An Algorithm for Transfer Learning in a Heterogeneous Environment

Andreas Argyriou[1], Andreas Maurer[2], and Massimiliano Pontil[1]

[1] Department of Computer Science
University College London
Malet Place, WC1E London, UK
{a.argyriou,m.pontil}@cs.ucl.ac.uk
[2] Adalbertstrasse 55
D-80799 München, Germany
andreasmaurer@compuserve.com

Abstract. We consider the problem of learning in an environment of classification tasks. Tasks sampled from the environment are used to improve classification performance on future tasks. We consider situations in which the tasks can be divided into groups. Tasks within each group are related by sharing a low dimensional representation, which differs across the groups. We present an algorithm which divides the sampled tasks into groups and computes a common representation for each group. We report experiments on a synthetic and two image data sets, which show the advantage of the approach over single-task learning and a previous transfer learning method.

Key words: Learning to learn, multi-task learning, transfer learning.

1 Introduction

Transfer learning uses the experience gathered from previous learning tasks in order to improve learning a new task. In the context of machine learning, past experience is provided by a collection of training sets, each sampled from a specific task. The underlying assumption is that the tasks belong to the same environment and share common properties. Uncovering these properties should thus enhance learning future tasks in the environment.

An important approach to transfer learning relies on the assumption that *all* the tasks are mutually related, in the sense that they share the same underlying representation, see [1, 2, 6, 7, 10, 13] and references therein. This requirement may be too strong for heterogeneous environments. As an illustrative example, consider object recognition involving different geometric invariances, such as rotation, scaling, illumination etc., where only one invariance is relevant for any given task.

The main contribution of this paper is a method to learn and represent the structure of such a heterogeneous environment. Our method naturally extends a previous method for multi-task and transfer learning with linear representations

[2, 10]. Furthermore, we connect this approach to previous work in the context of spectral regularization [3] and collaborative filtering [11].

Previous work on task clustering [5, 8, 12, 14] considers tasks to be related if the corresponding weight vectors are close to each other. In contrast, our approach assumes that tasks within the *same group* are related if their weight vectors span a low dimensional subspace. For example, in a binary classification task a target vector and its negative are far from each other in distance, but – lying in a one-dimensional subspace – closely related according to our assumption.

The paper is organized as follows. In Section 2, we introduce the transfer learning problem. In Section 3, we present our model for transfer learning over a heterogeneous environment. In Section 4 we describe the algorithmic implementation of the model. Next, in Section 5 we present numerical experiments with the algorithm. Finally, in Section 6 we present our conclusions.

2 Transfer Learning

We are interested in learning classification tasks, as they occur in a prescribed *environment*. For simplicity, we restrict ourselves to linear classification functions. However, our considerations apply to kernel methods as well as to regression or other learning problems.

Following [6], we regard an environment as a probability measure ρ on a set of learning tasks and, since the tasks we consider are described by weight vectors, we regard ρ as a probability measure on $\mathbb{R}^d$.

Information on the environment may be obtained by the following two-step procedure

- draw a weight vector $w \in \mathbb{R}^d$ from ρ
- generate a sample $\mathbf{z} \in \left(\mathbb{R}^d \times \{-1, 1\}\right)^m$ using w.

The vector w above corresponds to a classification function

$$f(x) = \operatorname{sign}\left(\langle w, x\rangle\right),$$

$x \in \mathbb{R}^d$. The sample (training set) $\mathbf{z} = ((x_1, y_1), \ldots, (x_m, y_m))$ is obtained by sampling the function f at m random locations $x_1, \ldots, x_m$ with labelling noise. The above procedure is then repeated n times, to yield a collection of n training sets

$$\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n).$$

Each of the $\mathbf{z}_t$ corresponds to a different classification task in the environment, for $t = 1, \ldots, n$. We have assumed, for simplicity, that the samples $\mathbf{z}_1, \ldots, \mathbf{z}_n$ have the same size, m, but what follows applies also when the sample sizes are different.

Transfer learning extracts structural knowledge from $\mathbf{Z}$, so as to enhance learning a future task drawn from the environment. This is particularly important when the number of examples for each task is relatively small in comparison

to the number of parameters. In this case, single-task learning – learning each task in isolation – leads to poor performance. However, if the tasks in the environment are related, transfer learning may work well [6, 10]. Thus, the problem is ultimately that of uncovering and exploiting relationships between the tasks.

As a very simple example, suppose that all the tasks are equal. In this case, a good transfer learning algorithm would combine all training sets to learn a single task. A more realistic situation is one in which all the tasks' weight vectors are a linear combination of a few feature vectors. A good transfer learning algorithm would learn these feature vectors from the data $\mathbf{Z}$. A classical approach, which goes back to work on learning to learn and multi-task learning [6, 7], is to search for such a low dimensional representation shared by all the tasks. Knowledge of the relevant features can reduce the burden of high dimensionality to the estimation of a small number of coefficients.

