Clustering Learning Tasks and the Selective Cross-Task Transfer of Knowledge

Sebastian Thrun Joseph O'Sullivan

November 1995
CMU-CS-95-209

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

The first author is also affiliated with the Computer Science Department III of the University of Bonn, Germany, where part of this research was carried out.

This research is sponsored in part by the National Science Foundation under award IRI-9313367, and by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant number F33615-93-1-1330. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of NSF, Wright Laboratory or the United States Government.

Keywords: Clustering, concept learning, knowledge transfer, lifelong learning, machine learning, object recognition, supervised learning

Abstract

Recently, there has been an increased interest in machine learning methods that learn from more than one learning task. Such methods have repeatedly found to outperform conventional, single-task learning algorithms when learning tasks are appropriately related. To increase robustness of these approaches, methods are desirable that can reason about the relatedness of individual learning tasks, in order to avoid the danger arising from tasks that are unrelated and thus potentially misleading.

This paper describes the task-clustering (TC) algorithm. TC clusters learning tasks into classes of mutually related tasks. When facing a new thing to learn, TC first determines the most related task cluster, then exploits information selectively from this task cluster only. An empirical study carried out in a mobile robot domain shows that TC outperforms its unselective counterpart in situations where only a small number of tasks is relevant.

1 Introduction

One of the exciting new developments in the field of machine learning are algorithms that can gradually improve their ability to learn when applied to a sequence of learning tasks. Motivated by the observation that humans encounter more than one learning task during their lifetime and that they successfully improve their ability to learn [2, 20], several researchers have proposed algorithms that are able to acquire domain-specific knowledge and re-use it in future learning tasks. For example, in the context of face recognition methods have been developed that improve the recognition accuracy significantly when learning to recognize a face, by learning and transferring face-specific invariances acquired through other face recognition tasks [6, 16]. Similar results in the context of object recognition, robot navigation and chess are reported in [31].

Technically speaking, the underlying learning problem can be stated as follows. Given

1. training data for the current learning task,
2. training data for N other, previous learning tasks, and
3. a performance measure

find a hypothesis which maximizes the performance in the current (the $N+1$-th) learning task. Notice that item 2 in this list, the data of previous learning tasks (called *support data* or *support tasks*), does not appear in the usual formulation of machine learning problems. This is because support data might only be indirectly related, *e.g.*, carry different class labels. To utilize this data, mechanisms are required that can acquire and re-use domain-specific knowledge in order to guide the generalization in a knowledgeable way. To date, there is available a variety of strategies for the transfer of domain-specific knowledge across multiple learning tasks:

- learning internal representations for artificial neural networks, *e.g.*, [1, 5, 9, 22, 23, 25, 26, 29, 27],
- learning distance metrics, *e.g.*, [4, 18, 33],
- learning to re-represent the data, *e.g.*, [14, 33],
- learning invariances in classification, *e.g.*, [6, 16, 31],
- learning algorithmic parameters and choosing algorithms, *e.g.*, [24, 30, 35], and

1. *For each pair of support tasks n and m: Compute the performance gain for task n, if knowledge is transferred from task m.*

2. *Arrange all tasks into a small number of clusters by maximizing the performance gain within each task cluster.*

3. *If a new task arrives, determine the most similar task cluster. Selectively transfer knowledge from that task cluster only.*

Table 1: TC: The general scheme for task clustering.

- learning domain models, *e.g.*, [15, 21, 31].

Many of these approaches have been demonstrated to reduce the sample complexity, given that the number of available support tasks N is sufficiently large and, equally importantly, that they are appropriately related. What it means for tasks to be related is currently not very well understood.

The relatedness of learning tasks cannot be seen independently of the particular learning algorithm. Two tasks that might be appropriately related for one algorithm may well be unrelated for another. For support tasks to improve the generalization accuracy, one has to provide a set of learning tasks that are known to be appropriately related with respect to the learning algorithm. This places a burden on the programmer, as he/she has to be knowledgeable about the learning algorithm and the relation of the learning tasks. To weaken this requirement, it is desirable to design algorithms that can discover the relation between multiple learning tasks, so that when a new task arrives they can identify the most strongly related tasks and use only those for the transfer of knowledge.

This paper describes the TC (task clustering) algorithm. TC transfers knowledge selectively. The general principle is outlined in Table 1. To impose structure on the space of learning tasks, tasks are grouped into clusters of related tasks. Relatedness is defined as the effect of transferring domain knowledge from one learning task to another: The more the performance in task n improves through knowledge transferred from task m, the more related they are (Step 1). Based on the performance gain, Step 2 clusters the tasks into bins, so that within each bin the relatedness is maximal. As a result, tasks for which the transfer mechanism produces the most mutual leverage are grouped together. When a new task arrives

(Step 3), the algorithm in Table 1 first determines the most related task cluster. It then transfers knowledge only from this single cluster—other task clusters are not employed.

The TC implementation presented here instantiates the general scheme shown in Table 1. At the lowest level of learning, TC uses K-nearest neighbor (KNN) for generalization [10, 28]. KNN memorizes all training data explicitly and interpolates them at query time. TC transfers knowledge across tasks by adjusting the distance metric that is used for determining the proximity of data points in KNN (see also [3, 4, 11, 12, 18, 33]). To determine task relatedness, TC inspects the effect on the generalization accuracy for each task when using the distance metric that is optimally adjusted for another task (Step 1). It then clusters tasks so as to maximize the accuracy gain within each of the clusters (Step 2). Based on the results of the clustering, TC determines an optimal distance metric for each task cluster. When a new task arrives (Step 3), TC inspects all distance metrics (one for each cluster) and picks the one that works best. This clustering strategy enables TC to handle multiple classes of tasks, each of which is characterized by a different distance metric.

To elucidate TC in practice, this paper also reports the results of a series of experiments carried out in a mobile robot domain. The three key results of this empirical study are:

1. The sample complexity of KNN can be reduced significantly when the distance metric is learned from previous, related tasks.

2. TC reliably succeeds in partitioning the task space into (a hierarchy of) meaningful, related tasks.

3. Selective transfer significantly improves the results in cases where only few support tasks are relevant, yet does not hurt the performance when all support tasks are appropriately related.

The remainder of this paper is organized as follows. Section 2 describes the TC algorithm in detail. Empirical results obtained in a mobile robot domain are described in Section 3. Finally Section 4 discusses some of the strengths and weaknesses of the approach and outlines open questions.

2 The TC Algorithm

2.1 Nearest Neighbor

K-nearest neighbor (KNN) is a well-known method for fitting functions [10, 28]—we will therefore review it only briefly here. Suppose one would like to approximate the function $f(\cdot)$ based on a finite and potentially noisy set of input-output examples of f. KNN approximates $f(\cdot)$ by searching this set for the k nearest neighbors and returns their average output value. More specifically, suppose x is a query point for which one would like to know the value $f(x)$. KNN first searches the nearest K data points $y_1, y_2, \ldots, y_K$:

$$\left\{ y \;\middle|\; \begin{array}{l} \nexists z_1, \ldots, z_K \text{ such that } \\ \end{array} \begin{array}{l} \forall i, j, i \neq j : z_i \neq z_j \\ \wedge \;\; \forall i : z_i \neq y \;\; \wedge \\ \mathit{dist}(x, z_i) < \mathit{dist}(x, y) \end{array} \right\} \tag{1}$$

according to some distance metric $\mathit{dist}(\cdot, \cdot)$. KNN estimates $f(x)$ by averaging $f(y_1), \ldots, f(y_K)$:

$$f(x) \;\; = \;\; \frac{1}{K} \sum_{k=1}^{K} f(y_k) \tag{2}$$

Notice that for discrete output spaces (as is the case in the experiments reported here) the result of KNN is truncated.

The generalization properties of KNN depend on the choice of the distance metric. A popular choice is the Euclidean (L_2) distance metric:

$$\mathit{dist}_{\text{Euclid}}(x, y) \;\; = \;\; \sqrt{\sum_i \left(x^{(i)} - y^{(i)} \right)^2} \tag{3}$$

Here the superscript $^{(i)}$ is used to refer to the i-th component of a vector. Notice that the Euclidean metric weighs each data dimension equally.

Following the ideas in [3, 4, 11, 12, 18, 33], the TC algorithm uses a globally weighted version of $\mathit{dist}_{\text{Euclid}}$

$$\mathit{dist}_d(x, y) \;\; = \;\; \sqrt{\sum_i d^{(i)} \left(x^{(i)} - y^{(i)} \right)^2}. \tag{4}$$

Here d (with $d^{(i)} \geq 0$ for all i) is a vector of parameters that determine the relative weight of each input dimension. Obviously, different vectors d will make KNN interpolate differently. Thus, the vector d determines how KNN generalizes. Learning d has been shown empirically to be an effective way to transfer knowledge across multiple learning tasks [4, 33].

2.2 Determining the Optimal Distance Metric

Each task (or each set of tasks, as discussed below) possesses an optimal distance metric. An optimal metric is one that, given a finite sample set, minimizes the error on future data, assuming that all data is generated using the same (unknown) probability distribution. Strictly speaking, to estimate the optimal d one has to estimate the generalization error on future data, and adjust d to minimize this error.

