). Therefore, I will derive an expression

for the gradient $\partial\hat{\mathbf{S}}/\partial\mathbf{B}$ of *multiple* samples with respect to the *full matrix* $\mathbf{B}$. Next, I will do the same for $\partial\hat{\mathbf{S}}/\partial\mathbf{P}$, used for fine-tuning the projection matrix. After unsupervised learning of $\mathbf{B}$ and $\mathbf{P}$ using Algorithm 5, these parameters are fine-tuned using the supervised gradients. This procedure is summarized in Algorithm 6 (p. 46).

4.3.1 Gradient with respect to the basis

The goal of this subsection is the derivation of $\partial\hat{\mathbf{S}}/\partial\mathbf{B}$. I will start by deriving the gradient $\partial\hat{\mathbf{S}}/\partial B_{dk}$ for all samples with respect to a single basis entry B_{dk}. The loss (4.1) associated with sample $i, i = 1, \ldots, n$ is given by:

$$L(\mathbf{x}_i, \mathbf{P}, \mathbf{B}, \mathbf{S}) = \frac{1}{2}\|\mathbf{P}\mathbf{x}_i - \mathbf{B}\mathbf{s}_i\|_2^2 + \frac{\lambda_r}{2}\|\mathbf{x}_i - \mathbf{P}^T\mathbf{P}\mathbf{x}_i\|_2^2 \tag{4.15}$$

$$+ \sum_k \left(\mathbf{s}_i^{(k)} \log \frac{\mathbf{s}_i^{(k)}}{p} - \mathbf{s}_i^{(k)} + p \right) \tag{4.16}$$

$$+ \mu_P \sum_{i,j=1}^{n} M_{ij}(\mathbf{P}\mathbf{x}_i)^T(\mathbf{P}\mathbf{x}_j) + \mu \sum_{i,j=1}^{n} M_{ij}\mathbf{s}_i^T\mathbf{s}_j \tag{4.17}$$

$$+ \gamma_P \sum_{i,j=1}^{n} L_{ij}(\mathbf{P}\mathbf{x}_i)^T(\mathbf{P}\mathbf{x}_j) \tag{4.18}$$

$$+ \frac{\lambda_o}{4}\|\mathbf{I} - \mathbf{P}\mathbf{P}^2\|_2^2 + \frac{1}{4}\|\mathrm{diag}\left(\mathbf{B}^T\mathbf{B} - \mathbf{1}_{D\times 1}\right)\|_F^2 \tag{4.19}$$

At the coefficient $\hat{\mathbf{s}}_i$ that minimizes the unsupervised loss of sample i, it holds that $\nabla_s L(\hat{\mathbf{s}}_i) = 0$. Hence,

$$\frac{\partial L_{data}}{\partial \hat{\mathbf{s}}_i} = -\lambda \frac{\partial L_{sparsity}}{\partial \hat{\mathbf{s}}_i} - \mu \frac{\partial L_{transfer}}{\partial \hat{\mathbf{s}}_i}, \tag{4.20}$$

$$\mathbf{B}^T(\mathbf{B}\hat{\mathbf{s}}_i - \mathbf{P}\mathbf{x}_i) = -\lambda \log \frac{\hat{\mathbf{s}}_i}{p} - \mu \sum_{j=1}^{n} M_{ij}\hat{\mathbf{s}}_j. \tag{4.21}$$

We wish to compute $\partial\hat{\mathbf{s}}_i/\partial B_{dk}$. This can be done using implicit differentiation of (3.18) with respect to B_{dk}:

$$\frac{\partial}{\partial B_{dk}}\left(\frac{\partial L_{data}}{\partial \hat{\mathbf{s}}_i}\right) = -\lambda \frac{\partial}{\partial B_{dk}}\left(\frac{\partial L_{sparsity}}{\partial \hat{\mathbf{s}}_i}\right) - \mu \frac{\partial}{\partial B_{dk}}\left(\frac{\partial L_{transfer}}{\partial \hat{\mathbf{s}}_i}\right), \tag{4.22}$$

The left-hand side equals:

$$\frac{\partial}{\partial B_{dk}}\left(\frac{\partial L_{data}}{\partial \hat{\mathbf{s}}_i}\right) = \mathbf{J}^{kd}(\mathbf{B}\hat{\mathbf{s}}_i - \mathbf{P}\mathbf{x}_i) + \mathbf{B}^T(\mathbf{J}^{dk}\hat{\mathbf{s}}_i + \mathbf{B}\frac{\partial \hat{\mathbf{s}}_i}{\partial B_{dk}} - \frac{\partial \mathbf{P}}{\partial B_{dk}}\mathbf{x}_i).$$

The gradients on the right-hand side are given by

$$\left(\frac{\partial L_{sparsity}}{\partial \hat{\mathbf{s}}_i}\right) = \text{diag}\left(\frac{1}{\hat{\mathbf{s}}_i}\right)\frac{\partial \hat{\mathbf{s}}_i}{\partial B_{dk}},$$
$$\frac{\partial}{\partial B_{dk}}\left(\frac{\partial L_{transfer}}{\partial \hat{\mathbf{s}}_i}\right) = \sum_{j=1}^{n} M_{ij}\frac{\partial \hat{\mathbf{s}}_j}{\partial B_{dk}}.$$

Here, $\text{diag}(1/\hat{\mathbf{s}}_i)$ refers to the matrix with diagonal $(1/\hat{s}_i^{(1)}, \ldots, 1/\hat{s}_i^{(k)})$ and zeros elsewhere. Substitution in (4.22) gives

$$\mathbf{J}^{kd}(\mathbf{B}\hat{\mathbf{s}}_i - \mathbf{P}\mathbf{x}_i) + \mathbf{B}^T(\mathbf{J}^{dk}\hat{\mathbf{s}}_i + \mathbf{B}\frac{\partial \hat{\mathbf{s}}_i}{\partial B_{dk}} - \frac{\partial \mathbf{P}}{\partial B_{dk}}\mathbf{x}_i) = \tag{4.23}$$
$$\text{diag}\left(-\frac{\lambda}{\hat{\mathbf{s}}_i}\right)\frac{\partial \hat{\mathbf{s}}_i}{\partial B_{dk}} - \mu\sum_{j=1}^{n} M_{ij}\frac{\partial \hat{\mathbf{s}}_j}{\partial B_{dk}}, i = 1, \ldots, n \tag{4.24}$$

Until this stage, the derivation is analogous to the derivation for the supervised learning procedure of Differentiable Sparse Coding (Section 2.4). There, the next step consisted of solving for a *single* $\partial \hat{\mathbf{s}}_i/\partial B_{dk}$. This cannot be done when including the transfer regularization $L_{transfer}$ which introduces a dependence between the optimal values of the coefficients. In the gradient, this dependency is represented by $\sum_{j=1}^{n} M_{ij}\partial \hat{\mathbf{s}}_j/\partial B_{dk}$. I address this difficulty by finding an expression for the partial derivatives with respect *all* coefficients $\mathbf{s}_1, \ldots, \mathbf{s}_n$. Implementation of this solution eliminates the need to loop over all samples by exploiting efficient matrix operations. The first step is collecting all partial derivatives in a single matrix:

$$\frac{\partial \hat{\mathbf{S}}}{\partial B_{dk}} = \left[\frac{\partial \hat{\mathbf{s}}_1}{\partial B_{dk}}, \ldots, \frac{\partial \hat{\mathbf{s}}_n}{\partial B_{dk}}\right]$$

denotes the $K \times n$ matrix with ith column equal to $\partial \hat{\mathbf{s}}_i/\partial B_{dk}$. Then the equations (4.23) can be combined in the following linear system:

$$\mathbf{J}^{kd}(\mathbf{B}\hat{\mathbf{S}} - \mathbf{P}\mathbf{X}) + \mathbf{B}^T(\mathbf{J}^{dk}\hat{\mathbf{S}} + \mathbf{B}\frac{\partial \hat{\mathbf{S}}}{\partial B_{dk}} - \frac{\partial \mathbf{P}}{\partial B_{dk}}\mathbf{X}) =$$
$$\text{diag}\left(\frac{-\lambda}{\hat{\mathbf{S}}}\right)\frac{\partial \hat{\mathbf{S}}}{\partial B_{dk}} - \mu\left(\mathbf{M} \otimes \mathbf{I}_K\right)\frac{\partial \hat{\mathbf{S}}}{\partial B_{dk}}.$$