We consider a linear representation described by a matrix $T \in \mathbb{R}^{d\times d}$, which maps the raw representation x to the feature vector Tx. Our first step is to design a quantity which measures the performance of T relative to a task. If T is fixed, we learn a weight vector from a sample $\mathbf{z}$ by regularization, that is, we solve the problem

$$r(T, \mathbf{z}) = \min_{v\in\mathbb{R}^d} \left\{ \frac{1}{m} \sum_{i=1}^{m} \ell\left(\langle v, Tx_i\rangle, y_i\right) + \lambda\|v\|^2 + \|T\|_2^2 \right\} \tag{1}$$

where λ is a positive parameter, $\|v\|^2 = \langle v, v\rangle$, ℓ a loss function and $\|T\|_2$ denotes the Frobenius norm of matrix T. The minimizing vector in (1) – let us call it $v(T, \mathbf{z})$ – is then used in the classification of future inputs preprocessed by T. Note that, with this notation, the target vector w mentioned previously corresponds to $T^\top v$.

The term $\|T\|_2^2$ plays no role in finding $v(T, \mathbf{z})$, but it allows one to regard $r(T, \mathbf{z})$ as a measure of the learning performance of the representation T on the sample $\mathbf{z}$. Indeed, if the term $\|T\|_2^2$ were omitted, the quantity $r(\gamma T, \mathbf{z})$ would decrease in γ and would converge as $\gamma \to \infty$ to the minimal empirical error of a linear function on the data $\mathbf{z}$.

We now consider a set of tasks as represented by the multi-sample $\mathbf{Z}$. A good representation T is one which gives good learning performance, on average, over the tasks. Hence, we may define the quantity

$$R(T, \mathbf{Z}) = \frac{1}{n} \sum_{t=1}^{n} r(T, \mathbf{z}_t)$$

and learn T by solving the problem

$$\underset{T\in\mathbb{R}^{d\times d}}{\text{minimize}}\, R(T, \mathbf{Z}). \tag{2}$$

We note that this problem is conceptually equivalent to the multi-task feature learning algorithm in [2]. Let us denote by $H(\mathbf{Z})$ the minimum in (2). Intuitively, this quantity is a measure of heterogeneity of the training sets. That is, the smaller $H(\mathbf{Z})$ the less heterogeneous (more related) the tasks, in that it is possible to find a common representation which fits the training sets $\mathbf{Z}$ well.

3 Heterogeneous Environment

The approach outlined above relies on the assumption that *all* the tasks are mutually related in the sense that they share the same representation. This requirement may be too strong when the environment is heterogeneous.

3.1 Method

Assume that there are several groups of tasks, so that the tasks within each group are related but tasks from different groups have little in common. If $I \subseteq \{1, \ldots, n\}$ is the set of indices of the tasks within such a group, we expect the training data $\mathbf{Z}(I) := (\mathbf{z}_t)_{t \in I}$ to be highly related, that is, the heterogeneity measure $H(\mathbf{Z}(I))$ to be small.

Our goal therefore is to partition the training set $\mathbf{Z}$ into K groups that minimize average heterogeneity. For this purpose, we let $\mathcal{P}$ be the set of all partitions of size K of the set $\{1, \ldots, n\}$ and solve the problem

$$\underset{\{I_1,\ldots,I_K\}\in\mathcal{P}}{\text{minimize}} \sum_{k=1}^{K} \frac{|I_k|}{n} H(\mathbf{Z}(I_k)). \tag{3}$$

The factors $\frac{|I_k|}{n}$ weight each group in proportion to its size.

This approach can be equivalently seen as that of learning a library of K feature maps $\mathbf{T} = (T_1, \ldots, T_K)$, where each T_k is a $d \times d$ matrix representing the k-th group of tasks in the environment. To see this, we rewrite problem (3) as

$$\underset{\{I_1,\ldots,I_K\}\in\mathcal{P}}{\text{minimize}} \; \underset{T_1,\ldots,T_K}{\text{minimize}} \sum_{k=1}^{K} \frac{|I_k|}{n} R(T_k, \mathbf{Z}(I_k)). \tag{4}$$

Interchanging the minimization over the partitions and the matrices T_k, we obtain

$$\underset{T_1,\ldots,T_K}{\text{minimize}} \left\{ \frac{1}{n} \sum_{t=1}^{n} \min_{k=1}^{K} r(T_k, \mathbf{z}_t) \right\}. \tag{5}$$

This observation reformulates the combinatorial optimization problem (3) as a continuous optimization problem and is analogous to the passage from the assignment problem to the objective function of k-means clustering.