Since in practice this error cannot be computed exactly, TC uses a slightly different criterion for determining d. Given a set of training examples, TC minimizes the distance between examples that belong to the same class, and at the same time maximizes the distance between examples that belong to different ones. In other words, d is obtained by minimizing the following expression:

$$E(d) \quad = \quad \sum_{x,y} \delta_{xy} \; dist_d(x, y) \tag{5}$$

where

$$\delta_{xy} \quad = \quad \begin{cases} 1 & \text{if } f(x) = f(y) \\ -1 & \text{if } f(x) \neq f(y) \end{cases} \tag{6}$$

Here each component $d^{(i)}$ is constrained to lie in $[0.01, 1]$ for all i.[1] Let

$$d^* \quad = \quad \underset{d}{\text{argmin}}\, E(d) \tag{7}$$

denote the parameter vector that minimizes E, henceforth called *E-optimal* or *task-optimal*. Let *dist** be the corresponding optimal distance metric. By minimizing the infra-class distance (*dist*$_d(x, y)$ when $f(x) = f(y)$) and simultaneously maximizing the inter-class distance (*dist*$_d(x, y)$ when $f(x) \neq f(y)$), *dist** focuses on the relevant input dimensions for that particular learning task. Since $E(d)$ is monotonic in d, d^* can be found using gradient descent.

Notice that d can be optimized simultaneously for multiple learning tasks. Let $A \subset \{1, 2, \ldots N\}$ denote a subset of the support tasks. Then

$$d^*_A \quad = \quad \underset{d}{\text{argmin}} \sum_{n \in A} E_n(d) \tag{8}$$

is the E-optimal parameter vector and *dist*$^*_A$ the corresponding distance metric for the task set A. The subscript n in (8) indicates that $E_n(d)$ is computed using the training data of the n-th learning task.

[1] In principle, it suffices to bound $d^{(i)}$ below by 0. In practice, however, the data set-optimal d often considers only an extremely small number of input dimensions that happen to be discriminative for the training set. This is particularly the case when the input space is high-dimensional and training data is scarce (as is the case in most of our experiments). To prevent this type over-fitting, it is beneficial to constrain $d^{(i)}$ to lie in some interval that does not include 0.

2.3 Clustering Tasks

Following Steps 1 and 2 in Table 1, TC arranges the different learning tasks into a disjoint set of task clusters. The incentive to do so is the expected gain of generalization accuracy when a task uses another task's E-optimal distance function. More specifically, TC generates the *task transfer matrix*

$$C \quad = \quad (c_{n,m}) \tag{9}$$

which has an entry $c_{n,m}$ for each pair of learning tasks n and m. The value $c_{n,m}$ is the expected generalization accuracy for task n when using m's E-optimal distance metric $dist^*_m$. Each element $c_{n,m}$ is computed by cross-validation as follows:

1. Firstly, the E-optimal distance metric $dist^*_m$ for task m is determined by minimizing $E_m(d)$.

2. Secondly, nearest neighbor is applied to task n using $dist^*_m$. To estimate the generalization accuracy, the sample set of the n-th support task is repeatedly split into a training and the testing set. $c_{n,m}$ is measured using the test set, averaged over all splits.

Let $A_1 \ldots A_T$ denote a disjoint decomposition of all support tasks into T clusters, *i.e.*, $A_1 \uplus A_2 \uplus \cdots \uplus A_T = \{1, \ldots, N\}$. The functional J

$$J \quad = \quad \frac{1}{N} \sum_{k=1}^{T} \sum_{n \in A_t} \frac{1}{|A_t|} \sum_{m \in A_t} c_{n,m} \tag{10}$$

measures the averaged estimated generalization accuracy that is obtained when each support task $n \in A_t$ uses the optimal distance metrics of all other tasks $m \in A_t$ in its cluster. Hence, maximizing J clusters tasks in such a way that the averaged generalization accuracy is maximal, assuming that distance metric are transferred within each cluster but not across them. Notice that each of the T clusters defines an E-optimal distance metric

$$d^*_{A_t} \quad = \quad \operatorname*{argmin}_d \sum_{n \in A_t} E_n(d). \tag{11}$$

which is obtained by minimizing E_{A_t} (*cf.* (8)). These T distance metrics—one for each task cluster—form the basis for the selective transfer described in the next section.

It remains to be discussed how to maximize J. In general, optimizing a functional J defined over a pairwise cost matrix is a well-understood combinatorial data clustering problem for which various algorithms exist (see for example [7, 13]

and references therein). In all our experiments, the set of support tasks N was sufficiently small so that the globally optimal solution could be computed explicitly. For larger N incomplete methods must be applied.

2.4 Selective Transfer

Each task cluster defines an E-optimal distance metric. By optimizing and using a single metric for *all* tasks in a cluster, knowledge is transferred within each task cluster.

Of particular interest in this paper are situations in which a new learning task arrives that may be related to previous learning tasks. In principal, there are two strategies to relate a new learning task to the set of existing ones.

1. **Non-incremental:** Every time a new task arrives, the set of all tasks (including the new one) is re-clustered. To achieve optimal results, this might require that tasks are clustered after the arrival of every single training example, which can be computationally very expensive.

2. **Incremental:** Instead of repeatedly clustering the task space, one can also leave the existing clusters unchanged and sort the new task into the most related task cluster. This strategy is faster, since it avoids the combinatorial problem of clustering N tasks. However, task clusters found by the incrementally approach are not necessarily optimal.

In the current implementation, TC employs both a non-incremental and an incremental strategy for clustering tasks. Once in a while (when the data collection for a learning task is finished), TC clusters all tasks into bins as described above. If an appropriate value for T is known, T clusters are generated. If T is unknown, however, TC builds the complete hierarchy of tasks by clustering the task space into $T = 1, \ldots, N$ clusters. In the worst case, there will be a total of

$$\sum_{T=1}^{N} T \quad = \quad \frac{N \cdot (N+1)}{2} \tag{12}$$

task clusters, but in our all experiments including those reported below TC generated much fewer clusters (25 or 27, with $N = 12$).

When a new task arrives, TC, determines the most related task cluster and uses that cluster's E-optimal distance metric. At first glance, to determine the most related cluster one could minimize the n-fold cross-validation error over all distance metrics for determining the most appropriate one. In our experiments, we adopted a different strategy that was consistently found to work slightly better.

To determine the most related task cluster, the TC algorithm maximizes the ratio between the inter-class and the infra-class distance

$$r(t) \quad = \quad \left(\sum_{x,y:\delta_{xy}=-1} dist^*_{A_t}(x,y) \right) \cdot \left(\sum_{x,y:\delta_{xy}=1} dist^*_{A_t}(x,y) \right)^{-1} . \qquad (13)$$

The value $r(t)$ divides the distance between instances of different classes by the distance for instances that fall into the same class. It gives a measure how well the two classes are separated by a particular distance metric $dist^*_{A_t}$. Thus, when a new task arrives, TC selects the distance metric $dist^*_{A_t}$ that maximizes $r(t)$, which it uses for nearest neighbor generalization.

To summarize the TC algorithm, TC transfers knowledge across multiple learning tasks by learning a distance metric on some tasks, then using it for nearest neighbor generalization in others. To focus the transfer on the most related tasks, TC clusters the support tasks space into sets of related tasks. When facing a new learning task, its distance metric is transferred from the most related task cluster.

3 Experimental Results

3.1 Setup

The TC algorithm was evaluated empirically using the mobile robot XAVIER shown in Figure 1. All our experiments employed the color camera (top of the robot) and the 24 sonar proximity sensors (arranged in a ring around the robot). The camera was pointed slightly downward.

The practical objective that drove the development of the TC algorithm was to design robust and computationally efficient learning algorithms that can be applied to a variety of perceptual tasks specifically in the context of mobile robot control. Therefore, we collected nine databases of sonar and camera snapshots of persons, landmarks, objects, and locations. Each database consisted of 100 snapshots (color camera images and sonar scans), examples of which are shown in Figures 2 and 3. The first four datasets were constructed with a particular person somewhere in front of the robot. Different persons wore different clothes, so that their recognition involved spotting certain colors. The next two databases contained generic sensor readings when the robot was in the laboratory or in the hallway, respectively. The seventh database contained images and sonar scans of a blue trash bin, and the final two databases consisted of snapshots of situations where the robot was facing an open and a closed door, respectively. Figure 3 illustrates some of the variations in the latter two database. When collecting the data, no attention was paid as to where