The transfer penalties for all samples are collected in the $Kn \times Kn$ matrix $\mathbf{M} \otimes \mathbf{I}_K$. We can find all partial derivatives by solving the linear system:

$$-(\mathbf{A} + \mu\mathbf{M} \otimes \mathbf{I}_K)\frac{\partial \hat{\mathbf{S}}}{\partial B_{dk}} = \left(\mathbf{J}^{kd}(\mathbf{B}\hat{\mathbf{S}} - \mathbf{P}\mathbf{X}) + \mathbf{B}^T(\mathbf{J}^{dk}\hat{\mathbf{S}} - \frac{\partial \mathbf{P}}{\partial B_{dk}}\mathbf{X})\right), \tag{4.25}$$

finding the following expression for $\frac{\partial \hat{\mathbf{S}}}{\partial B_{dk}}$:

$$-(\mathbf{A} + \mu\mathbf{M} \otimes \mathbf{I}_K)^{-1}\left(\mathbf{J}^{kd}(\mathbf{B}^T\hat{\mathbf{S}} - \mathbf{P}\mathbf{X}) + \mathbf{B}^T(\mathbf{J}^{dk}\hat{\mathbf{S}} - \frac{\partial \mathbf{P}}{\partial B_{dk}}\mathbf{X})\right). \tag{4.26}$$

Here, $\mathbf{A}$ is the $Kn \times Kn$ matrix defined as

$$\mathbf{A} = \mathbf{I}_n \otimes (\mathbf{B}^T\mathbf{B}) + \operatorname{diag}\left(\frac{\lambda}{\hat{\mathbf{S}}}\right),$$

This matrix is block diagonal, $\mathbf{A} = \operatorname{blkdiag}(\mathbf{A}_1, \ldots, \mathbf{A}_n)$ with block $\mathbf{A}_i \in \mathbb{R}^{K\times K}$ corresponding to sample i:

$$\mathbf{A}_i = \mathbf{B}^T\mathbf{B} + \operatorname{diag}\left(\frac{\lambda}{\hat{\mathbf{s}}_i}\right).$$

An explicit expression for $\partial\hat{\mathbf{S}}/\partial B_{dk}$ eliminates the need to loop over all samples providing a speed-up over the sample-wise result from Bagnell et. al. (Section 3.3). However, the expression still depends on the basis entries (d, k). Optimization using this gradient would therefore require us to consider each entry individually. For large-scale problems, this becomes a computational bottleneck. In order to solve this challenge, I will now derive an expression for the gradient of the matrix $\mathbf{S}$ with respect to the matrix $\mathbf{B}$. This eliminates also the need to loop over the entries of $\mathbf{B}$, leading to a further speed-up compared to the multiple sample result. We can expand the second term in (4.26) into a second order tensor, where the added dimensions correspond to the dimensions D, K of the basis:

$$\mathbf{D} = (\mathbf{BS} - \mathbf{PX}) \otimes \operatorname{vec}(\mathbf{I}_K) + \mathbf{S} \otimes \mathbf{B} + \mathbf{B}^T\frac{\partial\mathbf{P}}{\partial\mathbf{B}}\mathbf{X}.$$

The partial derivative $\partial\mathbf{P}/\partial\mathbf{B}$ is unknown. It can be numerically estimated by

$$\frac{\partial\mathbf{P}}{\partial B_{dk}} \approx \frac{\mathbf{P}(\mathbf{B} + \varepsilon\mathbf{J}^{dk}) - \mathbf{P}(\mathbf{B} - \varepsilon\mathbf{J}^{dk})}{2\varepsilon}. \tag{4.27}$$

for small ε. The right-hand side can be computed by solving (4.20) for $\mathbf{P}$:

$$\mathbf{P} = \left(\mathbf{BS} + \mathbf{B}^{-T}(\lambda \log\frac{\mathbf{S}}{\mathbf{p}} + \mu\sum_{j=1}^{n} M_{ij}\mathbf{s}_j)\right)\mathbf{X}^{-1}. \tag{4.28}$$

This requires the matrices $\mathbf{B}$ and $\mathbf{X}$ to be invertible. In particular, the number of basis entries K and the number of mini-batch samples n_b must be chosen equal to the dimensionality of the data d. In practice, it is likely that both constraints can be satisfied without compromising the performance since $K \approx d$ is approximately optimal when using Sparse Coding for classification [45] and the online algorithms for sparse coding converge well for different mini-batch sizes, with 256 or 512 being optimal [31].

The approximation (4.28) is widely used to test the implementation of the faster analytical derivation [8]. However, it is very slow when the dimensionality of $\mathbf{B}$ is high, because it requires us to loop over all entries (d, k) individually.

We therefore approximate $\mathbf{D}$ by omitting the term that contains the partial derivative:

$$\tilde{\mathbf{D}} = (\mathbf{BS} - \mathbf{PX}) \otimes \text{vec}(\mathbf{I}_K) + \mathbf{S} \otimes \mathbf{B}.$$

Remark (Assessment of the approximation)**.** *I performed preliminary small experiments in order to compare the performance of* $\mathbf{D}$ *where* $\partial\mathbf{P}/\partial\mathbf{B}$ *is estimated using the numerical approximation* (4.27) *with the performance of* $\tilde{\mathbf{D}}$ *where* $\partial\mathbf{P}/\partial\mathbf{B}$ *is set to zero. The data for the experiments consisted of* $n = 240$ *4-dimensional noisy unit vectors* $\mathbf{e}_i + \eta, i = 1, 2, 3, 4$*. The identity of the 'latent' noise-free unit vector* $\mathbf{e}_i$ *determined the class label of each sample. First, the method was trained unsupervised, subsequently the parameters were fine-tuned using* $\mathbf{D}$*. The resulting classification performance was then compared to the classification performance after fine-tuning the model using the approximation* $\tilde{\mathbf{D}}$*. The same unsupervised parameters were used to initialize both supervised optimization procedures in order to facilitate comparison. The experiment was run 5 repeated times to reduce any randomness caused by random generation of the data and initialization of the parameters. The average test accuracy (% correct) of* $\mathbf{D}$ *and* $\tilde{\mathbf{D}}$ *are 95.34% and 94.54% respectively. In 3 out of 5 runs, using* $\mathbf{D}$ *outperformed* $\tilde{\mathbf{D}}$*. In the remaining 2 runs, both methods achieved the same performance. This indicates that using the approximation* $\tilde{\mathbf{D}}$ *leads to a relatively modest decline in performance compared to using* $\tilde{\mathbf{D}}$ *with the numerical estimation* (4.27)*. See Chapter 6 for a discussion of future work on estimation of* $\partial\mathbf{P}/\partial\mathbf{B}$*.*

Using $\tilde{\mathbf{D}}$, we obtain the following approximation of the gradient needed for backpropagation:

$$\frac{\partial\hat{\mathbf{S}}}{\partial\mathbf{B}} \approx -(\mathbf{A} + \mu\mathbf{M} \otimes \mathbf{I}_K)^{-1}\tilde{\mathbf{D}}. \tag{4.29}$$

The indices are defined such that

$$\left(\frac{\partial\hat{\mathbf{S}}}{\partial\mathbf{B}}\right)^{k,i}_{d,l} = \frac{\partial\hat{\mathbf{s}}^k_i}{\partial B_{d,l}}.$$

To see how the exact expression

$$\frac{\partial\hat{\mathbf{S}}}{\partial\mathbf{B}} = -(\mathbf{A} + \mu\mathbf{M} \otimes \mathbf{I}_K)^{-1}\left[(\mathbf{BS} - \mathbf{PX}) \otimes \text{vec}(\mathbf{I}_K) + \mathbf{S} \otimes \mathbf{B} + \mathbf{B}^T\frac{\partial\mathbf{P}}{\partial\mathbf{B}}\mathbf{X}\right]$$

generalizes DSC (Section 3.3), consider $\partial\mathbf{S}/\partial\mathbf{B}$ without $\mathbf{P}$ and the transfer penalty. The result then reduces to

$$\frac{\partial\hat{\mathbf{S}}}{\partial\mathbf{B}} = -\mathbf{A}^{-1}\left[(\mathbf{BS} - \mathbf{X}) \otimes \text{vec}(\mathbf{I}_K) + \mathbf{S} \otimes \mathbf{B}\right].$$

This is an explicit formula for the gradient (3.20) of *all* coefficients with respect to the *full* matrix $\mathbf{B}$. Bagnell et. al. derived the equivalent expression for a *single* sample and a *single* basis entry B_{dk}. Chapter 5 reports an experimental result that quantifies the speed-up that is achieved by switching from the single sample, single entry result to the explicit formula.