When $K = 1$ problem (5) reduces to problem (2). For $K > 1$, the minimization over k effects an assignment of tasks to groups. The h-th group consists of those tasks t for which the regularization error $r(T_k, \mathbf{z}_t)$ is minimal for $k = h$. Therefore, the matrix T_k is a common representation for the tasks of the corresponding group.

We now describe how the library $\mathbf{T}$ is used to learn a new task from a given training set $\mathbf{z}$. First, we compute the weight vectors $v(T_k, \mathbf{z})$ for $k = 1, \ldots, K$ and the associated minimal values $r(T_k, \mathbf{z})$ in (1). Second, we assign the task to the group

$$h = \arg\min_{k=1}^{K} r(T_k, \mathbf{z}).$$

The corresponding weight vector, $v(T_h, \mathbf{z})$, is then used for the classification of future data from the same task, preprocessed by T_h.

The two norms appearing in the definition (1) of $r\,(T, \mathbf{z})$ have an effect of complexity regularization, which we briefly sketch. The term $\|T\|_2^2$ controls the complexity of candidate libraries. If a large number of tasks have been observed in the past, the quantity minimized in (5) is, with high probability in $\mathbf{Z}$ and uniformly in all libraries $\mathbf{T}$, a good upper estimate for the quantity

$$\mathbb{E}_{\mathbf{z}} \min_{k=1}^{K} r\,(T_k, \mathbf{z})\,,$$

where the expectation is over a training set generated from a random task drawn from the environment. Similarly, the term $\|v\|^2$ regularizes the complexity of candidate weight vectors and has the effect that $r\,(T, \mathbf{z})$ is, for sufficient sample size, a good upper estimate for the expected classification error incurred by the use of $v(T, \mathbf{z})$. Combining these observations, one finds that the quantity minimized by our method is close to a high-probability upper bound on the error incurred by the above algorithm using $\mathbf{T}$ on future tasks.

These arguments, which give a statistical justification for our method, are made rigorous in [4] for a closely related model.

3.2 Connection to Spectral Regularization

We now explain why the learned representations $T_1, \ldots, T_K$ in (5) are encouraged to be low dimensional. We first analyse the case $K = 1$. Let us use the notation $W = [w_1, \ldots, w_n]$ if $w_1, \ldots, w_n \in \mathbb{R}^d$.

Lemma 1. *Problem* (2) *is equivalent to*

$$\underset{W \in \mathbb{R}^{d \times n}}{\text{minimize}} \left\{ \frac{1}{mn} \sum_{t=1}^{n} \sum_{i=1}^{m} \ell(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \|W\|_1 \right\} \tag{6}$$

where $\gamma = 2\sqrt{\lambda}$ *and* $\|W\|_1$ *is the* ℓ_1 *norm of the singular values of* W*. Moreover if* $\hat{W}$ *solves* (6) *and* $\hat{T}$ *solves* (2)*, then*

$$\hat{T}^\top \hat{T} = (\lambda \hat{W} \hat{W}^\top)^{\frac{1}{2}}.$$

Proof. We define the matrix $D = T^\top T$. If T is full rank, we have, for every training set $\mathbf{z}$, that

$$r(T, \mathbf{z}) = \min_{w \in \mathbb{R}^d} \left\{ \hat{\ell}(w, \mathbf{z}) + \lambda \langle w, D^{-1} w \rangle + \mathrm{tr} D \right\},$$

where $\hat{\ell}(w, \mathbf{z}) := \frac{1}{m} \sum_{i=1}^{m} \ell(\langle w, x_i \rangle, y_i)$. Thus, problem (2) becomes

$$\inf_{D \succ 0} \min_{W \in \mathbb{R}^{d \times n}} \left\{ \frac{1}{n} \sum_{t=1}^{n} \hat{\ell}(w_t, \mathbf{z}_t) + \lambda \mathrm{tr}(D^{-1} W W^\top) + \mathrm{trD} \right\}.$$

Interchanging the infimum with the minimum and following [2], the infimum over D is realized by $(\lambda W W^\top)^{\frac{1}{2}}$. The result then follows. □

We note that regularization with $\|W\|_1$, the trace norm, has originally been considered by [11] in the context of collaborative filtering. As shown in [9], the trace norm is the convex envelope of the rank function in the unit ball of matrices. This provides some intuition as to why the optimal matrix $\hat{W}$ (and, by Lemma 1, $\hat{T}$) has low rank.

