UNIVERSAL TRANSFER LEARNING

BY

M. M. HASSAN MAHMUD

B.S., Stevens Institute of Technology, 2000
M.S., University of Illinois at Urbana-Champaign, 2002

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2008

Urbana, Illinois

Doctoral Committee:

Professor Gerald F. DeJong, Chair
Associate Professor Steven M. LaValle
Professor Stephen Levinson
Professor Mehdi T. Harandi

Abstract

The aim of transfer learning is to reduce sample complexity required to solve a learning task by using information gained from solving related tasks. Transfer learning has in general been motivated by the observation that when people solve problems, they almost always use information gained from solving related problems previously. Indeed, the thought of even children trying to solve problems *tabula rasa* seem absurd to us. Despite this fairly obvious observation, typical machine learning algorithms consider solving one task at a time and so do not take advantage of information that has become available from solving related tasks previously. Transfer methods aim to rectify this rather serious oversight and so have a potential to make a huge impact on how successful and widespread the use of machine learning is.

Practical methods to transfer information has been developed and applied successfully to difficult real life problems. In addition theoretical analysis of these methods have been developed. However one fundamental problem still remains unsolved, which is how one measures similarity between tasks. This problem is obviously quite troubling from a conceptual point of view, as the notion of relatedness seem central to the objective of transferring information between related tasks. Furthermore, it has been shown in experiments that transferring from 'unrelated' tasks hurts generalization performance of learning algorithms. So an appropriate notion of similarity between tasks seem necessary to design algorithms that can determine when to transfer information, when not to and how much information to transfer. In this dissertation we give a formal solution to the problem of measuring task relatedness and all its associated problems.

We derive a very general measure of relatedness between tasks. We show that this measure is *universal* – i.e. no other measure of relatedness can uncover much more similarity than our measure. We then use this measure to derive *universally optimal* transfer learning algorithms in a Bayesian setting. Universal optimality means that no other transfer learning algorithm can perform much better than ours. The methods we develop automatically solve the problems of determining when to transfer information and how much information to transfer. Indeed, we show

that transferring information is always justified – i.e. it never hurts too much to transfer information. This latter result is quite surprising indeed as the commonly held belief in the transfer learning community is that it should hurt to transfer from unrelated tasks. We also show how our transfer learning methods may be used to do transfer in Prediction with Expert Advice Systems and in Reinforcement Learning agents as well.

Our distance measures and learning algorithms are based on powerful, elegant and beautiful ideas from the field of Algorithmic Information Theory. While developing our transfer learning mechanisms we also derive results that are interesting in and of themselves. We also developed practical approximations to our formally optimal method for Bayesian decision trees, and applied it to transfer information between 7 arbitrarily chosen data-sets in the UCI machine learning repository through a battery of 144 experiments. The arbitrary choice of databases makes our experiments the most general transfer experiments to date. The experiments also bear out our result that transfer should never hurt too much.

To my father, mother and little sister.

Acknowledgments

I would like to thank all the people who have over the years provided feedback in various forms that has helped my intellectual and personal growth, which, I believe and hope, finds a manifestation through this dissertation.

I would like to Dr. Gerald DeJong, Dr. Steven LaValle, Dr. Stephen Levinson, Dr. Mehdi Harandi and Dr. Jean Ponce for being part of my thesis committee and providing valuable feedback for helping developing the thesis.

I would like to thank the various people who have over the years participated in the ANNCBT seminar and introduced me to various ideas etc. In particular I would like to thank Dr. Sylvian Ray, Dr. Thomas Anastasio and Dr. Stephen Levinson for running the seminar and always providing many very illuminating discussions and explanations of the technical and philosophical aspects of both human and artificial intelligence. Their ideas and thoughts provided very welcome depth, texture and nuance to my understanding of machine learning.

I would like to thank Dr. Gerald DeJong for making me excited about research, supervising my Master's thesis and helping me appreciate the need for precisely and formally stating our ideas and being able to defend them against penetrating and probing questions. I would also like to thank him for coming in at a crucial time near the end of my thesis and agreeing to take over when tragedy struck and Dr. Ray passed away.

I would like to express my deep gratitude to Dr. Sylvian R. Ray for advising me for most of my tenure as a Ph.D. student. Unfortunately, Professor Ray passed away on 12th December 2007, just a few months before this thesis was completed. He was the primary researcher in building the ILLIAC III, one of the first computers in the world; and not surprisingly, he was deeply interested in investigating what it would take to build one of the first truly intelligent robots, one that is capable of learning and developing itself autonomously and continually. It is largely for this reason I was able to pursue research into transfer learning. For this I am grateful, but I think I will remember him most fondly because of his humor, his kindness, his constant encouragement and the freedom he gave me to develop intellectually as I felt was best for me, while making sure I do not go too far off track.

As such, he made a permanent impact on my life, as he did with all of his students prior to me. I will miss him, and I profoundly regret that he was not able to see this completed dissertation.

I would like to thank fellow graduate students in the Biosignals Intelligence Group, Samarth Swarup, Kiran Lakkaraju, Tuna Oezer, Alex Kosukoroff and others who have over the years provided me with valuable feedback regarding this work.

I would also like to thank Ms. Barbara Cicone, Ms. Mary Beth Kelley and others at CS Department Academic Advising Office the for helping me negotiate the myriad rules and regulations that any graduate student has to contend with.

Finally I would like to thank for my parents, M. Jinnat Ali Mian and Kaniz L.L. Jinnat and my sister Zerin B. Jinnat for the constant love and support I have received from them beyond the call of duty. I would also like to thank all my family and friends back home in Bangladesh for their love and support over the years.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In Transfer Learning (TL) (Pratt, 1992; Singh, 1992; Schmidhuber, 1994; Caruana, 1993; Caruana, 1997; Thrun & Mitchell, 1995), we are concerned with reducing sample complexity required to learn a particular task by using information from solving *related tasks* – Fig. 6.8 gives a simple example of this idea (see Thrun & Pratt, 1998; Vilalta & Drissi, 2002 for reviews).

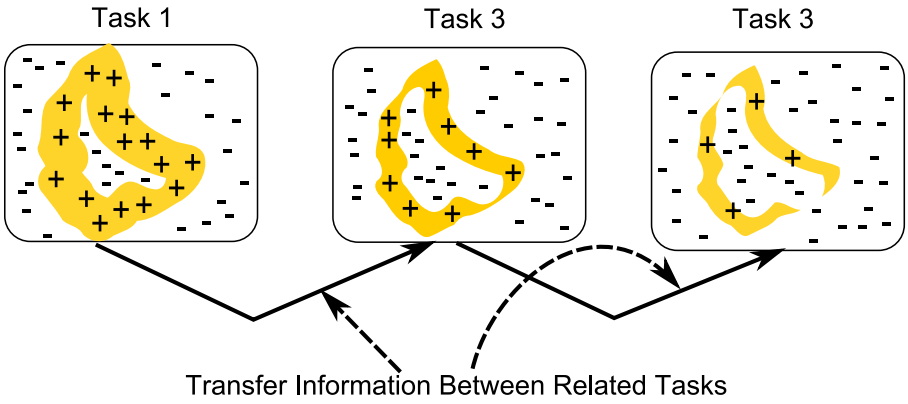

Figure 1.1: The figure shows three tasks which are related by virtue of the concepts to be learned having similar shapes.

Each task in TL corresponds to a particular probability measure generating the data for the task. Transfer learning has in general been inspired by noting that to solve a problem at hand, people almost always use knowledge from solving related problems previously. This motivation has been borne out by practical successes; TL was used to recognize related parts of a visual scene in robot navigation tasks (Caruana, 1997), predict rewards in related regions in reinforcement learning based robot navigation problems (Thrun & Mitchell, 1995), predict results of related medical tests for the same group of patients (Caruana, 1997), transfer information across relational/structured data sets (Mihalkova et al., 2007), transfer in difficult reinforcement learning problems (Taylor & Stone, 2007), and even transfer across *superficially* unrelated classification tasks (Mahmud & Ray, 2007; Mahmud, 2007). A key concept in transfer learning, then, is this notion of relatedness between tasks. As we will see, in the work preceding the contents of this disserta-

tion it was not clear what a proper way to define this notion is (see also Caruana, 1997; Ben-David & Schuller, 2003). This problem is conceptually quite troubling and has also hampered development of even more powerful and principled transfer algorithms that know how much information to transfer, when to transfer information, and when not to.

Many current TL methods are in essence based on the method developed by Caruana, 1997. The basic idea is to learn m related tasks in *parallel* using neural networks, with all the tasks defined on the same input space (Fig. 1.2). The assumption is that the different tasks are related by virtue of requiring the same set of good 'high level features' encoded in the hidden units. The goal now is to try to learn these high level features quicker by learning all the tasks at the same time by alternating the training samples from the different tasks. The same idea has been used for sequential transfer – i.e. input-to-hidden layer weights from previously learned related tasks were used to speed up learning of new tasks. So the notion of relatedness between tasks is 'functional' in nature – tasks are considered related if they can be learned faster together than individually, or in other words, if they have a *common near-optimal inductive bias* with respect to a given hypothesis space (e.g. the common hidden units in Fig. 1.2).

This case was analyzed extensively in a PAC setting by Baxter, 2000. Here a probability distribution P was assumed over the space of tasks, and bounds were derived on the sample complexity required to estimate the expected error (with respect to P) of the m tasks when the tasks were learned using a sub-space of the hypothesis space. That is bounds were derived for sample complexity for estimating fitness of inductive biases. Most work done on TL is subsumed by this analysis, and they all begin with the assumption that tasks have a common, near optimal inductive bias. So no actual measure of similarity between tasks is prescribed, and hence it becomes difficult to understand, let alone answer, questions such as 'how and when should we transfer information between tasks ?' and 'how much information should we transfer ?'[1].