4.3.2 Gradients with respect to projection

The derivation of $\partial\hat{\mathbf{S}}/\partial\mathbf{P}$ is analogous to the one of $\partial\hat{\mathbf{S}}/\partial\mathbf{B}$. From the starting point (3.18), the first step is computing $\partial\hat{\mathbf{s}}_i/\partial P_{de}$ by implicit differentiation:

$$\frac{\partial}{\partial P_{de}}\left(\frac{\partial L_{data}}{\partial\hat{\mathbf{s}}_i}\right) = -\lambda\frac{\partial}{\partial P_{de}}\left(\frac{\partial L_{sparsity}}{\partial\hat{\mathbf{s}}_i}\right) - \mu\frac{\partial}{\partial P_{de}}\left(\frac{\partial L_{transfer}}{\partial\hat{\mathbf{s}}_i}\right),$$

The left-hand side of this equation is given by:

$$\frac{\partial}{\partial P_{de}}\left(\frac{\partial L_{data}}{\partial\hat{\mathbf{s}}_i}\right) = \frac{\partial\mathbf{B}^T}{\partial P_{de}}(\mathbf{B}\hat{\mathbf{s}}_i - \mathbf{P}\mathbf{x}_i) + \mathbf{B}^T(\frac{\partial\mathbf{B}}{\partial P_{de}}\hat{\mathbf{s}}_i + \mathbf{B}\frac{\partial\hat{\mathbf{s}}_i}{\partial P_{de}} - \mathbf{J}^{de}\mathbf{x}_i).$$

the gradient on the right-hand side equals

$$\frac{\partial}{\partial P_{de}}\left(\frac{\partial L_{sparsity}}{\partial\hat{\mathbf{s}}_i}\right) = \operatorname{diag}\left(\frac{1}{\hat{\mathbf{s}}_i}\right)\frac{\partial\hat{\mathbf{s}}_i}{\partial P_{de}},$$
$$\frac{\partial}{\partial P_{de}}\left(\frac{\partial L_{transfer}}{\partial\hat{\mathbf{s}}_i}\right) = \sum_{j=1}^{n} M_{ij}\frac{\partial\hat{\mathbf{s}}_j}{\partial P_{de}}.$$

This translates to

$$\frac{\partial\mathbf{B}^T}{\partial P_{de}}(\mathbf{B}\hat{\mathbf{s}}_i - \mathbf{P}\mathbf{x}_i) + \mathbf{B}^T(\frac{\partial\mathbf{B}}{\partial P_{de}}\hat{\mathbf{s}}_i + \mathbf{B}\frac{\partial\hat{\mathbf{s}}_i}{\partial P_{de}} - \mathbf{J}^{de}\mathbf{x}_i) =$$
$$\operatorname{diag}\left(\frac{-\lambda}{\hat{\mathbf{s}}_i}\right)\frac{\partial\hat{\mathbf{s}}_i}{\partial P_{de}} - \mu\sum_{j=1}^{n} M_{ij}\frac{\partial\hat{\mathbf{s}}_j}{\partial P_{de}}.$$

Next, we need to solve for all the gradient of all samples:

$$\frac{\partial\hat{\mathbf{S}}}{\partial P_{de}} = (\mathbf{A} + \mu\mathbf{M}\otimes\mathbf{I}_K)^{-1}\left[\frac{\partial\mathbf{B}^T}{P_{de}}(\mathbf{B}\mathbf{S} - \mathbf{P}\mathbf{X}) + \mathbf{B}^T(\frac{\partial\mathbf{B}}{\partial\mathbf{P}_{de}}\mathbf{S} - \mathbf{J}^{de}\mathbf{X})\right].$$

Finally, we compute the gradient with respect to the matrix $\mathbf{P}$ by expanding the second term into a tensor, where the added dimensions correspond to the dimensions of $\mathbf{P}$. Let

$$\mathbf{E} = \left(\frac{\partial\mathbf{B}}{\partial\mathbf{P}}\right)^T(\mathbf{B}\mathbf{S} - \mathbf{P}\mathbf{X}) + \mathbf{B}^T\left(\frac{\partial\mathbf{B}}{\partial\mathbf{P}} - \mathbf{X}\otimes\mathbf{B}\right).$$

This expression includes the unknown partial derivative $\partial\mathbf{B}/\partial\mathbf{P}$. This is not as straightforward as solving for $\partial\mathbf{P}/\partial\mathbf{B}$, because (4.20) is quadratic in $\mathbf{B}$. There are different potential alternative approaches to solve such a matrix equation with a quadratic term. Another possibility is to approximate $\mathbf{E}$ by simply omitting the terms that involve the unknown $\partial\mathbf{B}/\partial\mathbf{P}$. This results in

$$\tilde{\mathbf{E}} = -\mathbf{B}^T\mathbf{X}\otimes\mathbf{B}.$$

Since the small scale experiments described in the previous section and the digit classification from Chapter 5 showed satisfying results using $\tilde{\mathbf{E}}$, I have not

pursued other solutions yet. See Chapter 6 for a discussion on future work. The final result reads

$$\frac{\partial \hat{\mathbf{S}}}{\partial \mathbf{P}} \approx (\mathbf{A} + \mu \mathbf{M} \otimes \mathbf{I}_K)^{-1} \tilde{\mathbf{E}}. \tag{4.30}$$

This is a $d \times e \times d \times K$ tensor, where the first indices correspond to the dimensions of $\mathbf{P}$ and the third and fourth index correspond to the dimensions of $\mathbf{B}$.

Input: Data matrix $\mathbf{X} \in \mathbb{R}^{D \times n}$, $k \in \mathbb{N}$ (number of basis vectors), λ_s (sparsity regularization parameter), $p \in \mathbb{R}$ (prior), λ_o (orthogonality), λ_r (signal fidelity), λ_b (column normalization), μ, μ_P (transfer regularization parameters), number of iterations $iter$, number of iterations for EGD $egdIter$, minibatch size n_b.

Output: Optimal coefficient matrix $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$, basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_K]$ and projection $\mathbf{P}$.

begin Initialization

  Randomly initialize $\mathbf{S}, \mathbf{B}$, and $\mathbf{P}$. Make sure basis columns are zero mean, unit variance.

end

begin optimization

  for $i \in [1, \ldots, iter]$ **do**

    Sample a mini-batch $\mathbf{X}_b = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_b}]$ from X. Let $\mathbf{S}_b$ denote the corresponding coefficients.

    for $b \in [1, \ldots, B]$ **do**

      Repeatedly update mini-batch coefficients using EGD:

      for $i_s \in [1, \ldots, egdIter]$ **do**

        Compute $\partial L / \partial \mathbf{S}_b$ using (4.4). Determine learning rate α using Algorithm 3, update:

$$\mathbf{S}_b := \mathbf{S}_b \circ \exp\left(-\alpha \frac{\partial L}{\partial \mathbf{S}_b}\right)$$

      end

      so that (approximately) $\mathbf{S}_b = \operatorname{argmin}_{\mathbf{s}_b} L(\mathbf{X}_b, \mathbf{P}, \mathbf{B}, \mathbf{S}_b)$.

      Compute $\partial L / \partial \mathbf{B}$ using (4.6), compute $\partial L / \partial \mathbf{P}$ using (4.8). Use L-BFGS [60] to compute the optimal $\mathbf{B}$ and $\mathbf{P}$, based on $\mathbf{X}_b$ and the current estimates of $\mathbf{S}_b, \mathbf{B}, \mathbf{P}$.

    end

  end

end

Algorithm 5: Unsupervised learning of the basis $\mathbf{B}$ and transformation $\mathbf{P}$, analogous to DSC Algorithm 4.