We now consider the general case $K \geq 1$. If W is a $d \times n$ matrix and $I \subseteq \{1, \ldots, n\}$, we let $W(I) = [w_t : t \in I]$. The proof of the following result is established along similar lines as in Lemma 1.

Theorem 1. *Problem* (3) *is equivalent to the problem*

$$\underset{W \in \mathbb{R}^{d \times n}}{\text{minimize}} \left\{ \frac{1}{nm} \sum_{t=1}^{n} \sum_{i=1}^{m} \ell(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \min_{\{I_1, \ldots, I_K\} \in \mathcal{P}} \sum_{k=1}^{K} \|W(I_k)\|_1 \right\}.$$

The above theorem states that problem (3) is equivalent to a regularization problem in which the tasks are partitioned into groups, so that the associated weight vectors have small trace norm on average. Regarding the trace norm as an approximation of the rank, we interpret this regularization as favoring groups of tasks which lie in low dimensional subspaces.

4 Learning Algorithm

We now describe an algorithm for solving problem (3). The algorithm performs stochastic gradient descent on the objective function (5). At each iteration, it selects a task index $t \in \{1, \ldots, n\}$ at random and computes the gradient[3] of the function

$$\min_{k=1}^{K} r(T_k, \mathbf{z}_t) .$$

Only the matrix which realizes this minimum needs to be updated. This step requires the computation of the K vectors $v(T_k, \mathbf{z}_t)$ for the current values of the matrices $T_1, \ldots, T_K$, that is, the solution of K standard regularization problems. Thus, if the time complexity for solving each of these problems is $C(m, d)$ then the total time complexity per iteration is $O(KC(m, d) + md^2)$. This linear dependence on K is appealing in practice since it allows for more complex models. We also note that, since the optimal matrices in the library will be low dimensional, we may further accelerate the algorithm by using rectangular matrices T_k with a small but sufficient number of rows.

Although this algorithm is guaranteed to converge, the objective function (5) is non-convex and hence the limiting point is not necessarily a global solution. We note, in passing, that we are not aware of a single multi-task method on

[3] We ignore the issue of non-differentiability, because the objective function is almost everywhere differentiable.

Algorithm 1

Inputs: number of groups K, regularization parameter $\lambda > 0$, learning rate $\eta > 0$, training sets $\mathbf{z}_t = \{(x_{t1}, y_{t1}), \ldots, (x_{tm}, y_{tm})\},\ t = 1, \ldots, n$.
Initialization: Randomly choose $d \times d$ real matrices $T_1, \ldots, T_K$.
Repeat until convergence of the objective
  Draw t at random from $\{1, \ldots, n\}$
  for $k = 1, \ldots, K$ **do**
    Compute a solution $v(T_k, \mathbf{z}_t)$ of (1)
  end for
  Set $h = \arg\min_{k=1}^{K} r(T_k, \mathbf{z}_t)$, $\bar{v} = v(T_h, \mathbf{z}_t)$
  Set $T_h = T_h - \frac{\eta}{m} \sum_{i=1}^{m} \ell' \left(\langle \bar{v}, T_h x_{ti} \rangle, y_{ti} \right) \bar{v} x_{ti}^\top - 2\eta T_h$

heterogeneous task environments that is convex. However, for the purposes of finding a good local minimum, the following initialization heuristic has been observed to lead to good empirical results. First we train a single feature map ($K = 1$) until convergence. Then from this map two slightly mutated matrices are created to initialize the library with $K = 2$ and the process is repeated up to the actual K we aim for. As we shall see in Section 5, in our experiments the algorithm has always converged to a good local minimum, in that the great majority of the tasks were assigned to correct groups.

5 Experiments

In classification experiments performed with synthetic and real data, we have verified the following hypotheses.

- The learning algorithm correctly distinguishes the heterogeneous groups of tasks and determines the appropriate subspaces within each group.
- The algorithm improves significantly over that in [2] (case $K = 1$) in terms of the transfer error on new tasks.[4] Moreover, when allowing more complex models (that is, when K is larger than the actual number of groups in the data) the transfer error does not improve.
- The performance improves monotonically with the number n of tasks available for training and deteriorates with the number G of underlying groups.

In all the experiments, we have used the SVM hinge loss and tuned the regularization parameter λ using cross validation. Choosing K was also done by cross validation, as we discuss below.

5.1 Synthetic Data

Environment In the first experiment, we assume that the environment distribution ρ is a uniform mixture of a number G of group-specific measures ρ_k on

[4] Average misclassification error on new tasks.