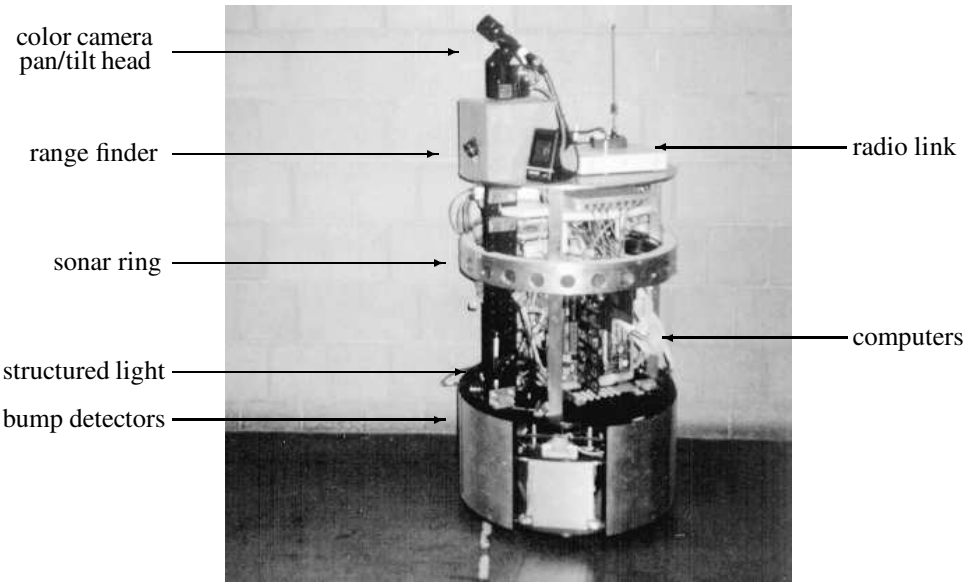


Figure 1: The mobile robot XAVIER.

in the image and how far away a person or a landmark occured. To keep the size of the data manageable, we down-sampled each image to a 10 by 10 matrix of RGB color triplets as shown in Figure 4. In all the experiments reported below, snapshots were represented by 300 values along with the 24 sonar scans, normalized to the unit interval.

To test TC under different conditions, we defined two families of 13 learning tasks. Task family $\mathcal{T}_1$ which is described in Table 2 consisted of various tasks involving the recognition of people, landmarks and locations. For reasons given below (Sect. 3.5), we used the thirteenth of these tasks, the recognition of open versus closed doors, as the testing task (*i.e.*, as the $N+1$-th learning task)—the other twelve tasks were used as support tasks.

Since all tasks in $\mathcal{T}_1$ are somewhat similar (they are based on the same sensors and they all address the recognition of similar features), we also assembled a second family of 13 tasks. Task family $\mathcal{T}_2$ consisted of

(a) tasks 2, 4, 5, and 6 from task family $\mathcal{T}_1$ (see Table 2),

(b) the same four tasks, but this time the input dimensions were permuted ran-

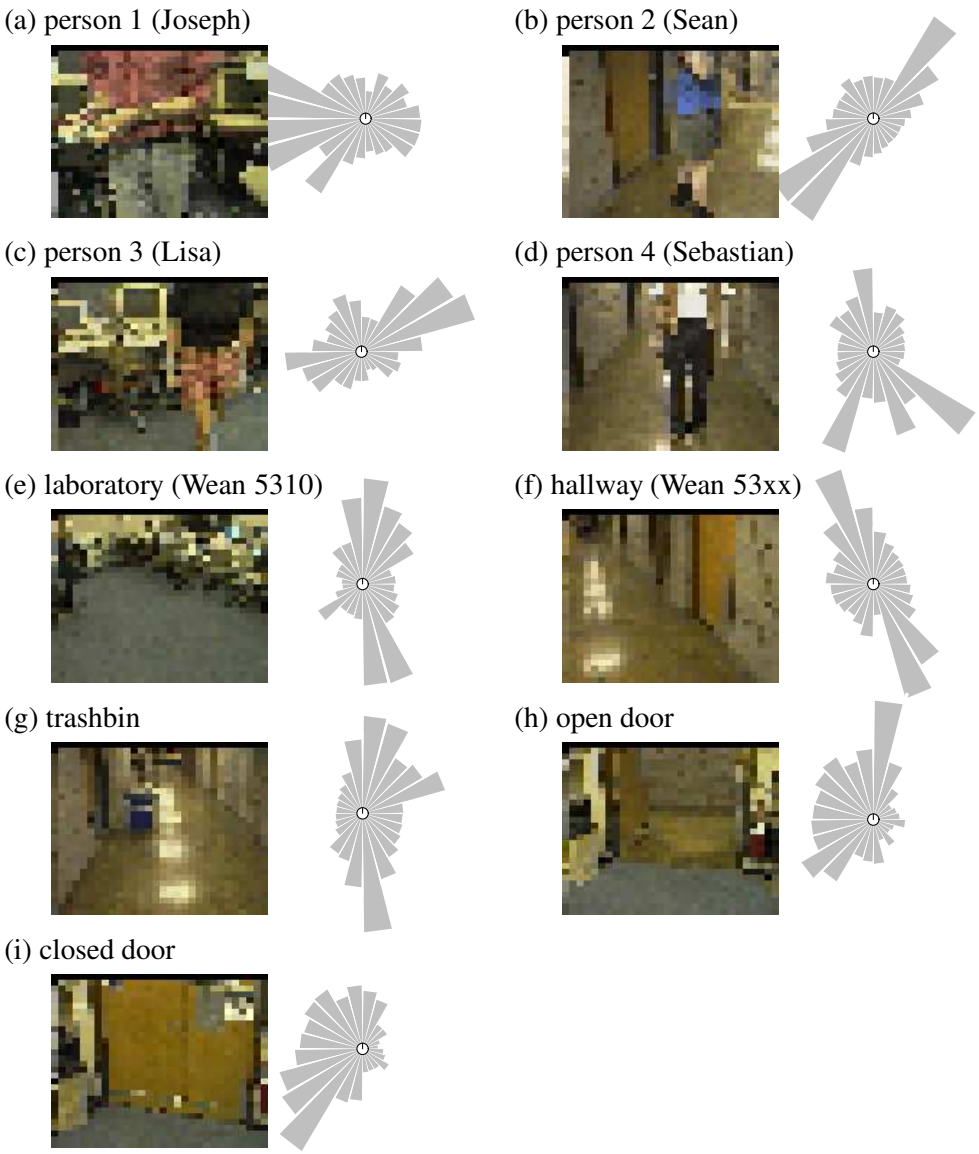


Figure 2: Examples of the nine databases (image and sonar scan).

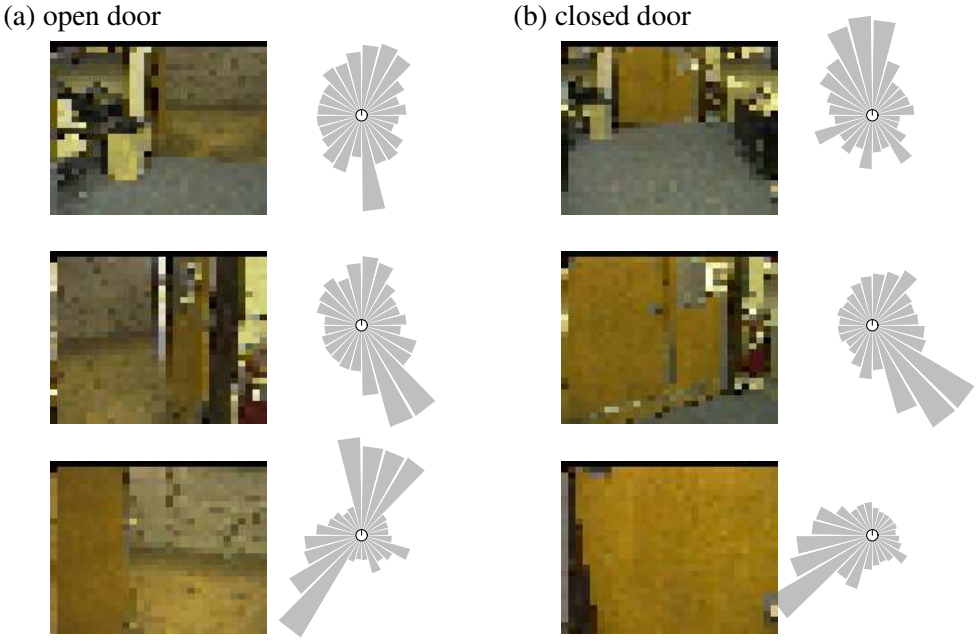


Figure 3: Variations in the data.

domly (all using the same permutation). These resulting tasks are called $2'$, $4'$, $5'$, and $6'$,

(c) a second, permuted version of tasks 2, 4, 5, and 6, but this time each tasks was permuted differently. These tasks are called $2''$, $4''$, $5''$, and $6''$, and

(d) the testing task 13 in Table 2, which was chosen to facilitate the comparison of the results obtained for both task families.

Task family $\mathcal{T}_2$, thus, contained only four tasks that were potentially related to the testing tasks. Four other tasks were mutually related but unrelated to task 13, and four tasks were neither related to task 13, nor mutually related. As we will see below, unselective transfer can suffer from such unrelated tasks.