Input: Data matrix $\mathbf{X} \in \mathbb{R}^{D \times n}$, class labels $\mathbf{y} \in \mathbb{R}^n$, initial (unsupervised) basis $\mathbf{B}$ and projection matrix $\mathbf{P}$, classifier parameters Θ, mini-batch size n_b, learning rate parameters α_0, k.
Output: Supervised basis $\mathbf{B}_{sup}$ and projection $\mathbf{P}_{sup}$.
begin initialization
| $\mathbf{B}_{sup} := \mathbf{B}, \mathbf{P}_{sup} := \mathbf{P}$.
end
begin optimization
| **for** $i \in [1, \ldots, iter]$ **do**
| | Sample a mini-batch $\{(\mathbf{x}_1, y_1, \ldots, \mathbf{x}_{n_b}, y_b)$ from $\mathcal{D}_S$. Determine the optimal coefficients $\hat{\mathbf{S}}_b$ of batch b using Algorithm 5, based on data $\mathbf{X}_b$ and matrix $\mathbf{B}_{sup}$. Compute the supervised error signal of the classifier:
| | $$\frac{\partial L_{sup}}{\partial \hat{\mathbf{S}}_b} = \text{classify}(\mathbf{X}_b, \mathbf{y}_b, \mathbf{S}_b, \Theta)$$
| | Determine $\partial \hat{\mathbf{S}} / \partial \mathbf{B}_{sup}$ using (4.29) and $\partial \hat{\mathbf{S}} / \partial \mathbf{P}_{sup}$ using (4.30). Compute supervised gradients with respect to $\mathbf{B}$ and $\mathbf{P}$:
| | $$\frac{\partial L_{sup}}{\partial \mathbf{B}_{sup}} := \frac{\partial L_{sup}}{\partial \hat{\mathbf{S}}_b} \cdot \frac{\partial \hat{\mathbf{S}}_b}{\partial \mathbf{B}_{sup}},$$
| | $$\frac{\partial L_{sup}}{\partial \mathbf{P}_{sup}} := \frac{\partial L_{sup}}{\partial \hat{\mathbf{S}}_b} \cdot \frac{\partial \hat{\mathbf{S}}_b}{\partial \mathbf{P}_{sup}}$$
| | Set learning rate
| | $$\eta := \frac{\alpha_0}{k \cdot i + 1}$$
| | and update $\mathbf{B}$ and $\mathbf{P}$ using SGD:
| | $$\mathbf{B}_{sup} := \mathbf{B}_{sup} - \eta \frac{\partial L_{sup}}{\partial \mathbf{B}_{sup}},$$
| | $$\mathbf{P}_{sup} := \mathbf{P}_{sup} - \rho \frac{\partial L_{sup}}{\partial \mathbf{P}_{sup}}.$$
| **end**
end

Algorithm 6: Supervised learning of the basis and projection. The procedure uses only source samples since no labeled target data is available. The mini-batch size and learning rate parameters were deterimined based on the classification accuracy of the validation set.

Chapter 5

Experimental Results

This chapter evaluates the ability of the proposed model to learn from different domains. The transfer learning problem is that of classifying hand written digits from two different benchmark datasets widely used to evaluate computer vision and machine learning algorithms. The source data is given by images from the USPS dataset [1], the target data is given by images from the MNIST dataset [38]. The USPS dataset consists of 7,291 training images and 2,007 test images of size 16×16. The MNIST dataset consists of 60,000 training images and 10,000 test images of size 28×28. USPS and MNIST share 10 classes, each corresponding to a digit $0, 1, \ldots, 9$. As can be seen in Figure 5.1, the datasets are related but still clearly distinguishable from each other. I will compare the performance of the proposed model with that of the Sparse Coding methods discussed in Chapter 3. I will start by evaluating the computational efficiency of the supervised derivation in Section 5.1. Then I will discuss the results on a non-transfer problem in Section 5.2 before presenting the results on the transfer task in Section 5.3.

5.1 Computational efficiency

Section 4.5 derived the gradient of the coefficients $\mathbf{S}$ with respect to the basis $\mathbf{B}$ and projection $\mathbf{P}$ for the proposed model. Without the transfer penalty and the projection, the model reduces to DSC. In the paper that introduces DSC [4], the authors derived the gradient of a single sample $\mathbf{s}$ with respect to a single basis entry B_{dk}. Tables 5.1 and 5.2 show that the tensor derivation significantly outperforms their result. Note that for a mini-batch size of 32 and 256 dimensional data, even the tensor version takes a considerable amount of time. This is because the number of basis vectors is chosen to be equal to the dimensionality of the data. In the experiments (Sections 5.2 and 5.3), the mini-batch size was therefore kept at a maximum of 8 samples. A larger number of samples could be used by first reducing the dimension of the data using PCA or letting $\mathbf{P}$ project the data to a subspace. Another approach would be to

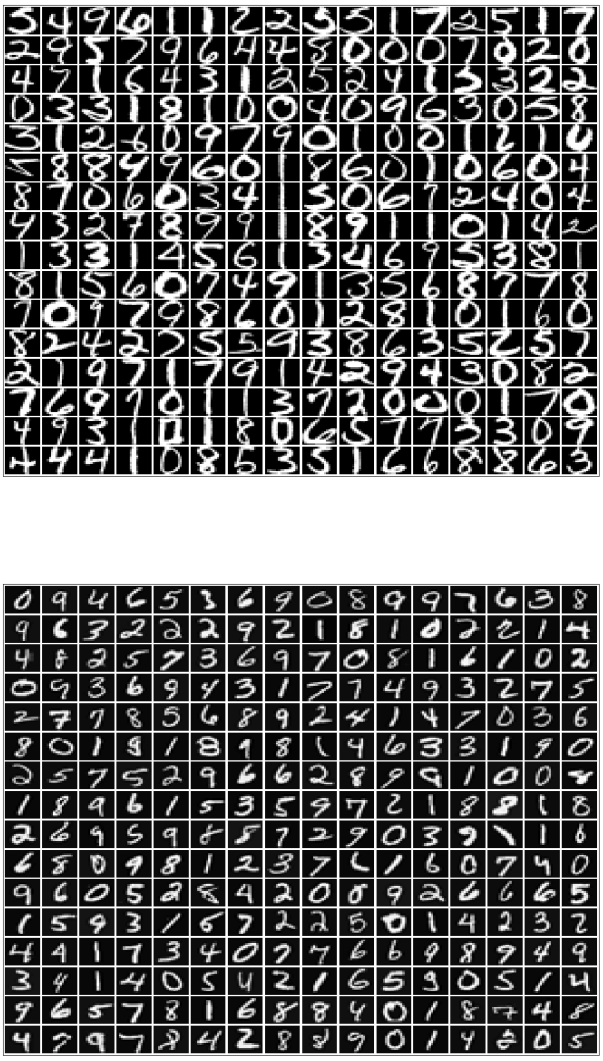

Figure 5.1: Random images of hand written digits from the USPS (upper panel) and MNIST dataset (lower panel) after preprocessing. The MNIST samples often occupy a smaller part of the image and tend to consist of thinner strokes.

Dimensionality	Loop	**Tensor**	Speedup factor
8	0.4626	0.0514	9
16	0.2333	0.0061	38
32	1.4227	0.0242	58
64	15.1697	0.2112	72
128	251.7168	2.4012	105
256	5072.0000	52.9000	96

Table 5.1: The tensor implementation derived in Chapter 4 is up to 100 times faster than the loop version. Numbers are computation time in seconds for a single backward pass using a mini-batch of size 8. In each experiment, the number of basis vectors was equal to the dimensionality of the data.

Dimensionality	Loop	**Tensor**	Speedup factor
8	0.1953	0.0124	16
16	0.9037	0.0212	43
32	5.6018	0.1631	34
64	60.0333	1.1711	51
128	997.700	17.8603	56
256	20198.00	250.0000	81

Table 5.2: Computation time in seconds for a single backward pass using a mini-batch of size 32.

use image patches instead of entire images, a popular approach when applying Sparse Coding to image classification (see e.g. [12], [62]).

5.2 Baseline experiment

I will start by comparing the result of training Sparse Coding (SC) and Differentiable Sparse Coding (DSC) on the MNIST data set. In order to speed up experiments and in keeping with previous work on transfer Sparse Coding [42], I construct two datasets by randomly selecting a subset of the images For both methods, the basis was trained using the same 4000 random images from the MNIST training set. I first reduced all images from 28×28 to 16×16 dimensions and represent each image as a 256 dimensional vector that encodes the gray-scale values of the image, scaled to the interval $[0, 1]$. The coefficients of 2,000 samples were used to train a classifier (L_2-regularized multinomial logistic regression). The remaining 2,000 samples were used as validation set to pick the optimal tuning parameters. For both SC and DSC, I set the sparsity regularization by searching $\lambda \in \{0.05, 0.075, 0.1, 0.0125, 0.15\}$. Additionally, I searched $p \in \{0.005, 0.01, 0.1\}$ for the prior parameter of DSC. After training the basis, this was used to code 2,000 images randomly sampled from the MNIST test set.

	Train	Validate	Test	Test [4]
SC [39]	97.00 %	94.05 %	93.40%	93.63%
DSC [4]	99.45%	94.60%	94.00%	94.94%

Table 5.3: Results (% correct) on the MNIST to MNIST experiment. The right most column shows the results reported by Bagnell et. al. [4] obtained from training the bases on a larger number of samples. See text for more details.