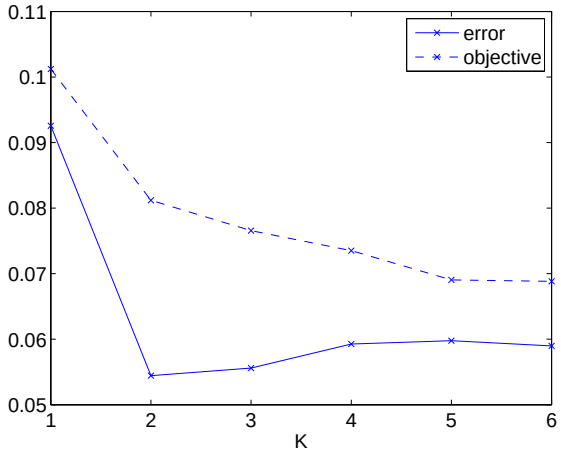

Fig. 1. Synthetic data (with $G = 2$). Plot of transfer error and objective function (5) versus K for $n = 1000$ tasks. The values of the objective have been rescaled to fit the plot. The error obtained using the identity matrix (training the tasks independently) was 0.35.

the unit sphere S^{d-1} in $\mathbb{R}^d$. In other words, $\rho = \frac{1}{G}\sum_{k=1}^{G} \rho_k$. A task is chosen by selecting the k-th group with probability $\frac{1}{G}$ and then drawing a task weight vector w from ρ_k.

We further assume that each measure ρ_k is concentrated and uniform on the intersection of a low dimensional subspace of $\mathbb{R}^d$ with S^{d-1} and that these subspaces are mutually orthogonal. Each task is a binary classification represented by a vector $w \in S^{d-1}$. An input x for each task is sampled uniformly on S^{d-1} and the outputs are obtained by taking the sign of $\langle w, x\rangle$.

Our experiment consists in using the training set sampled from the environment to identify the subspace on which each group of tasks lies and subsequently to reliably predict membership in such a subspace for new tasks drawn from the distribution ρ.

Results We first consider an environment with two groups of tasks (that is, $G = 2$), each of which lies on a 2-dimensional space of $\mathbb{R}^{100}$. We use $m = 50$ examples for training on each task and compute the transfer error over 200 new tasks sampled from the same environment, training on 50 examples and testing on 150 examples. In experiments with the synthetic data, the algorithm typically converged in less than 100,000 iterations.

As shown in Figure 1, using $K \geq 2$ yields a large improvement over grouping all tasks in the same group. Moreover, adding more than two groups in the model has a negative effect, the reason being that all matrices T_k are used in the obtained solution. This means that for $K > 2$ the objective function continues

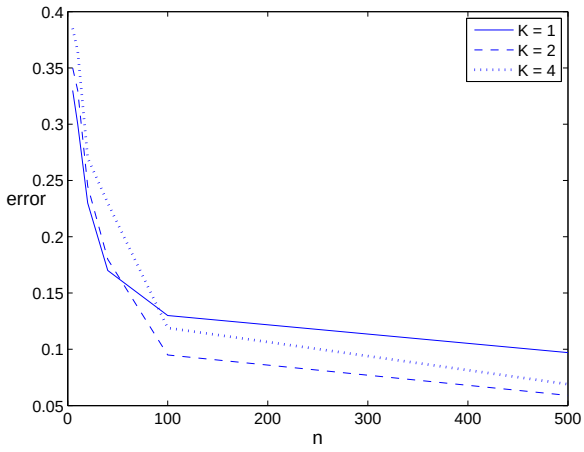

Fig. 2. Synthetic data (with $G = 2$). Plot of transfer error versus n using $K = 1, 2, 4$.

to decrease, as demontrated in the figure. Thus, K can be selected using cross validation on the training tasks but cannot be selected using the objective (5). Note that cross validation is meaningful when there is a sufficient number of related tasks, even if the sample per task is small.

In the case $K = 2$, the resulting matrices T_k reflect the structure of the environment as each of them projects on a 2-dimensional subspace of $\mathbb{R}^{100}$. Specifically, we found that the *two* largest singular values of T_1 and T_2 account for 97% of the spectrum, whereas for $K = 1$ the *four* largest singular values are needed. In addition, we have verified that the tasks are assigned to the correct groups with 99% accuracy.

An interesting observation is that even for $K \neq 2$ the 4-dimensional support of the whole environment is correctly identified. Thus, for $K = 1$ the obtained matrix projects on a 4-dimensional subspace, whereas for $K = 4$ the matrices project on parts of this subspace.

Other issues relate to the effects of the dimensions of the problem on transfer error. In Figure 2, we verify the known result – see e.g. [6] – that the error decreases with n. Note that for small values of n, the method performs better with $K = 1$ than with $K = 4$ and for even smaller values, it outperforms $K = 2$. The reason is that with a small number of tasks the data may be insufficient for learning each of the actual two subspaces but sufficient for learning the joint subspace.