Many attempts have been made to solve this problem in practice and, while quite effective in application domains considered, they are, unfortunately, ad-hoc in nature. There has been two major efforts to give a theoretical underpinning to this problem and we now briefly describe these methods and how they relate to our theory.

Ben-David & Schuller, 2003 give a more explicit measure of relatedness in which two tasks P and Q are said to be similar with respect to a given set of functions F if $\exists f \in F$ such that $P(a) = Q(f(a))$ for all events (i.e. measurable

[1]Indeed, the discussions in the Neural Information Processing Systems 2005 Workshop on Inductive Transfer (i.e. Transfer Learning) was largely focused on trying to answer this very question.

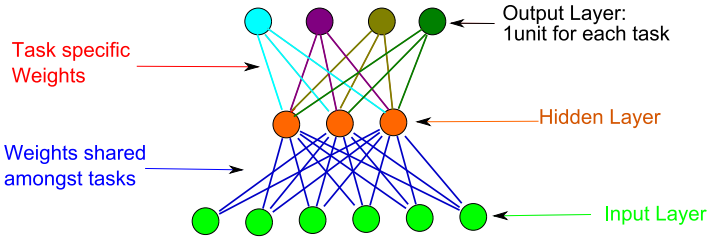

Figure 1.2: A typical Transfer Learning Method.

sets) a. Using F, the authors derive PAC sample complexity bounds for the error of each task (as opposed to expected error in Baxter, 2000), which can be smaller than single task bounds under certain conditions. So the measure of similarity used is *binary* in that the tasks are either related or they are not. So this does not help solve the problems of measuring *how much* information to transfer and so forth. And while the authors have presented applications where this approach applies (Ben-David et al., 2002), due to the dependence on an a-priori known space F, which needs to satisfy the stringent constraint, it is unclear just how general this approach is.

More interesting is the work by Juba, 2006 which extends Baxter, 2000. The paper deals with finite sample spaces, and computable tasks and hypothesis spaces, and gives PAC bounds, where the sample complexity required to bound the expected error is proportional to the *joint Kolmogorov complexity* of the m hypothesis being considered. The number of tasks required for the bounds to hold is ≥ 8192 (Theorem 3). Use of joint Kolmogorov complexity to measure relatedness is a step in the right direction as it measures how well the tasks compress together and hence the *total* absolute information content of the m tasks considered as a whole (see below). However what we actually want is the amount of information asks contain *about* each other, and for this we need to use the *conditional Kolmogorov complexity* and the *Information Distance* (see below). Indeed, this is basic idea that we explore and make concrete in this dissertation.

Let us take a brief look at our approach, which is essentially transfer learning in the setting of Solomonoff Induction (Solomonoff 1964a; 1964b; 1978) (Hutter, 2003). Recalling that each task corresponds to the probability measure generating the samples for that task, we assume that each hypothesis/probability measure is represented by a program – for example a decision tree is represented by a program that contains a data structure representing the tree, and the relevant code to compute the leaf node corresponding to a given input vector. The Kolmogorov complexity of a hypothesis $\mathbf{h}$ (or any other bit string) is now defined as the length of the shortest program that outputs $\mathbf{h}$ given no input. This is a measure of absolute information content of an individual object – in this case the hypothesis $\mathbf{h}$. It can be

shown that Kolmogorov complexity is a sharper version of Information Theoretic entropy, which measures the amount of information in an *ensemble of objects* with respect to a *distribution* over the ensemble. The conditional Kolmogorov complexity of hypothesis $\mathbf{h}$ given $\mathbf{h}'$, $K(\mathbf{h}|\mathbf{h}')$, is defined as the length of the shortest program that outputs the program $\mathbf{h}$ given $\mathbf{h}'$ as input. $K(\mathbf{h}|\mathbf{h}')$ measures amount of *constructive* information $\mathbf{h}'$ contains about $\mathbf{h}$ – how much information $\mathbf{h}'$ contains for the purpose of constructing $\mathbf{h}$. This is precisely what we wish to measure in transfer learning. Hence this becomes our measure of relatedness for performing sequential transfer learning in the Bayesian setting.

In the Bayesian setting, any sequential transfer learning mechanism/algorithm is 'just' a conditional prior $W(\cdot|\mathbf{h}')$ over the hypothesis/probability measure space, where $\mathbf{h}'$ is the task learned previously – i.e. the task we are trying to transfer information from. In this case, by setting the prior over the hypothesis space to be $P(.|\mathbf{h}') := 2^{-K(\cdot|\mathbf{h}')}$ we weight each candidate hypothesis by how related it is to previous task(s), and so we automatically transfer the right amount of information when learning the new problem. We show that in a certain precise sense this prior is never much worse than any *reasonable* transfer learning prior, or any non-transfer prior. So, sequential transfer learning is always justified from a theoretical perspective. This result is quite unexpected as the current belief in the transfer learning community that it should hurt to transfer from unrelated tasks. We show that similar results hold for the *correct* interpretation of parallel transfer learning, while current parallel transfer methods, used in practice, are in fact sequential transfer methods in disguise.

Kolmogorov complexity is computable only in the limit, that is with infinite time and resource. Hence our approach gives a transfer method that is only theoretically/formally optimal. At first blush, this might seem to reduce its importance for those who are interested in practical transfer. But this is not true, as what this method actually does is give us a 'gold standard' that transfer learning methods should be trying to achieve. This assertion is borne out by the fact that by approximating this method we were able to construct the most general possible transfer experiments to date. See also Cilibrasi & Vitanyi, 2005 for an impressive demonstration of the power of Kolmogorov complexity approximation based methods for difficult clustering problems.

We also note here that since we use a previously learned hypothesis as prior knowledge, and since we represent each hypothesis as simply a bit string without looking at its properties as a program, the prior knowledge being used can be any arbitrary bit string b at all. Hence the corresponding set of prior knowledge based schemes we get are $W(.|b)$ and all the optimality results for our sequential transfer holds – i.e. $2^{-K(.|.)}$ is the universally optimal Bayesian prior for arbitrary prior

knowledge based methods.

Before proceeding further, let us briefly return to the issue with Juba's approach where the joint Kolmogorov complexity was used to measure task relatedness. In the example above, this would be given by $K(h, h')$, which is the length of the shortest program that outputs $\mathbf{h}$ and $\mathbf{h}'$ in sequence. So in essence this measures the amount of information contained in both tasks $\mathbf{h}$ and $\mathbf{h}'$ together, whereas, as explicated above, what we require is the amount of information the tasks $\mathbf{h}$ and $\mathbf{h}'$ contain about *each* other. And for this reason use of joint $K()$ is inappropriate.

Our exposition in this dissertation takes the following course. In Chap. 2 we categorize and discuss various transfer methods that have been developed so far. We discuss their strengths and weaknesses and contrast it with our method in general terms. In particular we focus on how these methods measure relatedness between tasks and transfer information between tasks, and how they may be improved upon.

In Chap. 3 we introduce our measure of task relatedness. We start by describing some fundamental notions we need and learning framework we consider. Then we introduce notions from Algorithmic Information Theory that we use and extend to derive our measure of relatedness. We use and extend the theory of Information Distance (Bennett et al., 1998) to measure relatedness between tasks, transfer the right amount of information etc. For our task space we restrict ourselves to probability measures that are lower semi-computable, which is reasonable as it covers all situations where we can learn using computers. In this space the Information Distance is a universally optimal measure of relatedness between tasks. We give a sharp characterization of Information Distance by showing it is, upto a constant, equal to the Cognitive Distance (Theorems 3.3 and 3.4, which are quite interesting results in and of themselves).

Based on our transfer learning distance, in Chap. 4 we develop universally optimal Bayesian transfer learning methods for doing sequential transfer (Theorem 4.3). We show that sequential transfer is always justified from a formal perspective (Theorem 4.4). We also investigate parallel or multitask learning and show that while universally optimal methods exist for current *interpretation* of multitask learning schemes (Theorem 4.5), which we term joint-parallel transfer, it is just single task learning in a product space. We also show that transfer algorithms currently used in practice are just sequential transfer methods in disguise (Sect. 4.3.3). We also derive a different interpretation of parallel transfer we term online-parallel transfer and a universally optimal scheme for this interpretation (Theorem 4.6). We show that this scheme can be said to be performing actual transfer and is always justified like sequential transfer (Theorem 4.7). Finally, we show that our methods are also optimal with respect to other methods in a sense stronger than the classical

Universal sense. That is, it is a powerful base method (due to its universal optimality) that can be used any time, and can also be used to improve the performance of any other transfer method that we may feel more appropriate in a given situation.

We further extend the theory developed in Chaps. 3 and 4 in Chap. 5. We extend the universal optimality results of the Bayesian transfer methods to the case of arbitrary bounded loss function and the artificial agent setting via results in Hutter 2003; 2004. We also show how our universal distance measures may be used to construct universally optimal transfer method for the Prediction with Experts Advice setting (Littlestone & Warmuth, 1987; Vovk, 1990), in particular in the methods described in Vovk, 2001. Finally we briefly investigate Kolmogorov complexity of functions and show that under certain natural restrictions on the computability of this quantity it is, upto an additive constant, equal to the Kolmogorov complexity of bit strings (Lemma 5.2).

Finally, in Chap. 6, we apply an approximation our method to transfer learning in Bayesian decision trees. We were successfully able to transfer information between 7 databases from the UCI machine learning repository (Newman et al., 1998). At the time we performed the experiments, our experiments were the most general transfer experiments, in the sense that we were able to transfer information between databases that have little or no semantic relationship to each other. An equally interesting aspect of our result is that in our experiments transferring never hurt, which also confirm our theoretical result that sequential transfer learning is always justified. We performed a total of 144 individual transfer experiments.

For us, a most interesting aspect of this work is how beautifully and naturally AIT formally solves problems in transfer learning that has been vexing researchers for a long time. We hope the work done here will encourage machine learning practitioners to look to AIT for inspiration and perhaps for solutions to difficult fundamental problems that are unyielding in the face of more traditional approaches.