Subsequently, the classifier was used to predict the corresponding class labels.

As shown in Table 5.3, the features learned using Sparse Coding and Differentiable Sparse Coding result in a comparable accuracy, with the performance of DSC being slightly higher. This result is consistent with the performance reported in Bagnell et. al. [4]. Their results are slightly better, which may be attributed to the fact that they used a larger number of samples. Bagnell et. al. trained the bases on 50,000 instead of 4,000 images (a 12.5 fold differences). Subsequently they trained a classifier on 2,000 random samples from the training set (just like I did). They selected the tuning parameters that maximized the classification accuracy on a validation set of 10,000 samples (5 fold difference) and report the accuracy on 10,000 (a 5 fold difference) test images as the test accuracy.

While the classification performance of SC and DSC is comparable, the learned bases are qualitatively very different (Figure 5.2). In contrast to KL-regularized Sparse Coding, L_1-regularization produces both positive and negative weights. Therefore, the SC basis contains both images of a dark digit (positive pixel values) on a light background (negative values) and vice versa. The DSC basis consists of light digits on a dark background only. Moreover, compared to the differentiable penalty, L_1-regularization leads to sharper edges and homogeneous pixel intensities. This effect is less pronounced in the two reconstructions of the images (Figure 5.3). Recall from Chapter 3 that the methods learns a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_K]$ and coefficients matrix $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_K]$ that can be used to reconstruct the images $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, using

$$\mathbf{x}_i \approx \sum_k \mathbf{b}_k s_i^{(k)}.$$

Thus Figure 5.3 shows $\mathbf{B}\mathbf{s}_i$ for several images i.

In conclusion, SC and DSC achieve comparable performance on the MNIST data set by learning disparate bases and coefficients. The results from this section can serve as a reference when interpreting the results from the transfer experiment.

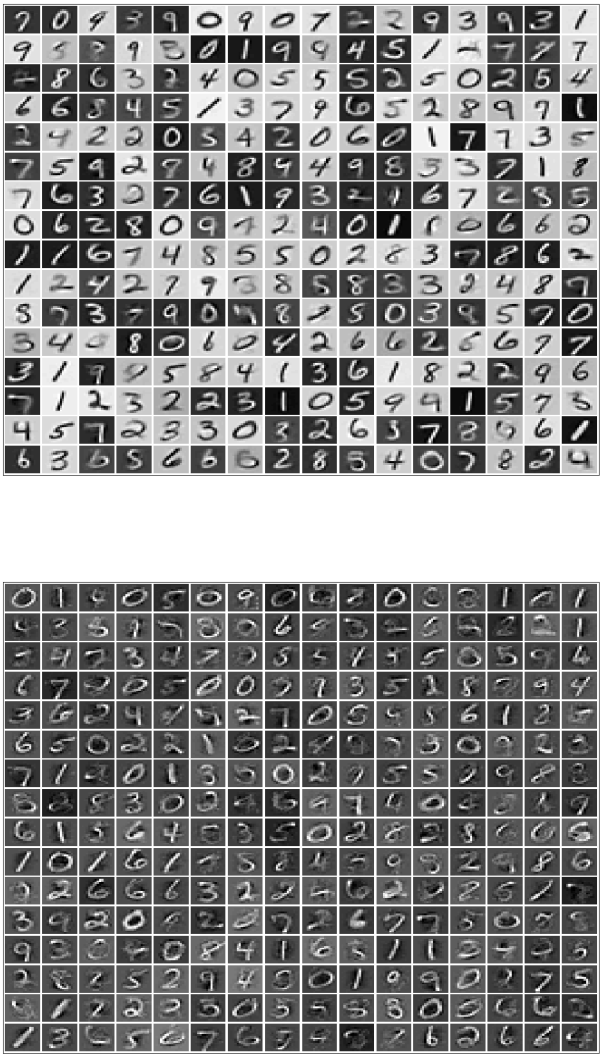

Figure 5.2: Bases learned using SC (upper panel) and DSC (lower panel) are clearly distinct.

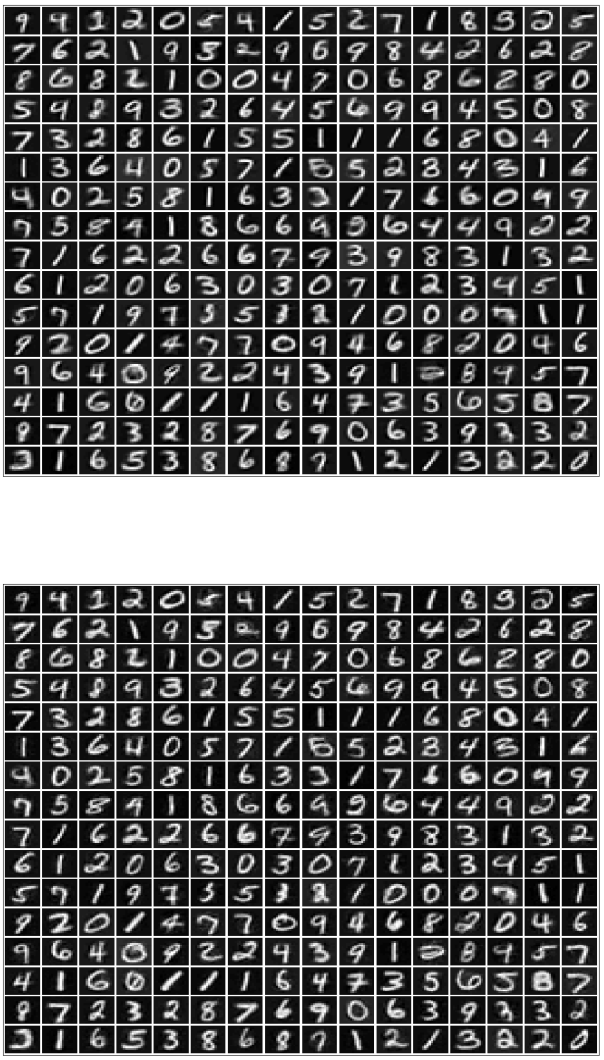

Figure 5.3: Reconstruction of MNIST images using the bases learned by SC (upper panel) an DSC (lower panel). The lower image of Figure 5.1 shows the ground truth.

5.3 Transfer experiment

Next, I will discuss the main experiment in which I compare the Sparse Coding models SC, Transfer Sparse Coding Differentiable Sparse Coding (Chapter 3) and the proposed method (Chapter 4). I did not perform experiments with the SDDL model (Section 3.4), since it requires labeled data from both the source and the target domain. The authors from a different paper [41] trained SDDL on USPS vs MNIST using some labeled MNIST data to obtain a classification accuracy of 35.21%. This is much lower than the performances reported here, but differences in experimental setup prevent direct comparison of the results. In order to speed up experiments and to keep with previous work on transfer Sparse Coding [42], I construct two datasets by randomly selecting a subset of the images from both datasets. The training set consists of 2,000 random samples from the USPS dataset and serves as the source domain for which class labels are available. The validation and test set each contain 2,000 samples from the MNIST dataset. It can be argued that using labeled target samples as a validation set violates the domain adaptation paradigm. However, I have chosen to keep with recent practice (see e.g. [25], [9]) by using labeled target samples as a validation set. This choice can be justified by the fact that in applications, often at least a small number of labeled target samples is available.

For SC, TSC and DSC, I search the same set of values for the tuning parameters. For TSC and the proposed model, I search for the optimal transfer parameter μ in the set $\{0, 10^3, 10^4, 10^5\}$. The proposed model has quite some tuning parameters, but I found that determining them using a combination of simple heurstics and optimization on the validation set resulted in reasonable performance. The first heuristic choice I made is to let the transformation preserve the dimensionality of the data. In particular, it does not perform dimension reduction. Furthermore, I fix the regularization and prior parameter to $\lambda = 0.05$ and $p = 0.005$, respectively. These were the optimal parameters found for the MNIST to MNIST experiment (Section 5.2), and they turn out to work well for the proposed model too. For each additional parameter I consider 0 (i.e. not including the penalty term) and a non-negative value that is inversely proportional to the magnitude of the corresponding term in the loss function. For the transfer penalty of the transformed data, I try $\mu_P \in \{0, 10^4\}$, and graph Laplacian regularization $\gamma_P \in \{0, 1\}$. For the signal fidelity parameter the values $\lambda_{retain} \in \{0, 10^3\}$ are considered The obtained combination of tuning parameters, only the latter one equals $\lambda_{retain} = 0$, the other parameters are positive. The orthogonality regularization was adjusted such that the projection was close to orthogonal. Specifically, the projection was initialized as the identity matrix, the orthogonality parameter λ_o was initialized to 1, and doubled when during optimization the mean difference in column norm exceeded 10^{-3}. The final value of λ_o depended on the values of the other tuning parameters.