Another effect is that of G, the actual number of underlying subspaces. Our method performs well with values of G much larger than 2 but as G increases the problem inevitably becomes harder. In Figure 3, we plot the error obtained by training our method and tuning K for different values of G.

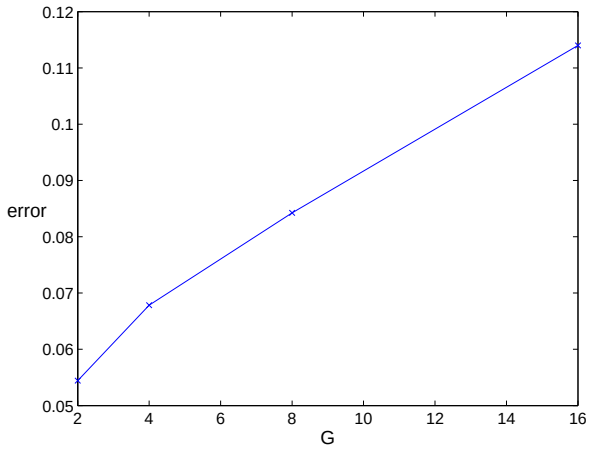

Fig. 3. Synthetic data. Plot of transfer error versus G using $n = 1000$.

5.2 Character Recognition Experiments

We now describe two experiments with images of handwritten characters[5], which further illustrate how our method works. The character images were obtained with a real camera. We also note that rotation/scaling (see below) of the character shown in an image was done mechanically during image acquisition. In the experiments, we have treated images simply as vectors of pixels and no use of image preprocessing techniques or special properties of image data was made.

Projection on Image Parts We consider the problem of invariant classification in the presence of noise. Every task is a pairwise classification of 28×56 images of characters. One half of the image contains the relevant character under an arbitrary rotation, whereas the other half contains a randomly chosen character also under an arbitrary rotation. There are two groups of tasks occurring with equal probability: tasks in which the relevant character appears on the left half of the image and tasks in which it appears on the right half. For example, the images of Figure 4 are examples of a task (classifying 6 versus 1) of the "right" group.

To train a library we selected pairs of alphabetic characters. Since we wish to obtain a library of features that represent the broader structural properties of the environment, namely rotation invariance on one half of the image with simultaneous irrelevance on the other half, we tested the learned library on samples generated from the digit character set (after removing digit 9). In this way we directly measured how well the representation transferred to novel but

[5] Available at `http://www.andreas-maurer.eu/similarity.htm`

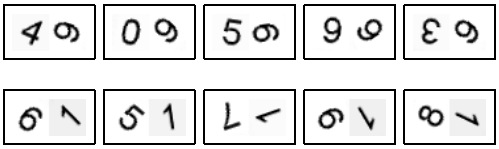

Fig. 4. Character recognition (left-right data set). Example images from one classification task (6 versus 1) in the group that focuses on the right part of the image.

Independent	$K = 1$	$K = 2$
0.27	0.036	0.013

Table 1. Character recognition (left-right data set). Transfer error for different methods.

structurally similar problems. We trained our algorithm until convergence, which was achieved in 400,000 iterations.

To measure the transfer error, we generated 500 tasks of pairwise classification from the digits data set. For each of these tasks, a sample of size 10 was chosen to train a classifier using the library found during the training phase. The performance of this classifier was then tested on 40 examples chosen from the same task. The high dimensionality of the input space (1568) and the small sample size (10) explain the high error (0.27) of learning each task independently, as Table 1 shows. At the same time, the large number of tasks observed for the training of the feature maps (1000) accounts for the spectacular improvement obtained by transfer learning ($K = 1$).

A convenient way to visualize the effect of the resulting maps on the image vector is to display their right singular vectors as 28×56 images. In Figure 5, we present these after normalization to the range $[-1, 1]$ and mapping of zero to gray. We show only the singular vectors corresponding to the 6 largest singular values for T and the 3 largest singular values for T_1 and T_2. The concentric features in both halves clearly reflect the rotational invariance properties present in the training data. In the single map case, the map does not distinguish the relevance of each part of the image relative to the tasks, whereas in the two-map case, matrix T_1 represents the invariance properties of the "left" group of tasks and matrix T_2 those of the "right" group. In addition, the singular values of T, T_1 and T_2 (Figure 6) show that this specialization of T_1 and T_2 results in more concise representations within each group of tasks. That is, the effective ranks of T_1 and T_2 equal 4, whereas that of T equals 8 and hence, setting $K = 2$ allows us to learn subspaces with lower dimensionalities.