Chapter 2

Previous Work

We gave an introduction to transfer learning in the previous chapter and here we will go into various transfer methods developed so far in greater depth[1].

The bulk of transfer methods developed to date can be divided into one of two distinct categories. The first is intra-domain transfer, where the primary focus is on transferring information between tasks defined on the same input-output space, and the second is cross-domain transfer where the focus is on transfer between tasks defined over different input and output spaces. The latter, very recent in origin, is in fact the current phase of transfer learning research, and as such, is a continuation of intra-domain transfer methods. However, to be able to transfer across domains the cross-domain transfer methods need to explicitly measure and use similarity between tasks to transfer information. Indeed, this explicit attempt to measure and exploit task similarity is the key property that distinguishes intra-domain transfer and cross-domain transfer.

Another strand of transfer learning research is that of programmatic transfer methods, which has been developed largely by Jurgen Schmidhuber and colleagues. The main distinguishing feature of these methods is that the hypothesis space considered are programs, and learning is performed by stochastically searching through program space. When learning a particular task, the search is biased by beginning with a program that had solved previous tasks.

Given the above, our work can be seen as a merging of cross-domain transfer and programmatic transfer methods. We provide a theoretical foundation for cross-domain transfer in the Bayesian setting and we do so by considering as our hypothesis space the most general space we will need – the set of computable probability measures.

In the following we will first discuss in succession intra-domain, cross domain, and programmatic transfer methods. In each case we will pay particular attention to exactly how task similarity is measured (implicitly or explicitly) and how this

[1] It is also interesting to note the review paper Vilalta & Drissi, 2001, where the authors discuss how transfer learning methods relate to other *meta-learning* methods. The term Meta-learning is used to refer to any method that dynamically learns the bias space, but not necessarily for multi-task learning - e.g. boosting (Schapire, 1997).

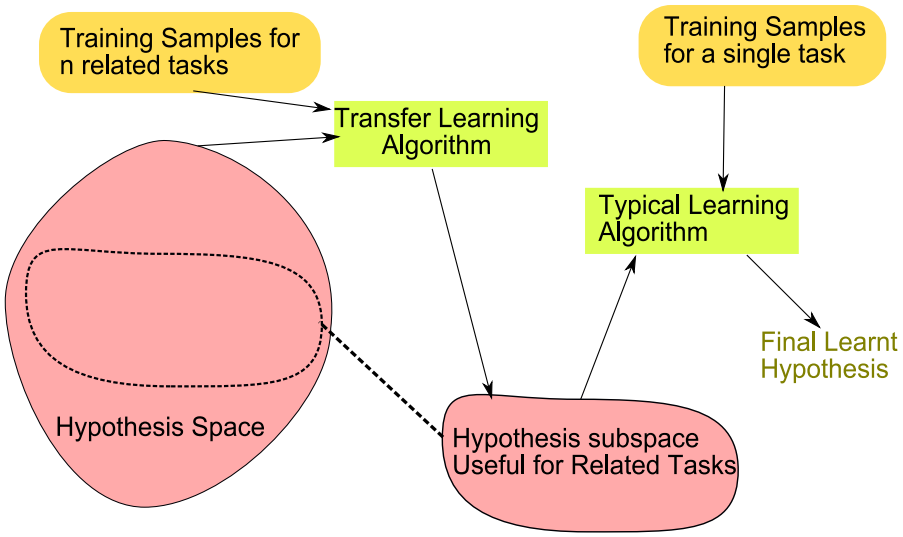

Figure 2.1: Schematic Illustration of typical Transfer Learning Methods.

measure is used to perform transfer. We will then end this chapter with a discussion of the relationship our methods to these methods.

2.1 Intra-Domain Transfer Methods

The fundamental idea behind most current intra-domain methods is exactly the same, which is that tasks have a near optimal common-inductive bias – i.e. in the hypothesis space being used by the algorithm there is a subspace that allows for faster learning of the tasks together. During transfer learning, this subspace is learned by using the tasks to determine which subspace is best for faster generalization for the related tasks. For instance, in the example from Caruana, 1997 in the Introduction, the learned weights from the input to the hidden layer correspond to the shared subspace that is learned from the related tasks. This basic idea is illustrated schematically in Fig. 2.1.

In this section we will look at intra-domain transfer methods developed for classification and for artificial agent setting. Then we will look at the theoretical framework for these methods developed so far and end with some final observations on intra-domain transfer algorithms.

2.1.1 Intra-Domain Transfer in Classification

One of the earliest studies of transfer learning in machine learning was done in Pratt, 1991. The author showed that, when learning using neural networks, sharing hidden units across related tasks is likely to improve performance (lower generalization error from the same number of training samples). This follows from the

fact alluded to earlier, that hidden layer units encode high level features that may be useful across related tasks. The authors demonstrated this by solving a single task problem faster by presetting weights in the network learning the task. The presets to be used were obtained by training smaller networks on the same task. Interestingly, the total time taken in this combine-smaller-networks approach was actually smaller than in when a single monolithic network was used.

Pratt, 1992 used a more sophisticated method to actually transfer information across a set of related tasks – e.g. transfer from detection of vowels uttered by females to detection of vowels uttered by males, transfer of diagnosis of heart problems from Californians to Swiss patients etc. The improvement achieved both in terms of generalization error and training time required were significant. In this paper, the authors used the Mutual Information between the hyper-surface defined by hidden units learned in a previous task and samples in the new task (as in decision tree learning (Breiman et al., 1993)) as a heuristic to determine which hidden layer weights from the previous task to use in the new task. Thus, this way transfer is achieved.

Mitchell & Thrun, 1993; Thrun, 1995; Thrun & Mitchell, 1995 developed the Explanation Based Neural Network method to transfer information across difficult real world problems. The idea is again to train a neural network for a particular task quicker by using information from related tasks. In this case, examples from the previous tasks are used to construct a function that computes an estimate of the the derivative of the task/function being learned at each sample point. That is, using the examples in the previous tasks, a function g is learned, such that for each sample point x, $g(x)$ is an estimate of the $\frac{df}{dx}$, where f is the function being learned for the current task. This gradient information is used to speed up convergence by using the TangentProp algorithm (Simard et al., 1992). The TangentProp algorithm is an extension of the famous BackProp algorithm for training neural networks that uses the gradient information at each point to converge faster. The assumption in EBNN is that the gradient functions of the tasks are close to each other, which is of course a heuristic. In classification, the EBNN was used to solve object recognition problems. Here each task corresponded to recognizing a particular object, given images of the object from various poses and lighting conditions. The EBNN was able to transfer the information about the transformations necessary to account for the changes in pose etc. EBNN based classification was also used to predict the next state for a given action in reinforcement learning problems. In this case each task corresponded to a particular room/environment.

In Thrun & O'Sullivan, 1996 the authors developed a transfer learning mechanism that uses the Nearest Neighbor algorithm as the underlying algorithm. In this case, task relatedness is identified as the degree to which the Nearest Neighbor

distance metric learned for one task is useful in another task. The set of tasks seen so far are clustered into groups where within each group the tasks are optimally related i.e. the tasks are clustered so that the distance metric learned using examples from all the tasks has minimal error over all the tasks. Given a new task, the the distance metric from the most related cluster is used to perform Nearest Neighbor classification for the new task.

The papers Caruana 1993; 1997 describe the Multitask Learning method that we mentioned in the Introduction. Here multiple tasks are learned in parallel and transfer between tasks occurs by virtue of the tasks having common high level features, as described in the Introduction.

Silver and Mercer 1996; 2001 described the ηMTL method to augment the MTL system in the previous paragraph. Here, the aim was to learn a particular task, while using the previous related tasks as 'hint' tasks (Abu-Mostafa, 1995) to speed up learning. Transfer from the hint tasks to the target task was controlled by heuristically measuring relatedness between tasks, and then using the measure to set the learning rate for each hint task. The more related a hint task is to the target task, the greater its learning rate is and the more influence it has on training the shared hidden layer units. This way, transfer is accomplished from task to task. The most interesting heuristic used was of mutual information between hidden layer units activations and target values for each task for each sample. If the mutual information was high, then it means that the contribution for a particular task to learning the shared weights are high and so the task is related.

Silver & McCracken, 2002; Silver & McCracken, 2003 and Silver and Poirier 2004; 2005 developed the same ideas to handle task consolidation via task rehearsal for Lifelong Learning. That is, these papers addressed the problems of how a lifelong learning/transfer learning agent may consolidate and retain knowledge gathered during its task. The solution the authors proposed was to use the ηMTL network collect all the knowledge. Problems of catastrophic interference was avoided by rehearsal using virtual examples generated by the learned network. When new tasks arrived, the heuristics mentioned above were applied to the ηMTL network to determine which previous tasks are most related to the current task, and these are used to learn the new task faster.

In Jebara, 2004, the author implemented multi-task learning for Support Vector Machines. In this case, the kernel function to be used for learning was considered to be a convex combination of a set of base kernels, where the weights were learned during training. The author further modified the SVM objective function so that the subset of features to used were also determined during the learning phase. In the multi-task learning setting, the subset of features and weights for kernels are learned so that they are good for all tasks simultaneously. The amount by which

feature and kernel weight selection in one task is affected by feature and kernel selection in another task is determined by a parameter that measures task relatedness, and it is set by the user. Hence, the idea was that by using information from related tasks, one should be able to learn better weights for the base kernels and features.

Multi-task learning was also implemented in Regularized Networks (Evgeniou et al., 2000) (a generalization of SVMs) by Evgeniou et al., 2005 and Evgeniou et al., 2004. The idea in these works was to convert the multi-task learning problem into a single task learning problem by constructing an objective function that minimizes error over all the tasks simultaneously. Again, while the mathematics is different from the methods used above, because of the framework being used, the basic idea is still the assumption that the tasks share a common inductive bias.