The classification accuracy of the tested models is shown in Table 5.4. Sev-

	Train	Validate	Test
SC [39]	93.35%	52.60%	49.30 %
Transfer SC [42]	93.35%	51.25%	49.75%
DSC [4]	99.95%	52.00%	47.40%
Transfer DSC	98.90%	51.70%	50.35%
Proposed method	99.10 %	54.20%	**53.35**%

Table 5.4: Results on the USPS to MNIST transfer experiment. The proposed method improves upon previous approaches.

eral conclusions can be drawn from these results.

- The proposed model outperforms all other models. In particular, it improves upon transfer regularized DSC, showing the added benefit of incorporating a projection.
- Each of the tested models including those designed for transfer learning fail to achieve much more than 50% accuracy. This is a dramatic degradation compared to the one obtained in the baseline experiment (Table 5.2). Of course, the transfer experiment is much more challenging than the baseline experiment and the degradation in performance is expected.
- The performance of the proposed method on the test set is only slightly worse than that on the validation set. Together with the fact that similar performance (51%-53% range) was achieved for different values of the tuning parameters, this suggests that a smaller validation set could be used. Therefore, one might get away with using a limited number of labeled target samples to tune the hyper parameters.
- SC and DSC with transfer regularization outperform their counterparts without a transfer regularization that seem more prone to over fitting. The advantage of incorporating transfer regularization with the SC method is rather modest, raising the interesting question of how the effectiveness of this penalty depends on the sparsity of the data.

Recall from Chapter 4 that the proposed method first projects the data to a shared subspace before learning the basis and the sparse codes (Figure 4.1). Figure 5.4 shows the result of applying the learned projections P_S, P_T to the data. It seems that the model tries to decrease the domain differences by combining large digits with smaller digits in a single basis vector. This might be the result from the fact that the USPS digits often occupy a larger part of the image (Figure 5.1).

The model does not completely remove the domain differences. This is clear from the performance after training on USPS (Table 5.4) compared to that after training on MNIST (Table 5.3). Investigating the learned representations gives

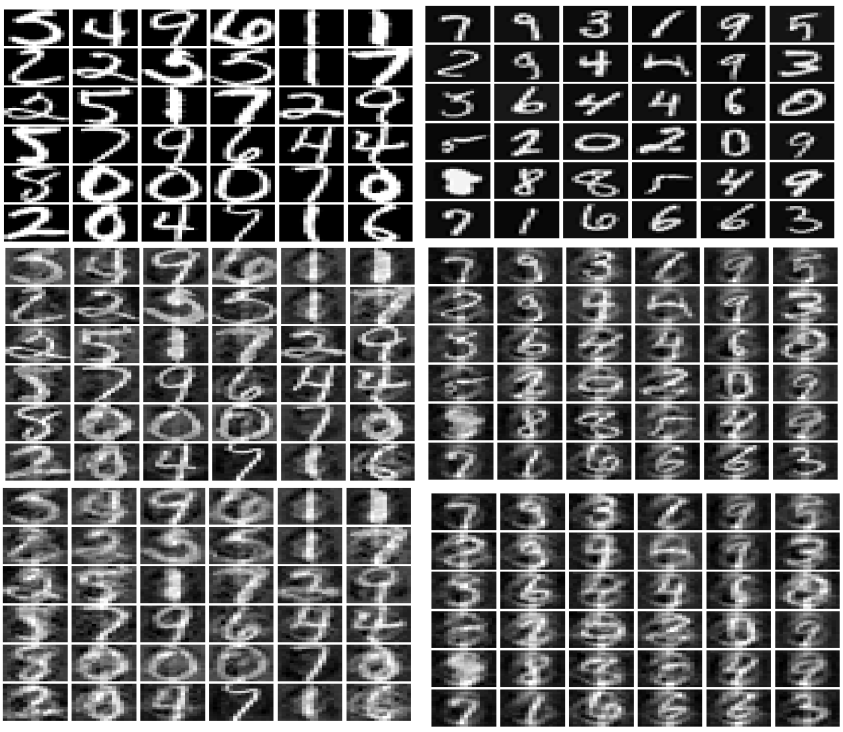

Figure 5.4: Stepwise reduction of domain differences. Upper to lower panel: original images **X**, transformed data **PX**, reconstruction **BS**. USPS is shown left, MNIST is shown right. The contours surrounding the MNIST digits on the middle-left panel are possibly used by the model to reconstruct the larger USPS digits.

a sense why this is the case. Figure 5.5 shows the basis vectors sorted by the mean magnitude of the coefficients of the MNIST (upper) and USPS (lower) samples. This figure shows that source and target samples are represented by using the basis vectors in distinct ways. This is more explicitly illustrated by Figure 5.6 which shows the graph Laplacian for DSC and the proposed model. On both the x and the y axis, the first 2,000 indices correspond to the USPS samples, the second 2,000 indices correspond to the MNIST samples. At each coordinate (i, j), a dot is shown when the sparse coefficients of sample i and sample j are nearest neighbors. For this experiment, I chose to include the 5 nearest neighbors. Similar conclusions hold when choosing a different number (data not shown). The samples are sorted by class identity; the blocks on the diagonal indicate that coefficients are most similar within a certain class. This is a desirable property when the coefficients are used for image classification. In addition, we would like the coefficients to be similar between the domains. Figure 5.6 shows that is true in a rather limited way, since most similar coefficients are found within the same domain. This means that the learned representation is not entirely invariant with respect to domain differences, explaining the difficulties of all models, including the proposed one, to generalize from source to target domain (Table 5.4). Therefore, there remains significant of room for further improvement as will be discussed in the final chapter.

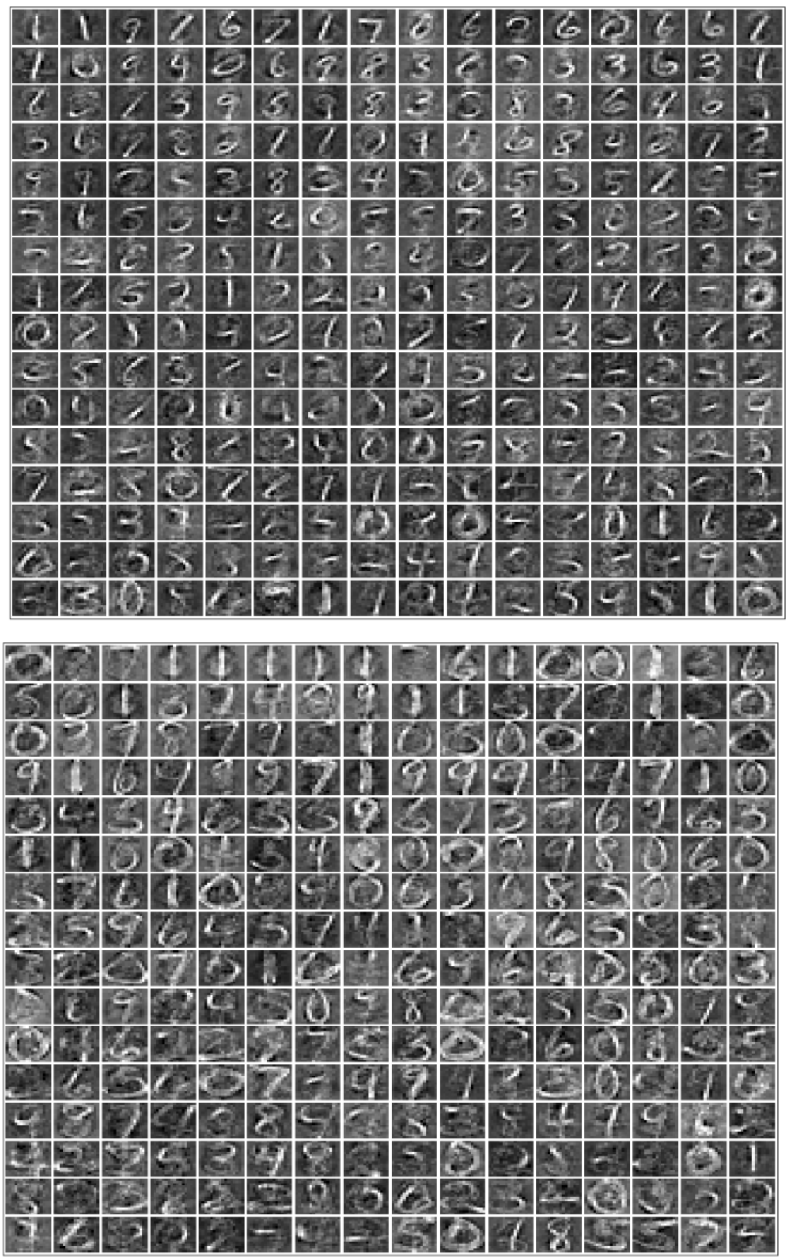

Figure 5.5: The two domains still represent the basis in distinct ways. Basis vectors learned using the proposed method, sorted by magnitude of their coefficients for the MNIST (upper) and USPS samples (lower)

.