Finally, in Table 2 we verify that the assignment of tasks into groups reflects the left-right structure of the data set. Here, the "Left" ("Right") percentage was measured over the sample that corresponds to classification on the left (right) half of the image.

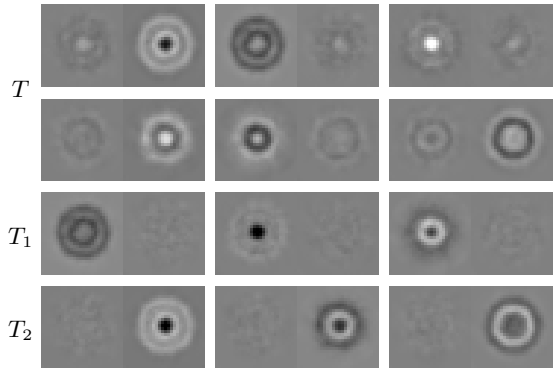

Fig. 5. Character recognition (left-right data set). Dominant right singular vectors of: T learned with $K = 1$ (top); T_1 and T_2 learned with $K = 2$ (bottom). See text for description.

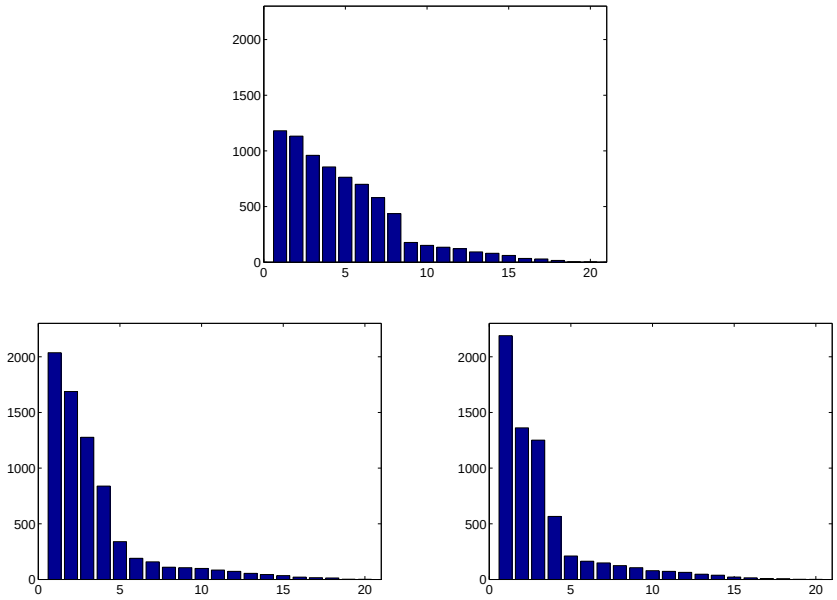

Fig. 6. Character recognition (left-right data set). Spectrum of T learned with $K = 1$ (top); spectra of T_1 (bottom-left) and T_2 (bottom-right), learned with $K = 2$.

Rotation and Scale Invariance Our final experiment is set in an environment which contains a mixture of tasks involving rotation invariant and scale invariant character recognition, with scale factors ranging from 2/3 to 3/2 (see Figure 7).

	T_1	T_2
All digits (left & right)	48.2%	51.8%
Left	99.2%	0.8%
Right	1.4%	98.6%
Training data	50.7%	49.3%

Table 2. Character recognition (left-right data set). Assignment of tasks in groups, when training with $K = 2$. The first three rows show percentage of assignments for the transfer tasks, the last row for the training tasks.

As in the previous experiment, our libraries were trained using pairwise classification of alphabetic characters and were transferred on pairwise classification of digits. Because of the difficulties caused by scale invariance, the images were preprocessed using a Gaussian kernel, with 1000 centers chosen randomly from the training data and Gaussian kernel width of $1/\sqrt{8}$. Again, 400,000 iterations were needed for training the algorithm.

Unlike the previous experiments, it is now unclear what the features relevant to each group of tasks may be, or how the feature space relevant to rotation invariant recognition relates to that of scale invariant recognition. To investigate the potential of our method we ask two questions:

1. Is the grouping method (with $K = 2$) able to improve the transfer performance of the more standard transfer learning algorithm with $K = 1$?
2. To what extent does the grouping reflect our intuition of rotation and scale invariant problems as corresponding to two distinct groups of tasks?

For training our method, we generated 2000 samples of size 10 (5 per class) from the alphabetic set, selecting rotation and scale invariant tasks with equal probability. The transfer data was generated as in the previous experiment: 500 tasks drawn from the digits, 10 training examples per task, 40 examples for testing.