When clustering tasks (Sect. 2.3), each value c_{mn} was estimated using 100-fold cross-validation with a training set of 5 examples per dataset and a testing set of 95 examples (per dataset). The distance metric (Sect. 2.2) was optimized by gradient descent, which was iterated for 100 steps using a step-size of 0.1 and a

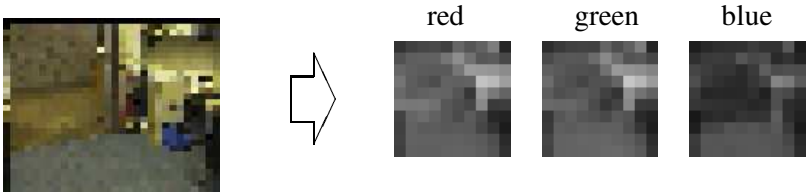


Figure 4: Down-sampling images to 10 by 10 red/green/blue values.

momentum of 0.9. Convergence, however, was consistently observed much earlier (often after 6 epochs). The results were not sensitive to these learning parameters. We also did not observe over-fitting, neither for the tasks that the distance metrics were optimized for, nor for any other task (such as the test task 13). The latter observation crucially depended on the existence of the bounds $[0.01, 1]$ for the distance parameters $d^{(i)}$. In the absence of such bounds, the performance on the training task increased only slightly, while it usually decreased significantly for any other task. All experimental results reported below are test set results (*i.e.*, performance was measured for data points that were not part of a training set). They are all averaged over 100 experiments using different sets of training examples. To illustrate the effect of transfer across tasks, we will compare the E-optimal distance metric with the equally-weighted Euclidean distance metric, which serves as the uninformed default metric in the absence of support tasks. Whenever appropriate, the diagrams also show 95% confidence bars for the true value. All performance graphs show the generalization accuracy (testing set accuracy) for the testing task 13.

3.2 Non-Selective Transfer

To elucidate the benefit and the limits of regular, unselective transfer, we first conducted experiments using all support tasks for computing the E-optimal distance metric. Unselective transfer can be understood as a special case of the TC algorithm in which the number of clusters T is set to 1.

The first key empirical result of this paper is shown in Figure 5. The thin curve in diagram 5a shows the generalization accuracy of the equally-weighted Euclidean distance metric for the testing task as a function of the number of training examples. This curve illustrates the accuracy of nearest neighbor in the absence of support tasks. The thick curve depicts the generalization accuracy using the E-optimal distance metric for all 12 support tasks. As can be seen from these graphs, the E-

	data set								
	Joseph	Sean	Lisa	Seb.	lab (no door)	hallway	trash bin (in hallway)	open door	closed door
1. Joseph/Sean	+	−							
2. Joseph/Lisa	+		−						
3. Joseph/Sebastian	+			−					
4. Sean/Lisa		+	−						
5. Sean/Sebastian		+		−					
6. Lisa/Sebastian			+	−					
7. Joseph/trash bin	+						−		
8. Sean/trash bin		+					−		
9. Lisa/trash bin			+				−		
10. hallway with/ without trash bin						+	−		
11. lab/hallway					+	−			
12. door/no door					+			−	−
13. door open/closed								+	−

Table 2: Task family $\mathcal{T}_1$. The first twelve tasks are support tasks, the thirteenth the testing task.

optimal distance metric shows significantly better results than the equally-weighted metric, illustrating the benefit of transferring knowledge across tasks.

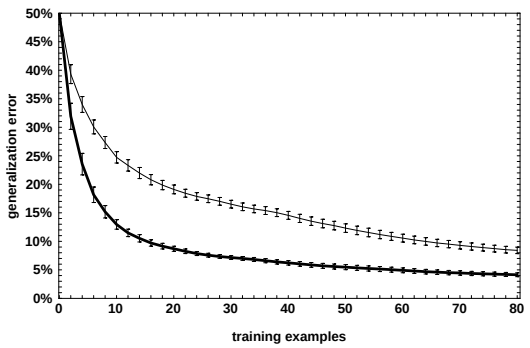
There are two ways to quantify these results.

1. **Relative generalization accuracy.** The generalization error averaged over different number training examples is shown in Figure 5b. The E-optimal distance metric infers an average error of 8.1%, which is only 50.3% of that of the equally-weighted distance metric (which is 16.0%). So for a fixed number of training examples one can expect the E-optimal distance metric to cut the error roughly in half.

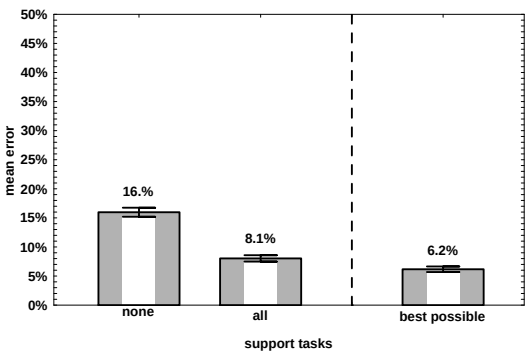
2. **Relative sample complexity.** The second quantity measures the reduction in sample complexity. Figure 5c shows the result of statistical comparisons of the generalization accuracies for the equally-weighted versus the E-optimal metric, using different number of training examples. In the white region,

$\mathcal{T}_1$, unselective transfer

(a) generalization error
equally weighted metric (thin)
vs. E-optimal metric (bold)



(b) average generalization error



(c) sample complexity
equally weighted metric
vs. E-optimal metric

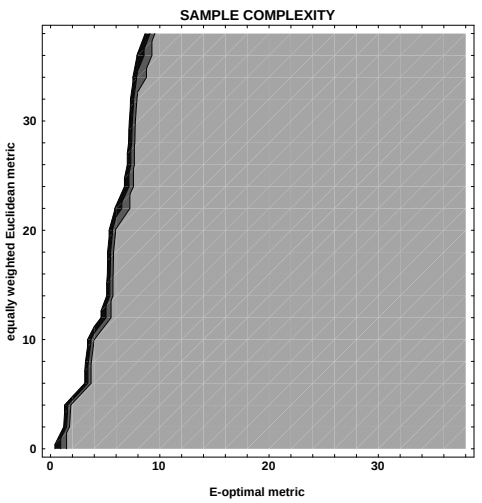
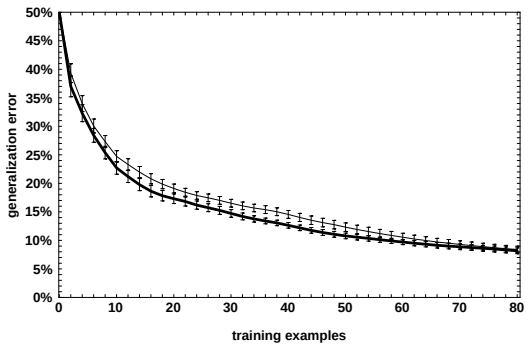


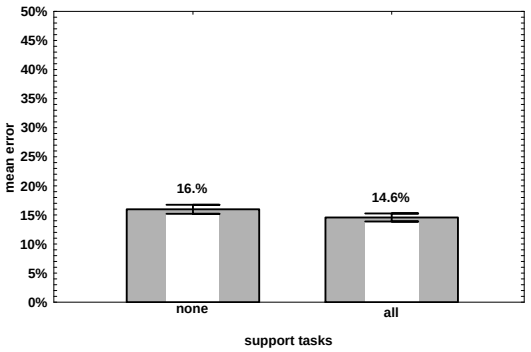
Figure 5: Generalization curves and (b) average generalization error of the testing task. (c) Statistical comparison of this error for the equally-weighted and the E-optimal distance metric with varying numbers of training examples. In the weight (grey) area the equally-weighted (E-optimal) distance metric is superior at the 95% confidence level. In the dark region between both methods generalize about equally well.

$\mathcal{T}_2$, unselective transfer

(a) generalization error
equally weighted metric (thin)
vs. E-optimal metric (bold)



(b) average generalization error



(c) sample complexity
equally weighted metric
vs. E-optimal metric

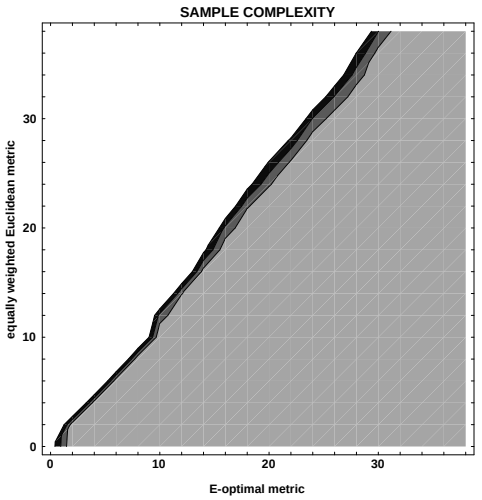


Figure 6: Unselective transfer in task family $\mathcal{T}_2$. Notice that the E-optimal distance metric is only slightly better than the equally-weighted metric.

the equally-weighted distance metric outperforms the E-optimal one with at least 95% confidence. In the large grey version the opposite is the case. In between, both methods work about equally well and their generalization accuracies did not differ significantly (at the 95% level). Notice that, on average, the E-optimal distance metric uses only 28.4% of the number of training examples that are required when using the equally-weighted Euclidean metric. For example, the average generalization accuracy of the E-optimal distance metric applied to 10 training examples is about the same as that of the Euclidean distance metric when 40 training examples are available. It is interesting to notice that the boundary in Figure 5c is approximately linear. This observation suggests that the reduction in sample complexity depends only weakly on the actual number of training examples.