Another very interesting work is Ando & Zhang, 2005 where the authors study multitask learning using linear classifiers. The setup is very similar to the methods described in the preceding paragraph, and the authors assume that the tasks share a common *structural parameter* θ that determines their relatedness. A key difference from regularized network based transfer learning is that the weights for each task are partitioned into two disjoint sets. Weights in one partition v determine contribution of θ to the separating hyperplane, while the weights in the other partition u are task specific. The authors then propose an iterative algorithm to optimize (θ, v) and u in alternate steps given the value of the other u and (θ, v) respectively. Another interesting contribution of the paper is the heuristic of using of unlabeled data to generate related tasks. Such tasks are generated by using one feature the class label to be predicted and the remaining as predictors. The efficacy of this algorithm was then established via experiments using text databases. This paper also begin with the same assumption as other intra-domain transfer methods, i.e. tasks are related via a shared parameter θ. However since the weights are divided into shared and not-shared portion, one may expect it to prevent unwanted transfer. However, this is not true as the value of θ learned is only locally optimal and hence may not capture the fact that tasks are unrelated. Furthermore, since θ is optimized for all tasks simultaneously, the value of θ that might be useful for transfer between certain subsets of tasks is also not learned.

2.1.2 Intra-Domain Transfer in Agents

Most research in transfer learning has been targeted at implementing transfer in classification algorithms. The reason is partly because solving the transfer problem in agent systems boils down to solving it in classification problems (see, for example, the EBNN algorithm mentioned above). In both cases the aim is to learn a distribution faster given other related distributions learned previously. In the case

of artificial agent systems the distributions are over the next state or observation the agent makes and the reward it obtains given its history (the sequence of action-observations that the agent has seen so far).

An example of this is Wilson et al., 2007, where the authors consider Multi-task Bayesian Reinforcement Learning using a hierarchical Bayesian approach. In essence, the idea is to assume that the MDPs describing the tasks are generated according to some generative process (hence the term *hierarchical* in the name of the approach). When learning a task, tasks encountered previously are used to induce a prior distribution over the parameters for the generative process, and then samples from the current task is used to induce a posterior distribution over MDP parameters for the current task. This posterior is then used in the usual way to perform Bayesian reinforcement learning (Sterns, 2000). The authors are able to show improvement of performance in some proof of concept problems.

However, there are some exceptions to the above, and we discuss them now. The following three agent based transfer methods perform transfer using the notion of subtasks. Use of this mechanism is missing from the classification based transfer systems discussed above, but it is obviously vitally important to investigate as people use it to do transfer all the time.

The first method (Singh, 1992; Barto et al., 1995; McGovern, 2002; Singh et al., 2004a; Singh et al., 2004b), is involved with learning temporally extended actions (called options) or *skills* that the agent can reuse across different tasks. For example, TURN-ON-THE-LIGHT-SWITCH is a skill that a robotic agent may use across different tasks. Essentially, the agent learns to solve a subtask which is present in different domains. A major hurdle for this methods is to determine what exactly constitutes a subtask. The authors suggest using the advent of a salient event in the world (a light turning on for example) as a way to determine what subtasks should be.

The second subtask based method we discuss was described in Drummond, 2002. In this method the author considered reinforcement learning agents in MDPs and proposed to identify subtasks 'automatically'. The learning algorithm analyzes the shape of the value function to determine subtasks. The value function is the function learned by an artificial agent which is defined on the state space, and for each state it gives the value of that state. This is all the agent needs to determine which action take at each state. Image analysis methods are applied the value function to determine which parts of it stand out visually and these parts are identified as subtasks. When solving a new task, given a rough shape of the value function learned after some exploration, the subtasks are searched to see which ones might apply at a particular part of the value function, and then that subtask is 'stitched' into the current estimate of the value function. By composing subtasks like this, a

solution to the new task is obtained. Hence, by using subtasks, the agent is able to solve the new task quite well after a little exploration (of course, only if the visual-based heuristic holds).

In Mehta et al., 2005 the authors also used the notion of subtasks. It was assumed that the policies for the set of related tasks to be solved are combinations of some base set of policies. The only thing that differs from task to task is the weight assigned to each base policy. The authors assumed that this weight information for each task is given to the agent, and it simply learns the base policies during its exploration. This way, it is able to learn to solve the related tasks faster.

Transfer Learning methods have also been applied extensively in certain Cognitive Architectures - that is comprehensive learning systems that are aimed toward replicating human level cognitive abilities, either for modeling human behavior or for controlling robots, intelligent agents etc. Here we consider some representative set of architectures such as Soar (Newell, 1990), Prodigy (Veloso et al., 1995), Icarus (Langley & Rogers, 2004) etc. All of these use some type of symbolic language (such as FOPL, or the STRIPS language) to encode knowledge of the agent. This knowledge describes the entities that exist in the world (e.g. DRILL-BITS in a robot drill press application) and the known effect of the agent's actions on these entities (e.g. APPLY-DRILL causes HOLE-IN-METAL-PLATE). In general, these architectures implement sophisticated extensions to classical planning, which learn to improve planning performance from experience. The actual methods employed vary from architecture to architecture, but they usually take the form of learning macro-actions (Russell & Norvig, 2003). For example, Prodigy uses Analogical Learning to determine what sequence of actions will be useful in a particular task, using knowledge about solutions/sequences of actions used in similar tasks.

2.1.3 Theoretical Framework

We have already discussed existing theoretical frameworks in the Introduction in all the detail we feel is necessary. So the contents of this section will be somewhat repetitive, but is included for completeness.

The major theoretical work done for transfer learning are Baxter 1995; 1998; 2000 Ben-David & Schuller, 2003; Juba, 2006. Baxter considers the following transfer learning framework. The fundamental assumption made is that tasks are drawn from the task space according to some distribution P. Now M different tasks are drawn from this space according to P and then the problem is to choose choose a hypothesis space, from a given set of hypothesis spaces, that minimizes the expected error for new tasks drawn according to P. In this framework the author derives PAC sample complexity bounds for the expected error of a particular

hypothesis space. All the intra-domain transfer methods discussed above fall under this framework. For instance, for the neural network learning case in Caruana, 1997, given n hidden layer units, each possible hypothesis space corresponds to a particular value for the input-to-hidden layer weights. Given such a hypothesis space, each possible hypothesis in this space corresponds to a particular assignment of values to the hidden to output layer weights.

Hence, Baxter 1995; 1998; 2000, and consequently in all the intra-domain transfer methods, the similarity between tasks is assumed to be in the form of common inductive bias, that is a subset of the given hypothesis space which is helpful in quicker generalization. Therefore this similarity measure is largely dependent on the algorithm being used. The similarity between hypotheses in a particular subspace is measured by the capacity of the subspace. The capacity of the subspace H for a given real ϵ is given by the size of the smallest subset B of H such that for each $h \in H$, there is a $b \in B$ with the absolute difference between the expected loss incurred by h and and the expected loss incurred by b is less than ϵ. This is a measure of 'richness' of the subspace, i.e. how many 'different' hypotheses are there in the space.

More interesting is the approach by Juba, 2006 who gives PAC bounds in the setting of Baxter, 2000, where the sample complexity is proportional to the joint Kolmogorov complexity of the m hypotheses. The joint Kolmogorov complexity measures how well the programs computing the hypothesis, when interpreted as bit strings, compress together than individually. So the Kolmogorov complexity is the measure of relatedness. However, the bounds hold only for ≥ 8192 tasks (Theorem 3), and as we establish in this work, the more appropriate measure of relatedness is the conditional Kolmogorov complexity and the Information Distance (Bennett et al., 1998).

In Ben-David & Schuller, 2003 on the other hand, the authors measure similarity in terms of the distributions that correspond to the tasks themselves. The authors defines two measures P and Q to be $\mathcal{F}$ similar if, in a given set of functions $\mathcal{F}$, there is a function f that maps between events that have the same probability under P and Q - that is $P(A) = Q(f(A))$. Using this measure of similarity, the authors are able to bound the sample complexity for generalization error of each task as opposed to expected generalization error.

2.1.4 Intra-Domain Transfer: Coda

We hope what becomes clear from the above discussion is that, as clever and as practically effective intra-domain transfer methods are, they are heavily reliant on the assumption that tasks are related functionally. That is, it is better to learn the

tasks together than separately. However, as pointed out in Caruana, 1997, transfer methods can significantly degrade classification accuracy if this a-priori assumption does not hold. Therefore, to make transfer algorithms more broadly applicable, it is imperative that we derive a general measure of task relatedness to develop transfer methods that know how much information to transfer, when to transfer information and when not to. This has been a major focus of research in transfer learning, and cross-domain transfer methods try to address this problem in a principled way. As we show in the rest of this dissertation, our work in this thesis gives a constructive theoretical foundation for cross-domain transfer in a Bayesian setting, and hence gives a completion of the programme for transfer learning research outlined in Caruana, 1997.

2.2 Cross Domain Transfer

A recent strand in transfer learning research is the so-called cross domain transfer method (Swarup & Ray, 2006; Mihalkova et al., 2007; Taylor & Stone, 2007). In this, the goal is to transfer across tasks that are in different domains, defined over different input, output and hypothesis spaces. Current methods handle these problems by assuming the existence of some kind of 'structural similarity' between hypothesis from different spaces that measure the amount of transformation necessary to convert a hypothesis in one space to another hypothesis in a different space. This structural similarity is then used as the measure of similarity between tasks. We now discuss the exact form this idea takes in the papers mentioned above, and how this measure is used to effect transfer.

In Swarup & Ray, 2006, the authors consider a proof-of-concept problem domain where they learn a sequence of boolean functions using sparse neural networks (Utgoff & Stracuzzi, 2002). Information between tasks is transferred by finding common substructures across neural networks learned in the previous tasks. These substructures are discovered using standard graph-mining algorithms from data-mining literature. Each task is learned using a genetic algorithm (Mitchell, 1996), and when learning a new task, these common substructures are used as *primitives* when constructing candidates for the new generation. Hence the search is biased toward networks that contain sub-structures that were found to be common across previous tasks. Our practical approximation to our theory in Chap. 6 in fact uses a similar idea, but for decision trees in a Bayesian setting.