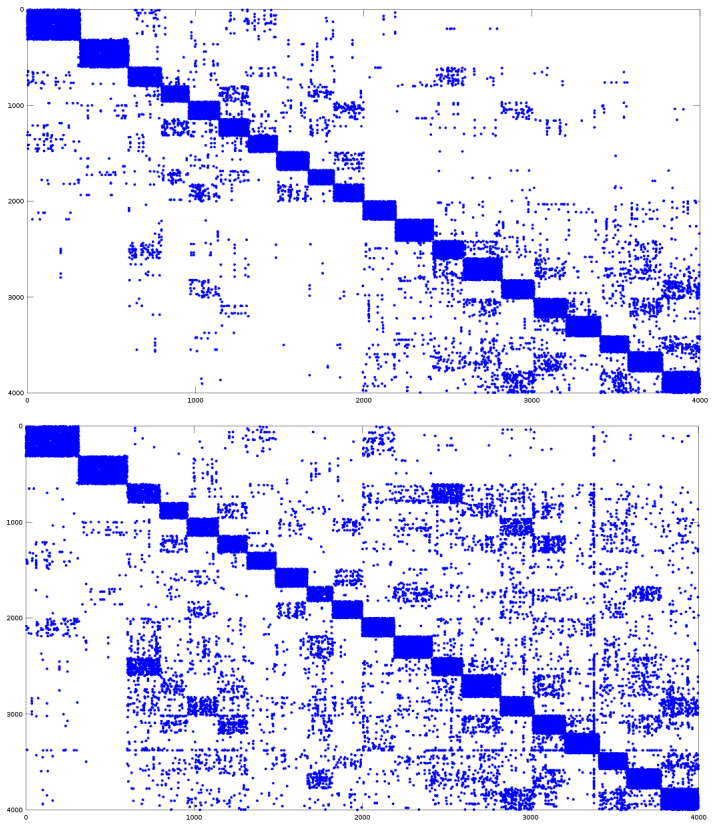

Figure 5.6: The proposed model reduces, but does not eliminate representational differences. At each position (i, j) the presence of a dot indicates that the coefficients of sample i and sample j are nearest neighbors. The first $2,000$ samples correspond to the source domain, the second $2,000$ samples correspond to the target domain. Since samples are sorted by class, the blocks on the diagonal reveal that most similarity is found between samples of the same class.

Chapter 6

Conclusion

This thesis introduces a Sparse Coding model capable of learning features that are both robust against domain differences and useful for classification. Robustness is achieved by linear transformation and a differentiable Sparse Coding step that both explicitly reduce domain differences between datasets. Usefulness of the features is increased by end-to-end training of the basis and projection using backpropagation of the classification loss. This can be done efficiently by exploiting tensor operations. I have tested the algorithm on a domain adaptation problem and shown that it achieves better performance compared to previous methods. Still, there is a large difference between the performance from the proposed model and the performance in the non-transfer experiment (Section 5.2). The source and target datasets (Figure 5.1) are very similar to the human observer, suggesting that it should be possible to devise a method that achieves comparable performance on both experiments. This chapter concludes the thesis by considering future research that might bridge this gap.

6.1 Discussion

Traditional theory and algorithms for learning from data assume that training and testing data are sampled from the same probability distribution defined on a single feature space. Yet this assumption is violated in many applications, leading to unsatisfying performance of the conventional methods. Transfer learning is concerned with the development of algorithms that are therefore robust to such domain differences. As discussed in Chapter 2, there are multiple scenarios when transfer learning can be applied. For each of these scenarios, different approaches are available for how to transfer between datasets. This thesis is concerned with the scenario of domain adaptation, which addresses the problem where labeled data from the source domain is sampled from a different distribution than the unlabeled data of the target domain. The model developed in this thesis applies the approach of feature-based transformation. This is done by building on previous work for Sparse Coding algorithms, discussed

Property	SC [39]	TSC [42]	DSC [4]	SDDL[61]	**Proposed**
Invariance:					
minimizing differences	N	Y	N	N	**Y**
modeling differences	N	N	N	Y	**Y**
Selectivity:					
stable encoding	N	N	Y	N	**Y**
supervision	N	N	Y	Y	**Y**

Table 6.1: Comparison of the Sparse Coding methods from the literature and the method developed in this thesis based on their ability to learn features that are invariant with respect to domain differences and selective for the classification task. The methods TSC, DSC and SDDL each offer an improvement over traditional SC, but simultaneously suffer from specific drawbacks that are resolved by the proposed method.

in Chapter 3. Table 6.1 summarizes the similarities and differences between the current method and those that have previously been proposed.

6.2 Future work

A drastic simplification used by the current approach is approximation of the partial gradients

$$\frac{\partial \hat{\mathbf{S}}}{\partial \mathbf{B}} \text{ and } \frac{\partial \hat{\mathbf{S}}}{\partial \mathbf{P}}.$$

These approximations are not needed in the DSC model and the DSC model with a transfer penalty since they do not incorporate $\mathbf{P}$. Therefore, the comparison (Section 5.1) of the efficiency of the derivation from Chapter 4 and the derivation from Bagnell et. al. is fair. Small scale experiments described in Chapter 4 show that leaving out the unknown partial derivatives leads to a minor decrease in performance compared to using a numerical approximation. However, currently no guarantees on the performance using the approximation are known. It is therefore not clear how much the results from Chapter 5 could be improved by using the true values instead of the approximations. Therefore, more work on the efficient computation of the partial gradients is needed in order to improve the result from the approximate supervised method.

There are also several approaches for improving the results using the model as-is. Training on more samples (either from the source or target domain) might be a way to exploit the additional model complexity introduced by the transformation. In a similar vein, it is straightforward to extend the current supervised fine-tuning step to to use a limited amount of labeled target data. It would be interesting to investigate the extent to which the model is able to benefit from labeled source samples by this semi-supervised training. Finally, better results might be obtained after a more elaborate hyper-parameter search than

the heuristically motivated search performed in Chapter 5. In particular, it would be interesting to see the effect of relaxing the constraints imposed on the transformation. For example, optimization of the method that inspired inclusion of a linear transformation (Section 3.4) relies on orthogonality of the transformation but this is not a requirement of the proposed method.

There are at least two ways in which future research could extend the model. The first one is by making it hierarchical. Current advances in learning from perceptual data are largely driven by so-called deep learning algorithms build from multiple layers that extract increasingly abstract features [37]. Due to its differentiability, the proposed model could also be extended to include multiple layers. Analogous to the procedure described in Chapter 4, the basis $\mathbf{B}^l$ of the l-th layer would be updated using

$$\frac{\partial L_{sup}}{\partial \mathbf{B}^l} = \frac{\partial L_{sup}}{\partial \mathbf{z}^l} \cdot \frac{\partial \mathbf{z}^l}{\partial \mathbf{B}^l}, \;\; l = 0, \ldots, L$$

where $\mathbf{z}^l$ is the output of the l-th layer. This generalizes the current model consisting of $L = 1$ layer which receives input $\mathbf{z}^0 = \mathbf{x}$ and computes $\mathbf{z}^1 = \mathbf{s}$. Supervised learning of the projection could be done using an analogous update rule. Hierarchy would introduce the possibility of reducing domain differences in a sequence of steps, each combining a transformation with Sparse Coding. This is an especially promising line of future research since a multi-layered Sparse Coding model recently achieved state-of-the-art performance in image classification [62].