To summarize, the generalization error when using the E-optimal distance metric is only 50.3% of that inferred by the equally-weighted Euclidean metric, and it requires only 28.4% of the samples that are required when using the equally-weighted metric for reaching a certain level of generalization accuracy. These results apply to task family $\mathcal{T}_1$.

The positive impact of the knowledge transfer depends crucially on the fact that the support tasks are sufficiently related to the test task. Task family $\mathcal{T}_2$, in which the majority of tasks is unrelated, fails to produce similar effects. As can be seen in Figure 6, the E-optimal distance metric is only slightly better than the equally-weighted counterpart. In particular, the generalization error is reduced to 91.2%, and the sample complexity is reduced to 86.6%, which is clearly less of an improvement than the corresponding values for $\mathcal{T}_1$ (50.3% and 28.4%). As will be shown in the next sections, by selectively transferring knowledge from the right cluster of tasks can improve these results greatly.

3.3 Clustering Tasks

Figure 7 shows a normalized version of the transfer matrix $(c_{n,m})$ for each task family. Each row depicts how a particular task n (including the testing task) benefits from knowledge transferred from task m. White boxes indicate that the generalization accuracy of task n improves when the E-optimal distance metric of task m is used instead of the equally-weighted distance metric. Black boxes indicate that the opposite is the case, meaning that tasks are "anti-related." The size of the box visualizes the magnitude of the effect.

In task family $\mathcal{T}_1$, most tasks are either related to the testing task or unrelated, but none of them is notably anti-related (first row in Figure 7a). The diagram for the

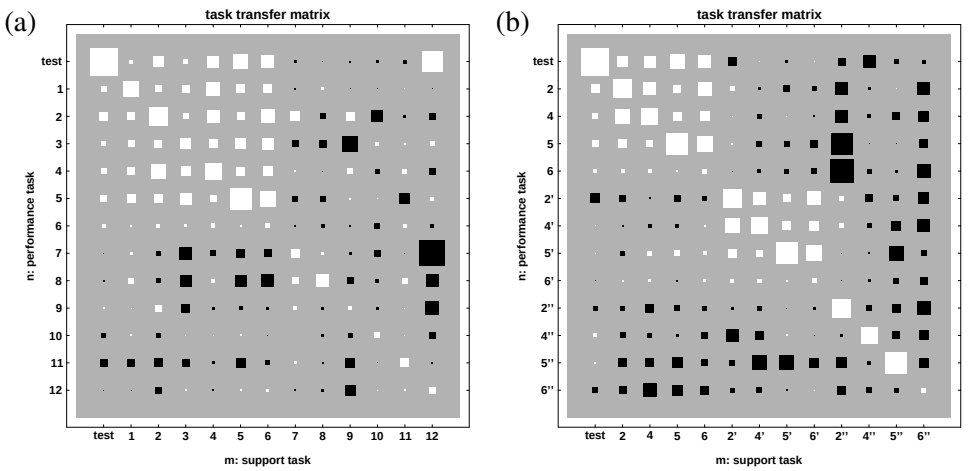


Figure 7: Task transfer matrix $(c_{n,m})$ for (a) task family $\mathcal{T}_1$ and (b) task family $\mathcal{T}_1$. White values show indicate that the error in task n is reduced when the E-optimal distance metric of task m is used. Black values indicate the opposite: tasks are anti-related. The relation of the individual tasks to the testing tasks is also depicted.

---

more interesting task family $\mathcal{T}_2$ shows that some tasks, in particular $2'$, $2''$, and $4''$, are anti-related to the testing task. In other words, using their respective E-optimal distance metrics will hurt the performance in the testing tasks. However, Figure 7a also shows that the non-permuted tasks 2, 3, 4, and 6 are indeed well-related to the testing task, showing that there exists the opportunity for synergy through knowledge transfer.

Figures 8 and 9 show the different clusters, found for both task families using different values for T. The task hierarchy for task family $\mathcal{T}_2$, which is depicted in Figure 9, illustrates the second key result of the empirical study: TC manages to discover meaningful tasks clusters. Early on in Figure 9, starting with $T = 3$ clusters, the two major task families that use the same encoding ($\{2, 4, 5, 6\}$ and $\{2', 4', 5', 6'\}$) are grouped together. For example, for T=4 partitions TC generates the following task clusters: $\{2, 4, 5, 6\}, \{2', 4', 5', 6'\}, \{2'', 4'', 6''\}, \{5''\}$. Here J=82.6%. For comparison, the worst partitioning with T=4 groups tasks into the following bins: $\{2, 5', 5''\}, \{4, 5, 4', 2'', 6''\}, \{6, 2', 4''\}, \{6'\}$ with J=77.6%. When $T \geq 4$ all three different task types are clustered into separate clusters. This finding illustrates that TC indeed manages to find meaningful clusters in $\mathcal{T}_2$, hence to discover the structure that inherently exists for the different tasks. A second interesting result depicted in Figure 9 is that TC with $T = 6$ groups exactly

Figure 8: The task hierarchy for task family $\mathcal{T}_1$. The hierarchy is obtained by clustering the task space using different numbers of clusters T. The right part of the diagram depicts the corresponding value for J, and the difference between the best and the worst partitioning.

---

those tasks together that rely on the same input encoding. Here the clusters are $\{2, 4, 5, 6\}$, $\{2', 4', 5', 6'\}$, $\{2''\}$, $\{4''\}$, $\{5''\}$, and $\{6''\}$. In task family $\mathcal{T}_1$, where the differences between different tasks are more subtle, it is interesting to note that the tasks involving the recognition of people form the most similar subgroup (particularly those involving two different people, *cf.* Figure 7a). When $T \geq 4$, those tasks that involve the recognition of a person (1 to 9) and than those that do not (10,11,12) are always arranged in different clusters.

3.4 Selective Transfer

Figures 10 and 11 summarize the results obtained using the TC algorithm in task family $\mathcal{T}_2$ using $T = 3$ and $T = 4$ clusters. Notice that these results can directly be

Figure 9: The task hierarchy for task family $\mathcal{T}_2$. Notice that early on, the task hierarchy separates the three different task types. The only related task cluster, $\{2, 4, 5, 6\}$, is identified when $T \geq 3$ clusters are available.

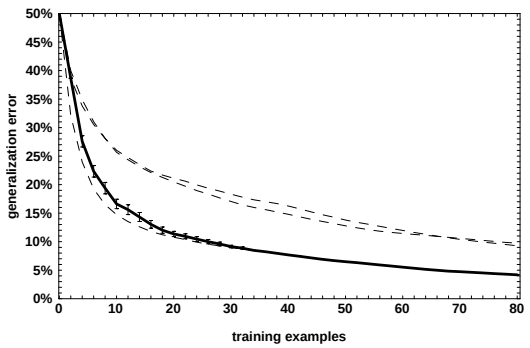
---

compared to those shown in Figure 6. The dashed curves in Figures 10a and 11a and the left bars in Figures 10b and 11b show for each task cluster the corresponding generalization accuracies in the test task when the cluster-optimal distance metric is used. In both experiments, only one of the clusters, namely $\{2, 4, 5, 6\}$, is appropriately related to the testing task, *i.e.*, leads to results that are better than those obtained with the equally-weighted distance function. If this were known in advance, TC would always use this task cluster to adjust the distance metric. However, this information is not available.

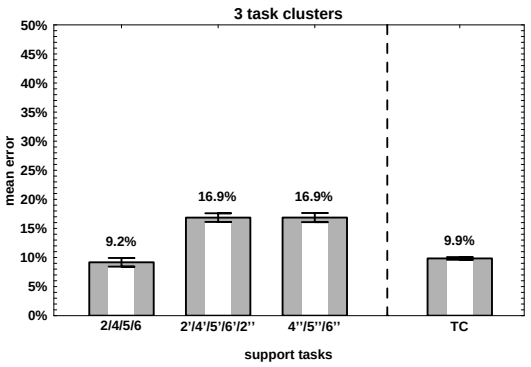
The results obtained with the TC algorithm are shown in Figures 10 and 11 (solid curve in diagrams (a), and right bars in diagrams (b)). Across the board, TC manages to guess the best task cluster considerably often, hence generalizes well. In the beginning of learning, the generalization accuracy of TC is slightly worse than that of the best task cluster, which is due to the fact that with a certain

$\mathcal{T}_2$**,** T**=3 clusters**

(a) generalization error
task clusters (dashed)
E-optimal metric (solid)



(b) average generalization error
individual task clusters (left)
and TC (right)