In Mihalkova et al., 2007, the authors transfer information between structured datasets which are learned using Markov logic networks (Richardson & Domingos, 2002). In this case transfer is performed by mapping between compatible predicates learned in one task to a new task. Predicates are compatible if they have the

same arity and the types of their argument agree with type-constraints induced by previous mappings (if any). The authors were successfully able to transfer between real world databases.

In Taylor & Stone, 2007, the authors transfer information between the complex reinforcement learning problems of Keepaway, Ringworld and Knight Joust. In this case transfer is achieved by defining transform functions that translates between source and target states and actions, and hence policies learned in an old task can be used to speed up learning of policies in the new tasks. These transform functions were constructed via specific, known prior knowledge about the problems being considered.

The most interesting thing about cross-domain transfer methods is that these methods actually try to measure the relatedness between tasks in a principled way and use it to determine how to transfer information. One way to view this thesis is as an affirmation that this is more or less the correct approach and that theoretically there is in fact an optimal method for measuring relatedness and transferring information.

2.3 Programmatic Transfer Methods

In this section we will discuss two other transfer methods that are sufficiently unique in their approach that we believe they deserve their own section. These methods are the Optimal Ordered Problem Solver (Schmidhuber, 2004) and the Gödel Machine (Schmidhuber, 2006). In both cases the learning algorithm searches through program space to find solution to a given task, and uses previously found programs, that solve previous tasks, to guide the search for the new task. This approach is interesting because when learning with computers the only hypotheses we can consider are ones that are computable (i.e. has representation as programs). In this respect, these methods are similar to Levin Search (Levin, 1973), and Hutter Search (Hutter, 2002) – but the key difference is that these methods use previous tasks to speed up search. We will describe each in turn.

OOPS solves a sequence of problems, where a problem is defined by a recursive function f_r that given a problem instance x and a solution instance y, outputs 1 if y is a solution to x and otherwise outputs 0. The goal now is to find a program that given problem instances outputs solution instances. In OOPS, one assumes a prior P over the set of all programs. When learning the i^{th} task, the learner spends half the time trying to use the program $p_{<i}$ learned so far, that solves *all* the $i-1$ previous tasks, to solve the i^{th} task and uses the other half of the time trying to construct a new program to solve the i^{th} task only. This learning system is bias optimal – that is OOPS will find the correct program q in time proportional to $P(q)$.

As an example application of OOPS, when solving classification problems, the set of allowed programs may be those computing particular type of hypothesis (such as decision trees). In this case, starting with a null hypothesis that has the highest possible error on the training sample, each 'task' would correspond to finding a hypothesis that is better than the previous one. So f_r in this case would output 1 if a found hypothesis is better than the previous one and output 0 otherwise.

In the Gödel machine, the goal is to construct an optimal reinforcement learning agent operating in some domain. The machine starts off with some generalized problem solver (such as OOPS) as the current learner, then at each step it uses the current learner to learn the value function. What makes the Gödel machine unique is that it also tries find a modification to the current learner that is *provably* optimal modification to the current problem solver (including the prover), where the proof is in some appropriate formal system. The Gödel machine is optimal in the sense that it tries to improve itself by finding provably optimal modifications, which is, of course, any computable learner can do.

The basic ideas and the optimality proofs in each of these methods are quite straightforward, but nonetheless quite interesting as they try to solve the problems by searching directly through the space of all programs. However both suffer from implementational issues. The main challenge in the case of OOPS is that to get it to work for some problem domain, one needs to spend significant effort constructing an instruction set that is useful for solving the problem at hand – without such prior knowledge, the machine may take too long. This is equivalent to selecting features for a particular problem, however the difference is that features are much easier to specify in the traditional learning setup as more often than not they are given in the problem definition itself, while construction of an appropriate instruction set is likely to be more time consuming and difficult. So this is justifiable for very difficult problems, as in the tower of Hanoi problem for large n in Schmidhuber, 2004. This is more difficult to justify for general inference problems where learning algorithms are meant to deal with diverse domains. The problem with the Gödel machine is that it requires implementation of an automated theorem prover suitable for the problems that the Gödel machine is using and it requires constructing the right proofs to operate successfully. Implementing appropriate theorem provers may not be that difficult, given that many such softwares already exist, but finding the right proof quickly requires appropriate heuristics which are likely difficult to construct for the the problems the Gödel machine is intended for. However, if these challenges are solved these methods will be quite formidable indeed.

2.4 Discussion

All current transfer methods begin with a particular assumption of relatedness between tasks, differing only in how explicitly they try to exploit this assumption. Intra-domain transfer methods rely on using this assumption implicitly and hence tend to suffer when this requirement is not met. Programmatic transfer methods also make a similar assumption, where tasks are related because by virtue of requiring similar subroutines. Cross-domain transfer methods, on the other hand, actually try to measure this relatedness and so transfers information more intelligently, and are not as susceptible to problems arising from tasks being unrelated. However, all of the above methods lack a general theory of task relatedness with which to perform transfer as intelligently as possible. And so there is as yet no clear theory of transfer algorithms that know how much information to transfer, when to transfer information, and when not to.

In this thesis, we present such a theory; we present formally optimal methods of measuring task relatedness and performing transfer in a Bayesian setting. Our method formally solves the current problems in transfer learning of determining when to transfer information when not to, and how much information to transfer. The approach we adopt is a hybrid of cross-domain transfer methods and programmatic transfer methods. We consider Bayesian learning, but the hypothesis space consists of programs computing probability measures. The measure of relatedness we use is the very general Information Distance (Bennett et al., 1998), which, in a sense, current cross-domain transfer methods approximate. Our learning method is updates according to Bayes rule in the formal setting, and posterior sampling using Markov Chain Monte Carlo methods in the practical setting. Swarup & Ray, 2006, a cross domain transfer method, also use stochastic learning methods. Among the programmatic transfer methods OOPS uses a similar stochastic depth first search through program space, however Gödel machine uses proof search, which may or may not be stochastic depending on the algorithm used (Fitting, 1996). In further contrast to the latter two, the practical approximations to our method we construct are easily able to make use of features provided in the problem description and also converge within an acceptable period of time by using Markov chain Monte Carlo methods (Chap 6).

Finally, our transfer method also translates quite readily to the Bayesian Reinforcement Learning framework (Dearden et al., 1998) via the results in Hutter, 2004. However the optimality results are weaker, and so this will need to be explored further in future work.

Chapter 3

Universal Transfer Learning Distances

In the previous chapter we looked at existing methods for performing transfer and in the process determined that the key unsolved problem here is that it is not clear how to measure task relatedness. We also showed that this problem makes it difficult to design algorithms that know how much information to transfer, when to transfer information and when not to. In this chapter we give a formally optimal solution to the problem of measuring task relatedness. Then in the next chapter we show how this may be used to derive formally optimal Bayesian transfer learning methods, and solve the problems with actually performing transfer. We proceed as follows.

First we introduce some basic notation, notions and some concepts from computability of real functions that we use. Then we describe the space of probability measures that we use as our task space. This space will be shown to be sufficiently general for the purposes of machine learning. Finally, we describe our universally optimal measures of transfer learning distance and show the sense in which this measure is optimal. Our goal in this chapter will be to explore how much similarity between tasks we can uncover using computers/Turing machines given infinite time and memory. We will explore this using tools from Algorithmic Information Theory.

3.1 Fundamentals

We use $a := b$ to mean expression a is defined by expression b. We use $\mathbb{N}_m$ to denote the numbers $1, 2, \cdots, m$. For any finite alphabet A, we use A^*, A^n, A^∞ to denote the set of all finite strings, length n strings and infinite sequences in A respectively. Let ε be the empty string. For $x, y \in A^*$, xy denotes y concatenated to the end of x. Let $l(x)$ denote the length of a finite string x. We will use $x_{1:t}$ to denote the first t elements of a sequence x and $x_{<t}$ the elements $x_{1:t-1}$. We use $x_{t:t}$ to refer to the t^{th} letter of the sequence x, and reserve single indices x_i to refer to different sequences. We will use $x_{1,n}$ as a shorthand for $x_1, x_2, \cdots, x_n$.

We use $\langle \cdot, \cdot \rangle$ to denote a standard bijective mapping from $A^* \times A^* \to A^*$.

$\langle\rangle^m$ denotes the m-arity version of this, and $\rangle\langle_i^m$ denotes the i^{th} component of the inverse of $\langle\rangle^m$. We assume the standard 'lexicographical' correspondence between A^* and $\mathbb{N}$ – e.g. for $A := \{0,1\}$:

$$\begin{aligned} \varepsilon &\leftrightarrow 0 \\ 1 &\leftrightarrow 1 \\ 00 &\leftrightarrow 2 \\ 01 &\leftrightarrow 3 \\ 11 &\leftrightarrow 4 \\ 000 &\leftrightarrow 5 \\ 001 &\leftrightarrow 6 \\ &\dots \end{aligned}$$

Depending on the context, elements of each pair will be used interchangeably (so 01 (and 4) may mean either 01 or 4). A rational number a/b is represented by $\langle a,b\rangle$. We use $\stackrel{+}{\leq}$ to denote $\leq$ upto an additive constant independent of the variables in the relation i.e. $f(x) \stackrel{+}{\leq} g(x) \equiv f(x) \leq g(x) + c$. We use the same convention for all the usual binary inequality relations. Let $2^{-\infty} := 0$, $\log := \log_2$ and $\bar{m}$ the self-delimiting encoding of $m \in \mathbb{N}$ using $l(m) + 2l(l(m)) + 1$ bits where $l(m) = \lfloor \log(m+1) \rfloor$ (Li & Vitanyi, 1997). Self-delimiting means that, given $\bar{m}$ is embedded in some longer bit string, and we are given where $\bar{m}$ begins, we can determine its end point (and hence m) without any further information.