As shown in Table 3, we observe again a dramatic improvement from independent learning to transfer learning. There is also an observable advantage of the grouping method but not as pronounced as in the previous experiments. The most likely explanation is that the two feature subspaces may now be far from orthogonal, unlike the previous experiments.

To answer the second of the above questions we counted the assignments to T_1, T_2 within exclusively rotation or scale invariant tasks (Table 4). We see a certain specialization of T_2 to rotation invariant tasks and of T_1 to scale invariant tasks, but it is not as clearly defined as in the previous experiment. Here it is important to realize that grouping into rotation and scale invariant tasks agrees with a certain human intuition, but there may well be other task-grouping criteria (for example rounded characters versus characters with sharp corners) which may be necessary for further improving performance.

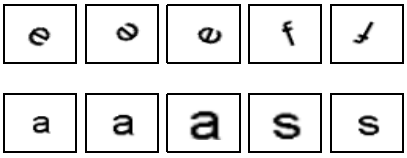

Fig. 7. Character recognition (rotation-scaling data set). Example images from a classification task in the rotationally invariant group (top) and one in the scale invariant group (bottom).

Independent	K=1	K=2
0.17	0.018	0.015

Table 3. Character recognition (rotation-scaling data set). Transfer error for different methods.

	T_1	T_2
All digits (rotation & scale)	44.8%	55.2%
Rotation invariance	10%	90%
Scale invariance	72%	28%
Training data	46.5%	53.5%

Table 4. Character recognition (rotation-scaling data set). Assignment of tasks in groups, when training with $K = 2$. The first three rows show percentage of assignments for the transfer tasks, the last row for the training tasks.

6 Summary

We have presented a method for transfer learning over environments in which tasks are concentrated on a number of low dimensional subspaces (*heterogeneous environments*). Our approach, which is justified theoretically by a generalization bound on the transfer error [4], uses gradient descent to learn a library of feature maps that describe such subspaces. The method naturally extends previous work on multi-task learning which considered only one common group of tasks [2].

We have reported experiments with synthetic and real data. These experiments are illustrative examples of the many real problems in which multiple heterogeneous tasks occur. They clearly demonstrate that our method is effective in identifying both *the groups of tasks* and *the underlying common feature maps*. Moreover, the method outperforms both single-task learning and the precursor method [2], which corresponds to the case that $K = 1$. We believe the work is a significant improvement over [2] (which in turn has shown state-of-the-art results in a number of benchmark data sets), since any algorithm using $K = 1$ cannot distinguish the different subspaces on which the tasks may lie.

We have also briefly sketched an interpretation of our approach in terms of spectral regularization. We speculate that following this observation our method

can be easily applied in the context of collaborative filtering, see e.g. [11]. Another interesting question that can be the topic of future research is to study conditions on the environment which ensure convergence of the algorithm to a good local minimum.

References

1. R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, *6*, pp. 1817–1853, 2005.
2. A. Argyriou, T. Evgeniou, M. Pontil. Convex multi-task feature learning. *Machine Learning*, 2008. http://www.springerlink.com/content/161105027v344n03
3. A. Argyriou, C. A., Micchelli, M., Pontil, Y. Ying. A spectral regularization framework for multi-task structure learning. *Advances in Neural Information Processing Systems*, 2007.
4. A. Argyriou, A. Maurer, M. Pontil. *Generalization Bounds for Task Grouping*, (Technical Report), University College London, February 2008.
5. B. Bakker and T. Heskes. Task clustering and gating for Bayesian multi–task learning. *Journal of Machine Learning Research*, *4*, pp. 83–99, 2003.
6. J. Baxter. A model for inductive bias learning. *Journal of Artificial Intelligence Research*, *12*, pp. 149–198, 2000.
7. R. Caruana. Multi–task learning. *Machine Learning*, *28*, pp. 41–75, 1997.
8. T. Evgeniou, C. A. Micchelli, M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, *6*, pp. 615–637, 2005.
9. M. Fazel, H. Hindi, S.P. Boyd. A rank minimization heuristic with application to minimum order system approximation. *Proceedings, American Control Conference*, pp. 4734–4739, 2001.
10. A. Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, *7*, pp. 117–139, 2006.
11. N. Srebro, J. D. M. Rennie, T. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pp. 1329–1336. MIT Press, 2005.
12. S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The TC algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 489-497, 1996.
13. A. Wilson, A. Fern, S. Ray and P Tadepalli. Multi-task reinforcement learning: a hierarchical Bayesian approach, Proceedings of the Twenty-Fourth International Conference on Machine Learning, pp. 1015–1022, 2007.
14. Y. Xue, X. Liao, L. Carin, B. Krishnapuram. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, *8*, pp. 35–63, 2007.