(c) sample complexity
equally weighted metric
vs. E-optimal metric

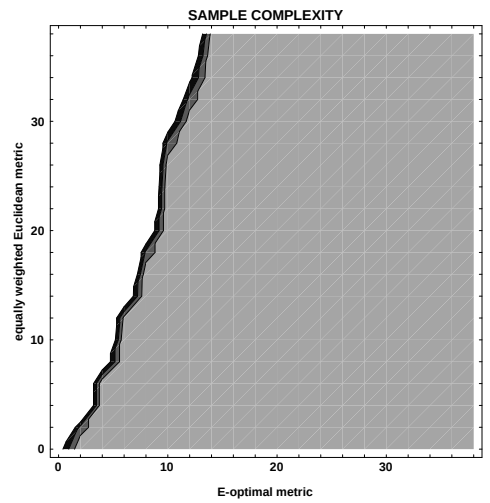
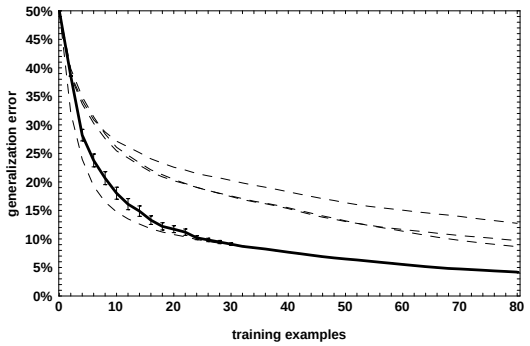


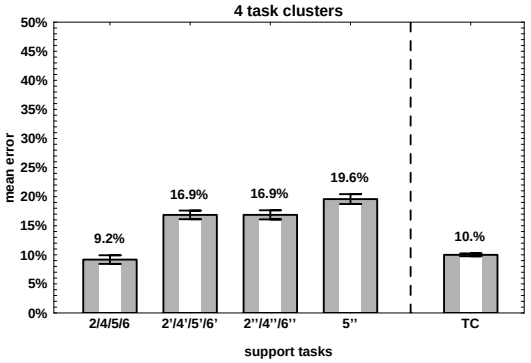
Figure 10: Selective transfer in task family $\mathcal{T}_2$ with $T = 3$ task clusters. The results are clearly superior to those obtained with unselective transfer. Notice also the difference between the different task clusters.

$\mathcal{T}_2$, T=4 **clusters**

(a) generalization error
task clusters (dashed)
E-optimal metric (solid)



(b) average generalization error
individual task clusters (left)
and TC (right)



(c) sample complexity
equally weighted metric
vs. E-optimal metric

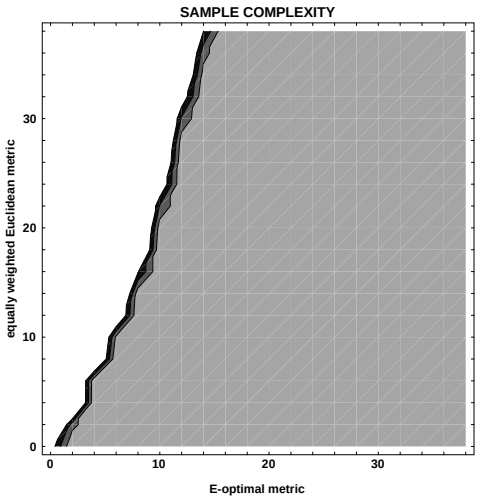
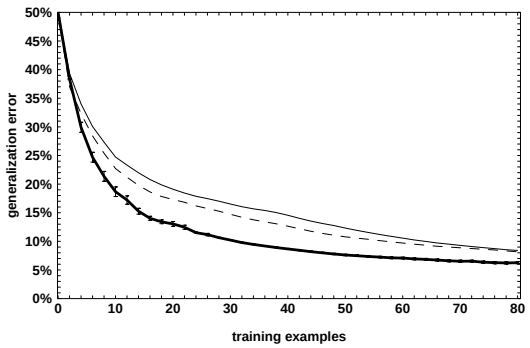


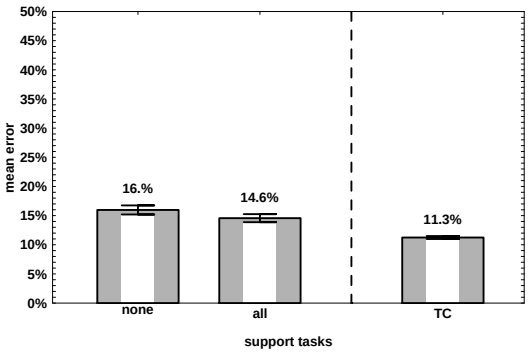
Figure 11: Selective transfer in task family $\mathcal{T}_2$ with $T = 4$ task clusters.

$\mathcal{T}_2$, all clusters

(a) generalization error
task clusters (dashed)
E-optimal metric (solid)



(b) average generalization error
individual task clusters (left)
and TC (right)



(c) sample complexity
equally weighted metric
vs. E-optimal metric

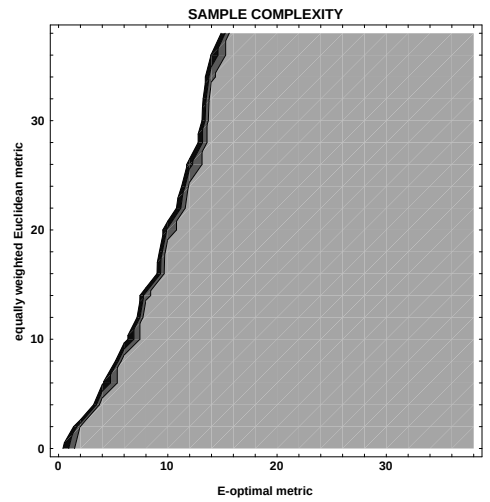
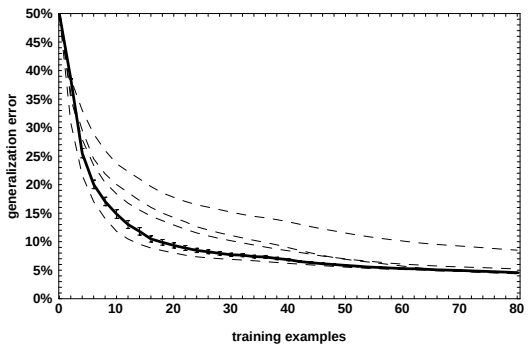


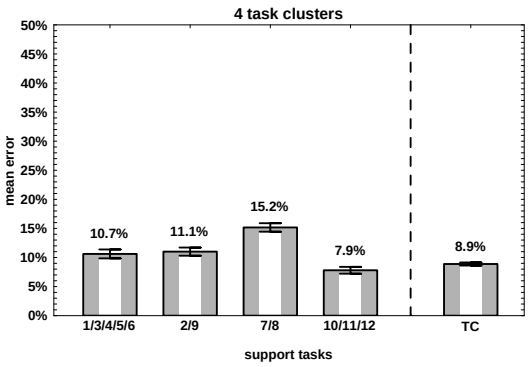
Figure 12: Selective transfer in task family $\mathcal{T}_2$ using the complete task hierarchy ($T = 1, \ldots, N$).

$\mathcal{T}_1$, T=4 **clusters**

(a) generalization error
task clusters (dashed)
E-optimal metric (solid)



(b) average generalization error
individual task clusters (left)
and TC (right)



(c) sample complexity
equally weighted metric
vs. E-optimal metric

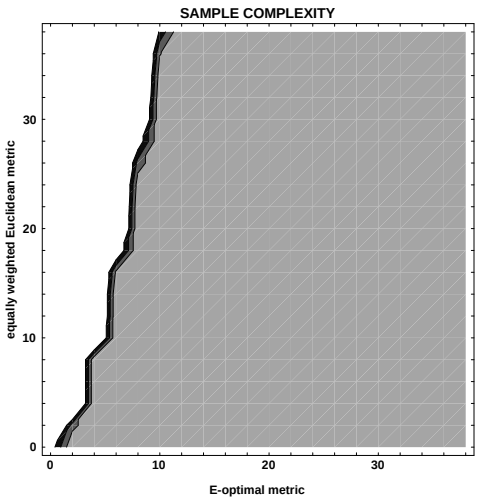


Figure 13: Selective transfer in task family $\mathcal{T}_1$ with $T = 4$ task clusters. Here selective transfer yields about the same results as unselective transfer, since most tasks are indeed related.

probability TC picks the wrong task cluster. For example, when $T = 4$ and only four training examples are given in the test task (one example of an open door, and one of a closed door), TC picks in 59.0% of our experiments the correct task cluster $\{2, 4, 5, 6\}$. In 24% of all experiments, however, TC selects task cluster $\{5''\}$, in 9% task cluster $\{2', 4', 5', 6'\}$, and in 8% task cluster $\{2'', 4'', 6''\}$. The situation changes as more training data arrives. With 20 training examples, TC correctly guesses the best task cluster in 91% of all experiments, and with 32 or more patterns it reliably (100%) identifies the best cluster. This illustrates that TC, with some error, manages to identify the most relevant tasks.

When T is unknown, TC uses the complete task hierarchy as a pool of potentially related tasks instead of a singe partitioning with a specific number of clusters T. As can be seen from Figure 12, the task hierarchy for task family $\mathcal{T}_2$ contains a total of 25 task clusters, which consists of 12 single-task clusters, one cluster that contains all tasks, and 12 other clusters containing more than one task. Results of applying TC with the complete task hierarchy are depicted in Figure 12. As to be expected, the performance if TC when using the whole task hierarchy is worse than the performance when using $T = 3$ or $T = 4$ clusters. However, the results are still clearly superior to the unselective approach, which is forced to transfer knowledge from all tasks.