We fix a reference prefix universal Turing machine $U : \mathcal{B}^* \times \mathcal{A}^* \to \mathcal{A}^*$, where $\mathcal{B} := \{0,1\}$ is the alphabet for programs, and $\mathcal{A}$, $\mathcal{A} \supset \mathcal{B}$, is an arbitrary alphabet for inputs and outputs. 'Prefix' means that the valid programs for U form a prefix free set, that is no program is a prefix of another. Standard programming languages satisfy this property by virtue of the begin and end markers (e.g. { and } in C,C++ and Java). The prefix property also entails another property which is crucial for us, which is that the lengths of valid programs p satisfy the **Kraft inequality** (see Li & Vitanyi, 1997; Cover & Thomas, 1991):

$$\sum_p 2^{-l(p)} \leq 1$$

$U(p,x)$ denotes running the program p on input x. When it is clear from the context that p is a program, we will denote $U(p,x)$ simply by $p(x)$. We need some notions of computability of real functions.

Definition 3.1. *A real function $f : \mathcal{A}^* \to \mathbb{R}$ is* **upper semicomputable** *if there is a program p such that for $x, t \in \mathbb{N}$,*

1. $p(\langle x, t\rangle)$ *halts in finite time*

2. $p(\langle x, t\rangle) \geq p(\langle x, t+1\rangle)$

3. $\lim_{t\to\infty} p(\langle x, t\rangle) = f(x)$.

A real function $f : \mathcal{A}^* \to \mathbb{R}$ *is* **lower semicomputable** *if* $-f$ *is upper semicomputable.*

A function $f : \mathcal{A}^* \to \mathbb{R}$ *is* **computable/recursive** *if there is a* p *such that* $\forall n, x \in \mathbb{N}$*,*

1. $|p(\langle x, n\rangle) - f(x)| < 2^{-n}$

2. $p(\langle x, n\rangle)$ *halts in finite time.*

We use $p(x) \Uparrow q(x)$ *to denote that at* x p *and* q *lower semicomputes the same function.* ■

So function f is upper semi-computable when there is a program that computes smaller and smaller approximations to $f(x)$ in finite time, but we never know how close the approximation is. Function f is lower semi-computable when there is a program that computes larger and larger approximations to $f(x)$ in finite time, but we never know how close the approximation is. And finally, a function f is computable when we can compute f to any arbitrarily specified precision in finite time.

As mentioned earlier, in this work we wish to investigate how much similarity we can uncover and the best transfer method we can derive, given infinite resources. We achieve this goal by considering only similarity functions and probability measures that are upper and lower semicomputable respectively. The reason for choosing lower semicomputable probability measures instead of upper semicomputable ones is given in the next section. The reason for choosing upper semicomputable similarity functions instead of lower semicomputable ones is because this is the only way we can use the similarity functions to induce lower semicomputable probability measures – see Sect. 4.2.

3.2 The Task Space and the Learning Problem

In transfer learning we wish to transfer information between tasks. Each task is a learning problem and recall that each task is identified with the probability measure generating the samples for that problem. Now the question becomes, which class of probability measures should we consider as our task space ? For a transfer framework to be reasonably powerful, it seems appropriate to require that any computable probability measure should be included in that class, as any problem that

we hope to be able to solve will either itself be computable, or have a reasonable approximation that is computable. In this section we describe such a class, which is the set of all lower semicomputable *semimeasures*. This space was introduced for use in inductive inference in Solomonoff, 1978, and discussed previously in Zvonkin & Levin, 1970 (see below). We also consider *sequence prediction tasks* instead of typical i.i.d. tasks considered in most machine learning literature. Sequence prediction tasks are generalizations of the i.i.d. case and so we do not lose anything by this choice. We refer the reader to Chap. 6 and Hutter, 2003, Sect. 6.2 for details on this issue.

As mentioned above, our task space is a particular subset of the set of all *semimeasures*:

Definition 3.2. *A* **semimeasure** *is a function* $f : \mathcal{A}^* \to [0,1]$ *such that*

$$\forall x \in \mathcal{A}^*, f(x) \geq \sum_{a \in \mathcal{A}} f(xa).$$

■

So $f(x_{1:t})$ is the 'defective probability' that a particular infinite sequence starts with the prefix $x_{1:t}$ (f is a probability measure if $f(\varepsilon) = 1$ and the inequality is an equality). So f is equivalent to a probability measure p defined on $[0,1]$ such that $f(x) = p([0.x_{1:t}, 0.x_{1:t} + |\mathcal{A}|^t))$ where $0.x_{1:t}$ is in base $|\mathcal{A}|$. The conditional probability of the next letter being a given the string $x_{1:t}$ observed so far is

$$f(a|x_{1:t}) := \frac{f(x_{1:t}a)}{f(x_{1:t})}$$

Zvonkin & Levin, 1970 showed that the set of all lower semicomputable semimeasures is recursively enumerable. That is, there is a Turing machine T such that $T(\langle i, \cdot \rangle)$ lower semicomputes $f_i(\cdot)$, the i^{th} semimeasure in this effective enumeration. Since U is universal, for each $i \in \mathbb{N}$, there is a program p_i such that $p_i(x) = T(\langle i, x \rangle)$. Let $\mathcal{V}$ be the enumeration of these programs – i.e. $p_i \in \mathcal{V}$ lower semicomputes f_i, and each lower semicomputable semimeasure f is computed by at least one $p_j \in \mathcal{V}$. We will consider enumerable subsets $\mathcal{V}'$ of $\mathcal{V}$ as our task space, as any probability measure that we may expect to be able to learn must either be computable, or have a reasonable approximation (however it may be defined) that is computable. $\mathcal{V}$ is the largest superset of this that contains any Bayes mixture of its own elements, which is important in Chap. 4 (see also Hutter, 2003, Sect. 2.6 and Li & Vitanyi, 1997). See also Hutter, 2004, Sect. 2.4.3 for more details on the class of semimeasures that contains mixtures of its own elements.

The learning problem we consider is the online Bayesian sequence prediction

setting (Fig. 3.1) :

Definition 3.3. (Learning in Bayesian Sequence Prediction) *When learning task μ, at each step t, $a \in \mathcal{A}$ is generated according to $\mu(.|x_{<t})$, where $x_{<t}$ was generated by μ in the previous $t-1$ steps. The learning problem is to predict the letter a at each step* ■

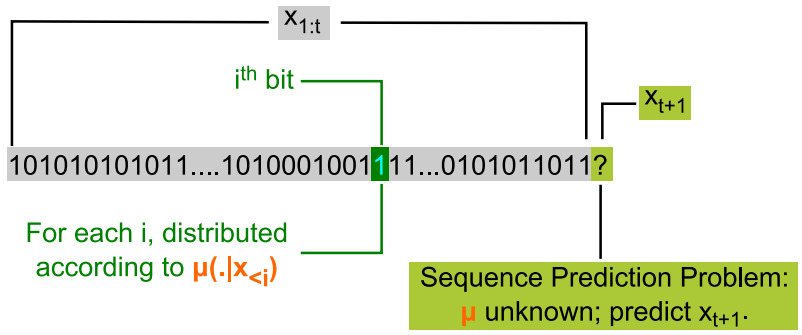

Figure 3.1: The Sequence Prediction problem when $\mathcal{A} \equiv \mathcal{B}$.

3.3 Distance Function for Tasks

In this section we will define our universally optimal measure of transfer learning distance. We will start by showing why classical Information Theory is inadequate for our purposes and through that motivate the use of Algorithmic Information Theory to measure relatedness between tasks, i.e. transfer learning distance. We will then define our measure and describe its optimality properties, which will be used later to derive universally optimal Bayesian transfer learning methods in Chap. 4.

3.3.1 Kolmogorov Complexity Basics

The main unsolved problem in transfer learning, as discussed in the preceding chapters, is to measure the amount of *information* tasks contain about each other. The traditional and accepted measure of information of a probability measure, which we have identified as our tasks, is the well known Information Theoretic Entropy (Cover & Thomas, 1991). For a probability measure P over a countable set X, this is defined as:

$$H(X) := -\sum_{x \in X} P(x) \log P(x) \ .$$

So $H(X)$ measures the information content of the ensemble of objects X with respect to the measure P. The amount of information that a measure Q contains

about P is given by the relative entropy/KL divergence:

$$D(P||Q) := -\sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)} \ .$$

Note that both P and Q need to be defined over the same set X. In transfer learning, we are interested in how much information measure $\mu, \varphi \in \mathcal{V}$ contain about each other – and it seems at first blush that KL divergence should be adequate for this purpose. However, it is often the case that we wish to transfer between measures defined on different spaces (Swarup & Ray, 2006) i.e. there exists points at which $Q(x) = 0$, but $P(x) \neq 0$, and in this case KL divergence is undefined. Furthermore, what we are really interested in is not just any type of information, but amount of constructive information that P and Q contain about each other – i.e. amount of information P contains for the purpose of constructing measure Q and vice versa. So in this case classical Information Theory does not give us an appropriate measure. To solve this problem, we have to turn to Algorithmic Information Theory (AIT) (see Li & Vitanyi, 1997 for the results below and a comprehensive introduction to AIT).