A second way that is likely to increase the model's performance is by improving the transfer penalty. The current model penalizes the mean difference between the sparse coefficients of the domains. A more powerful solution uses the Maximum Mean Discrepancy (MMD) [27] which measures the difference in mean kernel embeddings rather than the sample means of the sparse coefficients. This difference matters since asymptotically, the MMD equals zero if and only if the distributions agree, whereas the version used by the current model vanishes as soon as the distributions share their first moment.

More elaborate experiments, a hierarchical extension or a better transfer penalty might thus further improve the current results.

Bibliography

[1] Usps digits database. `http://statweb.stanford.edu/~tibs/ElemStatLearn/data.html`. Accessed: 2016-1-28.

[2] Michal Aharon, Michael Elad, and Alfred Bruckstein. *rmk*-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[3] Maruan Al-Shedivat, Jim Jing-Yan Wang, Majed Alzahrani, Jianhua Z Huang, and Xin Gao. Supervised transfer sparse coding. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. The AAAI Press, 2014.

[4] JA Bagnell and David M Bradley. Differentiable sparse coding. In *Advances in Neural Information Processing Systems*, pages 113–120, 2009.

[5] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[6] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

[7] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[8] Christopher M. Bishop. *'Pattern recognition*. Springer Science + Business Media, 2006.

[9] Konstantinos et. al. Bousmalis. Domain separation networks. *arXiv preprint arXiv:1608:06019v1*, 2016.

[10] Stephen Poythress Boyd and Lieven Vandenberghe. Convex optimization. 2004.

[11] S.S. Chen, Donoho D.L., and Saunders M.A. Atomic decomposition by basis persuit. *SIAM J. Sc. Comp.*, 1998.

[12] Adam Coates and Andrew Y Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 921–928, 2011.

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

[14] Jesse Davis and Pedro Domingos. Deep transfer via second-order markov logic. In *Proceedings of the 26th annual international conference on machine learning*, pages 217–224. ACM, 2009.

[15] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[16] David L Donoho. For most large underdetermined systems of linear equations the minimal 1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics*, 59(6):797–829, 2006.

[17] Lixin Duan, Ivor W Tsang, and Dong Xu. Domain transfer multiple kernel learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):465–479, 2012.

[18] Lixin Duan, Ivor W Tsang, Dong Xu, and Stephen J Maybank. Domain transfer svm for video concept detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1375–1381. IEEE, 2009.

[19] Julio Martin Duarte-Carvajalino and Guillermo Sapiro. Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. *IEEE Transactions on Image Processing*, 18(7):1395–1408, 2009.

[20] B. Efron, T. Hastie, I. Johnstone, and Tibshirani R. Least angle regression. *Ann. Sat.*, 2004.

[21] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.

[22] Peter Foldiak. Sparse coding in the primate cortex. *The handbook of brain theory and neural networks*, 2003.

[23] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[24] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[25] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[26] Ioannis A Gkioulekas and Todd Zickler. Dimensionality reduction using the sparse linear model. In *Advances in Neural Information Processing Systems*, pages 271–279, 2011.

[27] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[28] Roger Grosse, Rajat Raina, Helen Kwong, and Andrew Y Ng. Shift-invariance sparse coding for audio classification. *arXiv preprint arXiv:1206.5241*, 2012.

[29] Yahong Han, Fei Wu, Dacheng Tao, Jian Shao, Yueting Zhuang, and Jianmin Jiang. Sparse unsupervised dimensionality reduction for multiple view data. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(10):1485–1496, 2012.

[30] Irina Higgins, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner. Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579*, 2016.

[31] Mairal J., Bach F., Ponce J., and Sapiro G. Online dictionary learning for sparse coding. pages 686–696.

[32] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *ACL*, volume 7, pages 264–271, 2007.

[33] Koray Kavukcuoglu, Rob Fergus, Yann LeCun, et al. Learning invariant features through topographic filter maps. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1605–1612. IEEE, 2009.

[34] Michael J Kearns and Umesh Virkumar Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[35] Jyrki Kivinen and Manfred K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

[36] Guy Le Besnerais, J-F Bercher, and Guy Demoment. A new look at entropy for solving linear inverse problems. *IEEE Transactions on Information Theory*, 45(5):1565–1578, 1999.

[37] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[38] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[39] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2006.

[40] Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 410–419. Association for Computational Linguistics, 2012.

[41] Xiao Li, Min Fang, Wang Hongchung, and Zhang Ju-Jie. Supervised transfer kernel sparse coding. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. The AAAI Press, 2016.

[42] Mingsheng Long, Guiguang Ding, Jianmin Wang, Jiaguang Sun, Yuchen Guo, and Philip S Yu. Transfer sparse coding for robust image representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 407–414, 2013.

[43] Mingsheng Long and Jianmin Wang. Learning transferable features with deep adaptation networks. *CoRR, abs/1502.02791*, 1:2, 2015.

[44] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144, 2016.

[45] Julien Mairal, Francis Bach, and Jean Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):791–804, 2012.

[46] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Discriminative learned dictionaries for local image analysis. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[47] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2272–2279. IEEE, 2009.

[48] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse representation for color image restoration. *IEEE Transactions on image processing*, 17(1):53–69, 2008.

[49] Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis R Bach. Supervised dictionary learning. In *Advances in neural information processing systems*, pages 1033–1040, 2009.

[50] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

[51] Lilyana Mihalkova, Tuyen Huynh, and Raymond J Mooney. Mapping and revising markov logic networks for transfer learning. In *AAAI*, volume 7, pages 608–614, 2007.

[52] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[53] Günther Palm. On associative memory. *Biological cybernetics*, 36(1):19–31, 1980.

[54] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[55] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

[56] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[57] Qiang Qiu, Vishal M Patel, Pavan Turaga, and Rama Chellappa. Domain adaptive dictionary learning. In *European Conference on Computer Vision*, pages 631–645. Springer, 2012.

[58] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

[59] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

[60] M. Schmidt. minfunc: unconstrained differentiable multivariate optimization in matlab. `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`, 2005.

[61] Sumit Shekhar, Vishal M Patel, Hien V Nguyen, and Rama Chellappa. Generalized domain-adaptive dictionaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 361–368, 2013.

[62] Xiaoxia Sun, Nasser M Nasrabadi, and Trac D. Tran. Supervised multilayer sparse coding networks for image classification. *arXiv preprint arXiv:1701.08349*, 2012.

[63] Annegreet van Opbroek, M Arfan Ikram, Meike W Vernooij, and Marleen De Bruijne. Transfer learning improves supervised image segmentation across imaging protocols. *IEEE transactions on medical imaging*, 34(5):1018–1030, 2015.

[64] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[65] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):1–40, 2016.

[66] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[67] William Whitney. *Disentangled Representations in Neural Models*. PhD thesis, Massachusetts Institute of Technology, 2016.

[68] David J Willshaw, O Peter Buneman, and Hugh Christopher Longuet-Higgins. Non-holographic associative memory. *Nature*, 1969.

[69] Pengcheng Wu and Thomas G Dietterich. Improving svm accuracy by training on auxiliary data sources. In *Proceedings of the twenty-first international conference on Machine learning*, page 110. ACM, 2004.

[70] Jun Yang, Rong Yan, and Alexander G Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 188–197. ACM, 2007.

[71] Meng Yang, Lei Zhang, Xiangchu Feng, and David Zhang. Fisher discrimination dictionary learning for sparse representation. In *2011 International Conference on Computer Vision*, pages 543–550. IEEE, 2011.

[72] Zhirong Yang, He Zhang, Zhijian Yuan, and Erkki Oja. Kullback-leibler divergence for nonnegative matrix factorization. *Artificial neural networks and machine learning–ICANN 2011*, pages 250–257, 2011.

[73] Qiang Zhang and Baoxin Li. Discriminative k-svd for dictionary learning in face recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2691–2698. IEEE, 2010.

[74] Shizhou Zhang, Jinjun Wang, Xiaoyu Tao, Yihong Gong, and Nanning Zheng. Constructing deep sparse coding network for image classification. *Pattern Recognition*, 64:130–140, 2017.

[75] Miao Zheng, Jiajun Bu, Chun Chen, Can Wang, Lijun Zhang, Guang Qiu, and Deng Cai. Graph regularized sparse coding for image representation. *IEEE Transactions on Image Processing*, 20(5):1327–1336, 2011.

[76] Sijun Zhou, Shizhou Zhang, and Jinjun Wang. Deep sparse coding network for image classification. In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*, page 24. ACM, 2015.

[77] Michael Zibulevsky and Barak A Pearlmutter. Blind source separation by sparse decomposition in a signal dictionary. *Neural computation*, 13(4):863–882, 2001.