All the performance results illustrate the third key result of the empirical study: Selective transfer is superior to unselective transfer in situations where many tasks are irrelevant. For example, if T=3, TC achieves 9.8% average generalization error in the test task, if knowledge is transferred selectively from the support tasks. Relatively speaking, this is only 67.1% of the average error that is being observed in the unselective approach (which is 14.6%, *cf.* right bar in Figure 6b), and it is also considerably close to the theoretically optimal value (6.2%, *cf.* right bar in Figure 5b). The relative improvement in the sample complexity is even more significant: The sample complexity in the test set is only 48.8% when TC transfers knowledge selectively, when compared to the unselective counterpart.

When TC is compared with the equally-weighted distance metric (nearest neighbor in the absence of support sets), TC with T=3 uses only 41.4% of the samples to reach the same level of generalization accuracy, and its generalization accuracy is on average on 60.9% of that inferred by the equally-weighted distance metric. These results are remarkably close to those that could have been achieved if one knew in advance which of the 12 support sets were appropriately related. From the results shown in Figures 10 and 11, it appears that the number of task clusters T only weakly impacts the results. In other experiments not shown here we found this to be the case as long as $T \geq 3$. For smaller values of T, the number of task clusters is insufficient, and TC's performance degrades to that of the regular

nearest neighbor with the equally-weighted distance metric.

For the sake of completeness, Figure 13 show the results for TC with T=4 in task family $\mathcal{T}_1$. Recall that most tasks were related in $\mathcal{T}_1$, and even the unselective transfer mechanisms produced satisfactory results. An interesting finding here is that TC identifies clusters of tasks that significantly differs in their relatedness to the testing task, and even though only a small number of tasks is used to determine the E-optimal distance metric, TC generalizes approximately as well as the unselective approach. The latter approach benefits from the fact that it can use more data to adjust the distance metric.

3.5 Practical Utility of the Results

The ability to learn to recognize people, landmarks, or locations are useful in the context of robot navigation and human robot interaction. The TC algorithm has particularly been designed to facilitate the reliable and fast recognition of such concepts. In addition to the small the sample complexity due to knowledge transfer, TC works very fast in practice, since it relies on nearest neighbor generalization. Nearest neighbor memorizes training examples, avoiding the need for long training times. For efficient retrieval, tree-based algorithms exist that facilitate fast access of the memorized data (see *e.g.*, [17]).

The particular testing task, the distinction of open and closed doors, is useful in the context of mobile robot navigation. Figure 14 shows how the XAVIER robot navigates using an occupancy map approach [19] that has originally been developed in the RHINO mobile robot project at the University of Bonn [32, 8]. Since occupancy maps assume the world is static, they cannot handle well failures that arise from the dynamics in real-world environments. A typical failure situation is shown in Figure 15. Here the door, which was previously open, is suddenly locked. XAVIER's navigation system detects this failure quite inefficiently: It moves towards the door and circles in front of the door until its internal occupancy map is corrected. In practice, this may be quite time consuming, since previous evidence of the door being open has now to be overridden by the new evidence that indicates the presence of an obstacle.

To recognize the upcoming plan failure much earlier, the navigation routines were augmented by the learned door status recognizer (along with the result of task 12 above: door/no door) [22]. Training examples for the door recognition task are easily constructed, since plan failures (the robot turns around) are easily detected post the fact. In [22], it has been shown that an artificial neural network is able to reliably (100% in more than 30 trials) detect closed doors, enabling the robot to change its path accordingly. An example is shown in Figure 16. To reduce the ratio

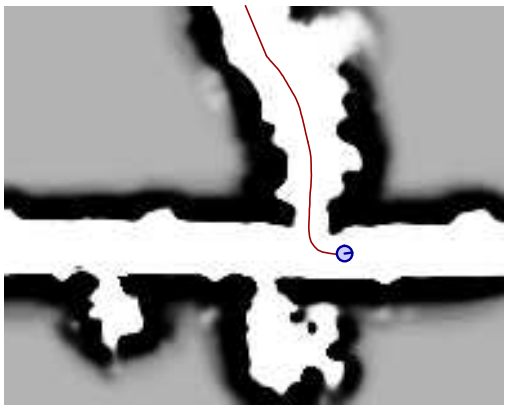


Figure 14: Example of the XAVIER robot leaving a room. The two-dimensional occupancy map (bird's eye view of the robot's environment) has been constructed previously, based on sonar sensor information. Bright values correspond to free space, and dark values to occupied regions.

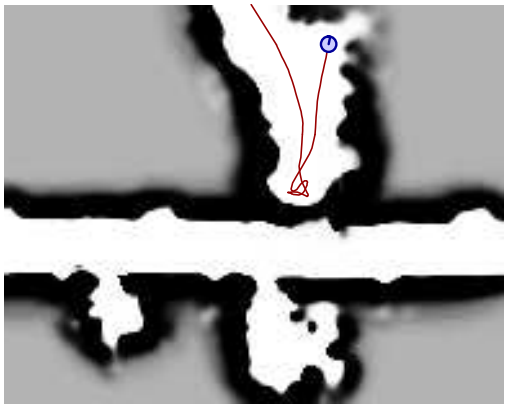


Figure 15: Example of a plan failure. XAVIER, which find part of its path blocked, has to correct its occupancy map before taking a different route. This correction is time-consuming.

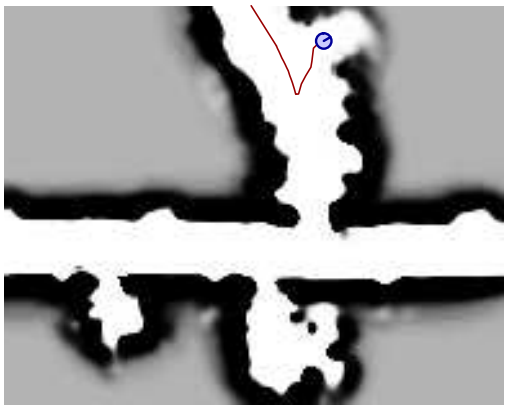


Figure 16: Failure anticipation after learning to recognize closed doors. The robot turns around much earlier.

of false alarms, the robot's motion direction was only changed if three consecutive sensor readings indicated the presence of a closed door with high confidence. The neural network approach, however, suffered from enormous training times (which also prohibited a systematic evaluation of the approach such as the one that is presented here). We expect that the TC approach will be much faster and easier to handle in practice.

4 Discussion

This paper describes an instantiation and first results of a method for the selective transfer of knowledge across multiple learning tasks. The TC algorithm employs a nearest neighbor algorithm, which transfers knowledge by adjusting the distance metric in some tasks while re-using it in others. To transfer knowledge selectively, TC clusters tasks into bins of related tasks. Relatedness is defined in the context of TC's knowledge transfer mechanisms. When facing a new learning task, TC determines the most related task cluster and selectively transfers knowledge form this one cluster only. In an experimental comparison conducted in a mobile robot perceptual domain it was shown that

1. If tasks are appropriately related, TC's transfer mechanisms successfully reduces the sample complexity. For example, in task family $\mathcal{T}_1$ TC consumes only 28.4% of the training examples that an uninformed distance metric with equal weights requires.

2. TC's clustering mechanisms manages to discover meaningful task clusters and to build hierarchies of tasks. For example, in task family $\mathcal{T}_2$ TC groups the three different task types into separate clusters, given that $T \geq 4$ clusters are available. In $\mathcal{T}_1$, tasks that involve the recognition of people always fall into different clusters than those that do not (10,11,12).

3. If tasks are not appropriately related (task family $\mathcal{T}_2$), selectively transferring knowledge from the most related task cluster improves the results significantly. For example, in task family $\mathcal{T}_2$, in which most tasks are not appropriately related to the testing task, selective transfer requires only 42.0% of the amount of training data required by the corresponding unselective transfer mechanisms.

A key assumption made in the TC approach is the existence of groups of tasks so that all tasks are related within each group. TC also benefits if the appropriate number of task clusters is known beforehand. TC is particularly appropriate when

there is a small, known number of distinct problem categories, and within each problem category TC's transfer mechanisms successfully reduces the sampling complexity. Our empirical results provide evidence that TC is somewhat robust with respect to the choice of the number of task clusters T (TC even managed to reduce the sample complexity in task family $\mathcal{T}_2$ when the complete task hierarchy was employed which contains all clusters for all possible values of T). However, little is known for cases where the class boundaries are smoother. In cases where task boundaries are smoother, smoother arbitration schemes (*e.g.*, weighting the impact of a task cluster in proportion to $r(\cdot)$) might be superior.

A second limitation of the current implementation arises from the fact that the space of all partitions is searched exhaustively. Clearly, the complexity of exhaustive search prohibits global optimization for large values of N and T. However, we do not view this as a principal limitation of the TC algorithm, since heuristic and/or stochastic optimization methods are certainly applicable. If learning tasks arrive one after another, task clusters may also be learned incrementally, by determining cluster membership when a task arrives. Little is known concerning how much the results presented here depend on the fact that the partitioning always represents the global minimum of J.