AIT remedies the above problem by giving a *constructive* measure of information that *individual* objects contain about each other via the beautiful notion of (prefix) Kolmogorov complexity of strings (Levin, 1974; Gacs, 1974; Chaitin, 1975):

Definition 3.4. *The* **Kolmogorov Complexity** *of $x \in \mathcal{A}^*$ is given by the length of the shortest program that on input ε, outputs x:*

$$K(x) := \min_p \{l(p) : p(\varepsilon) = x\} \ .$$

■

The intuition is that, the minimum number of bits we would need to communicate to someone so that they can reconstruct the string x is the length of the shortest program p the outputs x given no input. Hence the length of p is a measure of the amount of absolute information content of $x \in \mathcal{A}^*$. As we shall see, for transfer learning we will need to measure the amount of information a string y contains about another string x, and this is given by the conditional version of K:

Definition 3.5. *The* **conditional Kolmogorov complexity** *of x given y, $x, y \in \mathcal{A}^*$, is the length of the shortest program that outputs x given y :*

$$K(x|y) := \min_p \{l(p) : p(y) = x\} \ .$$

■

Again, the intuition is that length of $p := \arg K(x|y)$ is the minimum number of bits we would need to communicate to someone so that they can reconstruct x given that they already have string y. Hence $l(p)$ is an absolute measure of the amount of information that y contains for the purpose of constructing x. Not only are these quantities intuitively satisfying, as we shall soon show, the Kolmogorov complexity and conditional Kolmogorov complexity are both sharper versions of the entropy and conditional entropy in classical Information Theory.

To define the above quantities for m strings we simply use the $\langle\rangle^m$ map to encode the m strings to a single string, and use the definitions on that single string. So for instance $K(x, y|z, w, v) := K(\langle x, y\rangle|\langle z, w, v\rangle)$ etc. This does not cause any problems because $\langle\rangle^m$ is computable with a short, constant length program computing it. We also note that fixing U as a reference universal Turing machine does not cause problem because of the celebrated *Invariance Theorem* (Kolmogorov, 1965): given any two universal Turing machines U_i and U_j

$$|K_{U_i}(x|y) - K_{U_j}(x|y)| \stackrel{+}{=} 0$$

where K_Z is the conditional Kolmogorov complexity where the programs are for universal Turing machine Z. We list some fundamental properties of K:

Lemma 3.1. $\forall x, y, y_{1,m} \in \mathcal{A}^*$*:*

1. $K(x|y) \stackrel{+}{\leq} K(x)$.
2. $K(x) \stackrel{+}{\leq} K(\langle x, y_{1,m-1}\rangle^m)$.
3. *The function $K(x|y)$ is upper semicomputable.*
4. $K(x|\arg K(y)) + K(y) \stackrel{+}{=} K(x, y)$ *(Gacs, 1974; Chaitin, 1975)*
5. $K(x, y|z) \stackrel{+}{=} K(x|\arg K(y|z), z) + K(y|z) = K(x|y, K(y|z), z) + K(y|z)$.

Proof. (*Sketch*) The first two properties follow from the definition of the K functions and the following. The first property follows from the fact that any program that computes x, with a constant length modification to ignore any input, is also a program to output x given y. The second property follows because any program that outputs $\langle x, y_{1,m-1}\rangle^m$, with a constant length modification to output $\rangle\langle x, y_{1,m-1}\rangle^m\langle^m_1$, also outputs x.

For the third property, we note that the following program p upper semicomputes $K(x|y)$: $p(\langle\langle x, y\rangle, t\rangle)$ runs all programs q on y with $l(q) \leq 2l(x)$ (a loose upper bound on $K(x|y)$), in parallel by 'dovetailing', for t steps each. p then outputs the length of the shortest program found thus far.

The fourth property, discovered first by Gacs, and then independently by Chaitin, is one of the deepest and fundamental results in Kolmogorov complexity theory. The proof is quite long and complex, and we refer the reader to Li & Vitanyi, 1997. The fifth property is a conditional version of the the fourth property – note the lack of a constant of equality in the final equality. This is because $\arg K(y|z)$ and $\langle y, K(y|z)\rangle$ contain the same amount of information – given one we can compute the other and vice versa.. $\square$

The function $K(x|y)$ is upper semicomputable which is in agreement with our goal to investigate what type of transfer is possible given infinite resources. We will also make extensive use of the following minimality property of $K(x|y)$:

Lemma 3.2. *For any partial, non-negative, upper semicomputable function $f : \mathcal{A}^* \times \mathcal{A}^* \to \mathbb{R}$, with $f(x,y) = \infty$ when it is undefined, we have:*

$$K(x|y) \stackrel{+}{\leq} f(x,y) \quad \textit{if} \quad \sum_x 2^{-f(x,y)} \leq 1 \ .$$

where the constant in $\stackrel{+}{\leq}$ is equal to $K(f) + O(1)$ where $O(1)$ is quite small (see Li & Vitanyi, 1997).

In the above lemma the dependence of the constant on $K(f)$ can be ignored in this work for two reasons. First, in our applications f will either be symmetric distance functions (see Def. 3.8) and Bayesian priors (see Def. 4.4). We assume that all such distance functions and probability measures are *reasonable* – i.e. that they have short $O(1)$ length. That this is a acceptable assumption to make can be seen by contemplating the distance functions and priors used in practice. Second, should the reader find the first assumption onerous, we refer them to Sect. 4.4, where we dispense with even this very reasonable assumption and induce a different and arguably more robust interpretation of our optimal methods.

Using the above lemma, we can now show the relationship between K and H, where H is the classical information theoretic entropy. Set $y := \varepsilon$, and let $P \in \mathcal{V}$. Now $f := -\log P$ satisfies the condition for lemma 3.2. Hence we have

$$K(x) \stackrel{+}{\leq} -\log P(x) \ .$$

Taking expectation with respect to P, we have

$$E_P(K(x)) \stackrel{+}{\leq} H(x) \ .$$

However, since $K(x)$ is code word length of a prefix free code, the Noiseless Cod-

ing Theorem (Cover & Thomas, 1991) states:

$$H(x) \leq E_P(K(x)) \ .$$

And hence, we have that P-expected $K(x)$ is, upto an additive constant equal to $H(P)$. In a similar manner it can be shown that $H(P|Q)$ is equal to P-expected $K(x|y)$.

3.3.2 Universal Transfer Learning Distance for Tasks

$K(x|y)$ measures the amount of information string y contains about x. Now, to measure the amount of information string x contains about string y *and* string y contains about string x, Bennett et al., 1998 defined the following function:

Definition 3.6. *The* **Information Distance** *between $x, y \in \mathcal{A}^*$ is the length of the shortest program that given x outputs y, and vice versa:*

$$E_0(x, y) := \min_p \{l(p) : p(x) = y, p(y) = x\} \ .$$

■

The punchline is as now this. For $\mu, \varphi \in \mathcal{V}$, μ and φ are also strings (interpreted as programs). Hence $K(\mu|\varphi)$ measures the amount of information φ contains for the purpose of constructing μ, which is exactly the type of information we want to measure for transfer learning. Similarly, $E_0(\mu, \varphi)$ measures the amount of constructive information μ and φ contain about each other, which is exactly the measure of distance that we have been looking for when trying to measure task relatedness. Furthermore, both K and E_0 are upper semicomputable, which is again in agreement with our desire to investigate transfer in the limit. Upper semicomputability of K was established in Lemma 3.1. To do the same for E_0, consider the following program p that upper semicomputes $E_0(x, y)$. $p(\langle\langle x, y\rangle, t\rangle)$ runs all programs q on y and x, with $l(q) \leq 2 \max\{l(x), l(y)\}$ (a loose upper bound on $E_0(x, y)$), in parallel by 'dovetailing', for t steps each. p then outputs the length of the shortest program found thus far.

Hence E_0 is the natural candidate for a transfer learning distance. We will however use a sharper characterization of E_0:

Definition 3.7. *The* **Cognitive Distance** *between $x, y \in \mathcal{A}^*$ is given by*

$$E_1(x, y) := \max\{K(x|y), K(y|x)\} \ .$$

■

E_1 is upper semicomputable - we simply upper semicompute in 'parallel' (by dovetailing) each term in the definition of E_1. Bennett et al., 1998 proved:

Theorem 3.1.

$$E_0(x,y) = E_1(x,y) + O[\log(E_1(x,y))] \ .$$

The above has been termed the *conversion theorem*. Hence, E_1 is considered a sharper version of E_0 because it is expressed in terms of the more well understood and investigated function K and is also equal to E_0 upto a logarithmic term. In fact, we will enhance the status of E_0 by proving an improved version of Theorem 3.1 where the $\log$ term is replaced by a constant.

The reason E_1 is particularly interesting is because it uncovers, in a very formal and precise sense, more information than any other *admissible distance* function, which is a class of distances that we define below. The reason it is sufficient to consider only admissible distances in this dissertation is because, as we show Sect. 4.2, any transfer learning distance function that can be used to construct a transfer learning algorithm in a Bayesian setting must be an admissible distance. So without further ado, we define:

Definition 3.8. *An* **admissible distance** D *is a partial, upper semicomputable, non–negative, symmetric function on* $\mathcal{A}^* \times \mathcal{A}^*$ *with* $\forall y$

$$\sum_x 2^{-D(x,y)} \leq 1 \ .$$

(we will assume $D(x,y) = \infty$ *when it is undefined). Let* $\mathcal{D}$ *be the set of admissible distances. A* $D \in \mathcal{D}$ *is* **universal** *in* $\mathcal{D}$ *if* $\forall D' \in \mathcal{D}, \forall x,y \in \mathcal{A}^*$, $D(x,y) \stackrel{+}{\leq} D'(x,y)$. ■

Bennett et al., 1998 also showed that

Theorem 3.2. $\forall D \in \mathcal{D}, \forall x,y \in \mathcal{A}^*$

$$E_1(x,y) \stackrel{+}{\leq} D(x,y) \ . \tag{3.1}$$

That is, E_1 is universal in $\mathcal{D}$ (this was proved via Lemma 3.2 with $f = D$, as D satisfies the requisite conditions due to its admissibility).

So what the inequality in Theorem 3.2 translates to is that E_1 uncovers more similarity than any other admissible distance function. In the paper Bennett et al., 1998 itself the authors showed that the above holds for admissible *metrics*, but as pointed out in Li et al., 2004 this holds for admissible distances as well. Admissible distances include admissible versions of Hamming, Edit, Euclidean, Lempel-Ziv etc. distances (Bennett et al., 1998; Li et al., 2004; Cilibrasi & Vitanyi, 2005).