In the experiments reported here, TC has been applied to analyze images and sonar measurements. Comparing images and sonar scans pixel-by-pixel is certainly not the most effective strategy for analyzing images. This simple encoding has been chosen mainly because of our interest to rely as little as possible on prior, domain-specific knowledge. However, TC, as it is currently implemented, is unable to discover dependencies beyond the level of individual pixels, since it only learns a distance value for each input dimension. Applying the TC algorithm to more sophisticated image encodings, and applying algorithms that can discover more sophisticated commonalities between multiple tasks such as those surveyed in [34], is subject to ongoing research.

The reader may notice that the general scheme outlined in the very beginning of this paper (Table 1) may be applicable to other approaches that transfer knowledge across multiple learning tasks. Of course for some approaches this will be computationally infeasible, since this scheme requires in the order of N^2 comparisons, each involving repeated experiments with transfer across tasks. One of the main limitations of TC arises from the fact that task clustering is based on pairwise comparisons. TC will not capture effects of transfer that arise if only three or more tasks are involved. It remains to be seen whether pairwise comparisons will prevent TC from finding useful clusters in different application domains. However, a full evaluation of transfer in all subsets of tasks requires time exponentially in the number of tasks N, whereas TC time requirements are quadratic. It even appears

feasible to design incremental strategies whose time requirements are linearly in $N \cdot T$, which will be more efficient than the current implementation of TC if T is small.

The key difference of the TC approach to previous approaches lies in TC's ability to transfer knowledge selectively. Rather than weighting all previous learning tasks equally when learning bias for a new one, TC structures the space of learning tasks and reasons about their relatedness. In the light of the experimental findings, we conjecture that the TC approach scales much better to diverse application domains, *i.e.*, domains in which the learning tasks are not all just of a single type.

References

[1] Abu-Mostafa, Y. S. *A Method for Learning from Hints*. in: **Advances in Neural Information Processing Systems 5**, edited by S. J. Hanson, J. Cowan, and C. L. Giles. Morgan Kaufmann, San Mateo, CA, 1993, pp. 73–80.

[2] Ahn, W.-K. and Brewer, W. F. *Psychological Studies of Explanation-Based Learning*. in: **Investigating Explanation-Based Learning**, edited by G. DeJong. Kluwer Academic Publishers, Boston/Dordrecht/London, 1993.

[3] Atkeson, C. A. *Using Locally Weighted Regression for Robot Learning*. in: **Proceedings of the 1991 IEEE International Conference on Robotics and Automation**, edited by . Sacramento, CA, 1991, pp. 958–962.

[4] Baxter, J. *The Canonical Metric For Vector Quantization*. Flinders University, Department of Mathematics and Statistics, Australia, 1995. *Submitted for publication*.

[5] Baxter, J. *Learning Internal Representations*. in: **Proceedings of the Conference on Computation Learning Theory**, edited by . 1995, p. . *To appear*.

[6] Beymer, D., Shashua, A., and Poggio, T. *Example Based Image Analysis and Synthesis*. Massachusetts Institute of Technology, Artificial Intelligence Laboratory, November 1993. *A.I. Memo No. 1431*.

[7] Buhmann, J. *Data Clustering and Learning*. in: **Handbook of Brain Theory and Neural Networks**, edited by M. Arbib. Bradfort Books/MIT Press, 1995, pp. 278–282.

[8] Buhmann, J., Burgard, W., Cremers, A. B., Fox, D., Hofmann, T., Schneider, F., Strikos, J., and Thrun, S. *The Mobile Robot Rhino*. **AI Magazine**, vol. 16 (1995), p. .

[9] Caruana, R. *Multitask Learning: A Knowledge-Based of Source of Inductive Bias*. in: **Proceedings of the Tenth International Conference on Machine Learning**, edited by P. E. Utgoff. Morgan Kaufmann, San Mateo, CA, 1993, pp. 41–48.

[10] Franke, R. *Scattered Data Interpolation: Tests of Some Methods*. **Mathematics of Computation**, vol. 38 (1982), pp. 181–200.

[11] Friedman, J. H. *Flexible Metric Nearest Neighbor Classification*. Department of Statistics and Linear Accelerator Center, Stanford University, Stanford. CA 94305, November 1994.

[12] Hastie, T. and Tibshirani, R. *Discriminant Adaptive Nearest Neighbor Classification*. Dept. of Statistics and Biostatistics, Stanford University, Stanford, CA, December 1994. *Submitted for publication*.

[13] Hertz, J., Krogh, A., and Palmer, R. G. **Introduction to the theory of neural computation**. Addison-Wesley Pub. Co., Redwood City, California, 1991.

[14] Hild, H. and Waibel, A. *Multi-Speaker/Speaker-Independent Architectures for the Multi-State Time Delay Neural Network*. in: **Proceedings of the International Conference on Acoustics, Speech and Signal Processing**, IEEE, edited by . 1993, pp. II 255–258.

[15] Jordan, M. I. and Rumelhart, D. E. *Forward models: Supervised learning with a distal teacher*. **Cognitive Science**, vol. 16 (1992), pp. 307–354.

[16] Lando, M. and Edelman, S. *Generalizing from a single view in face recognition*. no. CS-TR 95-02, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel, January 1995.

[17] Moore, A. W. *Efficient Memory-based Learning for Robot Control*. Trinity Hall, University of Cambridge, England, 1990.

[18] Moore, A. W., Hill, D. J., and Johnson, M. P. *An Empirical Investigation of Brute Force to choose Features, Smoothers and Function Approximators*. in: **Computational Learning Theory and Natural Learning Systems, Volume 3**, edited by S. Hanson, S. Judd, and T. Petsche. MIT Press, 1992.

[19] Moravec, H. P. *Sensor Fusion in Certainty Grids for Mobile Robots*. **AI Magazine**, vol. (1988), pp. 61–74.

[20] Moses, Y., Ullman, S., and Edelman, S. *Generalization across changes in illumination and viewing position in upright and inverted faces*. no. CS-TR 93-14, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel, 1993.

[21] O'Sullivan, J., Mitchell, T. M., and Thrun, S. *Explanation-Based Neural Network Learning from Mobile Robot Perception*. in: **Symbolic Visual Learning**, edited by K. Ikeuchi and M. Veloso. Oxford University Press, 1995.

[22] O'Sullivan, J. and Thrun, S. *A Robot That Improves Its Ability To Learn*. Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, September 1995. *Submitted for publication.*

[23] Pratt, L. Y. *Transferring Previously Learned Back-Propagation Neural Networks to New Learning Tasks*. Rutgers University, Department of Computer Science, New Brunswick, NJ 08904, May 1993. *also appeared as Technical Report ML-TR-37*.

[24] Rendell, L., Seshu, R., and Tcheng, D. *Layered Concept-Learning and Dynamically-Variable Bias Management*. in: **Proceedings of IJCAI-87**. 1987, pp. 308–314.

[25] Sejnowski, T. J. and Rosenberg, C. R. *NETtalk: A Parallel Network that Learns to Read Aloud*. no. JHU/EECS-86/01, Johns Hopkins University, 1986.

[26] Sharkey, N. E. and Sharkey, A. J. C. *Adaptive Generalization and the Transfer of Knowledge*. in: **Proceedings of the Second Irish Neural Networks Conference**, edited by . Belfast, 1992, p. .

[27] Silver, D. and Mercer, R. *Toward a model of consolidation: The retention and transfer of neural net task knowledge*. in: **Proceedings of the INNS World Congress on Neural Networks**, edited by . Washington, DC, 1995, pp. 164–169, Volume III.

[28] Stanfill, C. and Waltz, D. *Towards Memory-Based Reasoning*. **Communications of the ACM**, vol. 29 (1986), pp. 1213–1228.

[29] Suddarth, S. C. and Holden, A. *Symbolic neural systems and the use of hints for developing complex systems*. **International Journal of Machine Studies**, vol. 35 (1991), p. .

[30] Sutton, R. S. *Adapting Bias by Gradient Descent: An Incremental Version of Delta-Bar-Delta*. in: **Proceeding of Tenth National Conference on Artificial Intelligence AAAI-92**, AAAI, edited by . AAAI Press/The MIT Press, Menlo Park, CA, 1992, pp. 171–176.

[31] Thrun, S. **Explanation-Based Neural Network Learning: A Lifelong Learning Approach**. Kluwer Academic Publishers, Boston, MA, 1996. *to appear*.

[32] Thrun, S. *Exploration and Model Building in Mobile Robot Domains*. in: **Proceedings of the ICNN-93**, IEEE Neural Network Council, edited by . San Francisco, CA, 1993, pp. 175–180.

[33] Thrun, S. *Is Learning the n-th Thing Any Easier Than Learning the First?* in: **Advances in Neural Information Processing Systems 8**, edited by D. Touretzky and M. Mozer. MIT Press, Cambridge, MA, 1996, p. . *to appear*.

[34] Thrun, S. *Lifelong Learning: A Case Study*. no. CMU-CS-95-208, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15213, November 1995.

[35] Utgoff, P. E. **Machine Learning of Inductive Bias**. Kluwer Academic Publishers, 1986.