See Bennett et al., 1998 for an eloquent account of why admissible distances (and distances satisfying the Kraft Inequality) are interesting for strings. Normalized, practical versions of E_1 has been applied very successfully in various clustering tasks (Li et al., 2004; Cilibrasi & Vitanyi, 2005).

We now state our improvement of the conversion theorem, (proof in Sect.).

Theorem 3.3.

$$E_0(x,y) \stackrel{+}{=} E_1(x,y) \ .$$

Given Theorem 3.3, we now define:

Definition 3.9. *The* **transfer learning distance** *between two tasks $\mu, \varphi \in \mathcal{V}$ is defined as $E_1(\mu, \varphi)$.* ■

So from the above, we immediately get that transfer learning distance is universal in the class of admissible distances that may be used for measuring task similarity. This formally solves the conceptual problem of how one measures task similarity. We will use this distance function in Chap. 4 to formally solve other problems in transfer learning mentioned in the Introduction and give more reasons why it is sufficient to consider only admissible distances (see discussion following the proof of Theorem 4.3).

3.3.3 Proof of Theorem 3.3

Proof. Let p be a program such that $p(x) = y$ and $p(y) = x$. So by definition $E_1(x,y) \leq l(p)$ for all such p. Since $\arg E_0(x,y)$ is a such a p, we have $E_1(x,y) \stackrel{+}{\leq} E_0(x,y)$.Now we prove the inequality in the other direction. Fix any two strings α, β and set $E_1(\alpha,\beta) = E1$. Now we will derive a program q_{E1} with $l(q_{E1}) \stackrel{+}{=} E1$ which given α outputs β and given β outputs α. We will do so by constructing a graph G that assigns a unique color/code of length $\leq E1 + 1$ to each pair of strings x, y with $E_1(x,y) \leq E1$, and the code will turn out to be more or less the program q_{E1} we need to convert α to β and vice versa. We note that the proof of Theorem (3.1) also uses a similar graph construction method. Define $G := (V, E)$ with vertices V and undirected edges E:

$$V := \{x : x \in A\} \text{ and } E := \{\{x,y\} : x \in A, y \in A_x\}, \text{ where,}$$
$$A := \{x : \exists y, E_1(x,y) \leq E1\} \text{ and } \forall x \in A, A_x := \{y : E_1(x,y) \leq E1\} \ .$$

The degree of $x \in V$ is $|A_x|$ by construction. Hence the maximum degree of G is $\Delta_G = \max_{x \in A} |A_x|$. We define the set of colors/code $\mathcal{C}_{E1}$ as:

$$\mathcal{C}_{E1} := \{p0 : p \in B\} \cup \{p1 : p \in B\}, \text{ where,}$$
$$B := \{p : p(x) = y, x \in A, y \in A_x, l(p) \leq E1\} \ .$$

q_{E1} will need to dynamically construct G and $\mathcal{C}_{E1}$, and assign a *valid* coloring to the edges in G using $\mathcal{C}_{E1}$. For this, all we need is $E1$. We run all programs p with $l(p) \leq E1$ on all $x \in \mathcal{A}^*$ in 'parallel' by dovetailing and record triples (p, x, y) such that $p(x) = y$. Whenever we record (p, x, y) we check to see if we have previously recorded (q, y, x). If so, we add $p0, p1, q0, q1$ to $\mathcal{C}_{E1}$, x, y to V and $\{x, y\}$ to E. Of course, if any of these already exist in the respective sets, we do not add it again. We color a newly added edge $\{x, y\}$ using a color from $\mathcal{C}_{E1}$ using the First-Fit algorithm - i.e. the first color that has not been assigned to any other $\{x, w\}$ or $\{y, z\}$. So, by dynamically reconstructing G, given x (y) and the color for $\{x, y\}$, q_{E1} can use the color to recognize and output y (x).

That $\mathcal{C}_{E1}$ has sufficient colors to allow valid coloring can be seen as follows. $p \in B$ iff $l(p) \leq E1$ and for some A_x, $y \in A_x, p(x) = y$. So for each A_x, for each $y \in A_x$, $\exists p_y \in B$, and $p_y \neq p_{y'}$ $\forall y' \in A_x, y' \neq y$ since $p_y(x) \neq y'$. This means, for each A_x, $|\mathcal{C}_{E1}| \geq 2|A_x|$, or $|\mathcal{C}_{E1}| \geq 2\Delta_G$. By the same reasoning and the construction procedure above, as we dynamically construct G and $\mathcal{C}_{E1}$, the estimates $\mathcal{C}^t_{E1}$ and Δ^t_G at step t of the construction process also satisfies $|\mathcal{C}^t_{E1}| \geq 2\Delta^t_G$. Now at step t First-Fit requires at most $2\Delta^t_G - 1$ colors to assign a valid color, as two vertices could have exhausted at most $2\Delta^t_G - 2$ colors between them. Therefore First-Fit always has sufficient colors to assign a valid coloring.

Each color/code in $\mathcal{C}_{E1}$ is at most $E1 + 1$ in length by construction. So, as we construct G, α and β shows up in the graph at some point with code/color (say) γ, and $l(\gamma) \leq E1 + 1$. From construction of $\mathcal{C}_{E1}$, γ is a self-delimiting string p, followed by 0 or 1. γ and $E1$ can be encoded by a string $pa0^{E1-l(p)}1$, where a is 0 if $\gamma = p0$, or 1 if $\gamma = p1$, and $0^{E1-l(p)}$ is 0 repeated $E1 - l(p)$ times.

The desired program q_{E1} has encoded in it the string $pa0^{E1-l(p)}1$ at some fixed position, and $q_{E1}(z)$ works as follows. q_{E1} decodes p (which is possible as it is self-delimiting) and then reads the next bit, which is a, to get γ. It computes $E1$ from counting the number of 0s after a and $l(p)$. When $a = 0$, it is not confused with the 0s following it because it is the bit that appears immediately after p, and p can be decoded by itself. q_{E1} then reconstructs G using $E1$, and finds the edge $\{z, w\}$ with color γ, and outputs w. By construction, if $z = \alpha$ then $w = \beta$ and if $z = \beta$ then $w = \alpha$. Since $l(q_{E1}) \stackrel{+}{=} E1$ (the constant being for the extra bits in $pa0^{E1-l(p)}1$ and other program code in q), we have $E_0(\alpha, \beta) \leq l(q_{E1}) \stackrel{+}{=}$

$E_1(\alpha, \beta)$, and therefore $E_0(\alpha, \beta) \stackrel{+}{=} E_1(\alpha, \beta)$. □

3.3.4 Universal Transfer Learning Distance for m Tasks

In this section we will extend the definition of transfer learning distance to m task case. The material in this section may be skipped as it is not used below, but we include it here for completeness and because the results are interesting in and of themselves. We also hope that the functions here will find application in task clustering problems which are important for designing 'Long Lived' transfer learning agents (Thrun & Pratt, 1998), and in clustering problems in general, as in Cilibrasi & Vitanyi, 2005. The distance functions in this section apply to arbitrary strings in addition to elements of $\mathcal{V}$.

Let $X := \{x_{1,m}\}, x_j \in \mathcal{A}^*$, $X_i^{m_1}$ the i^{th} subset of X of size m_1, $0 < m_1 < m$, $0 < i < \binom{m}{m_1}$. Let $\sigma(X_i^{m_1})$ be the set of permutations of elements of $X_i^{m_1}$. Then, to generalize E_0 to measure how much each group of m_1 x_js, $0 < m_1 < m$, contain about the other $m - m_1$ x_js, we define:

Definition 3.10. *The m* **fold information distance** *$E_0^m(x_{1,m})$ between $x_{1,m} \in \mathcal{A}^*$, is the length of the shortest program that given any permutation of m_1 x_js, $1 < m_1 < m$, outputs a permutation of the other $m - m_1$ x_js. That is:*

$$E_0^m(x_{1,m}) := \min_p \{l(p) : \forall m_1, i, x,\ 0 < m_1 < m,\ 1 \leq i \leq \binom{m}{m_1},$$
$$x \in \sigma(X_i^{m_1}),\ p(\langle\langle x\rangle^{m_1}, m_1\rangle) = \langle y\rangle^{m-m_1},\ \textit{where } y \in \sigma(X \backslash X_i^{m_1})\} \ .$$

■

In contrast to E_0 the additional information m_1 is included in the definition for E_0^m to determine how to interpret the input, – i.e. which $\rangle\langle^{m_1}$ to use to decode the input. E_0^m is upper semicomputable by the same reasoning E_0 is (Bennett et al., 1998). To give a sharper characterization of E_0^m, we define:

Definition 3.11. *The m* **fold Cognitive Distance** *for $x_{1,m} \in \mathcal{A}^*$ is:*

$$E_1^m(x_{1,m}) := \max_{x_i} \max_{y \in \sigma(X \backslash \{x_i\})} E_1(x_i, \langle y\rangle^{m-1}) \ .$$

■

E_1^m is upper semicomputable by the same reasoning E_1 is. We can now state the analogue of Theorem 3.3 for m strings (the proof is given below):

Theorem 3.4.

$$E_0^m(x_{1,m}) \stackrel{+}{=} E_1^m(x_{1,m}) \ .$$

Definition 3.12. *The m-**fold transfer learning distance** between m tasks $\mu_{1,m} \in \mathcal{V}$ is defined as $E_1^m(\mu_{1,m})$.* ■

We can also define admissible distances:

Definition 3.13. *The m **fold admissible distances** between m tasks $\mu_{1,m} \in \mathcal{V}$ are defined as functions $D_m : \times_m \mathcal{A}^* \to \mathbb{R}$ that are non-negative, upper semi-computable, m-wise symmetric, and satisfies the following version of the Kraft inequality: $\forall x, y_{1,m-1} \in \mathcal{A}^*$*

$$\sum_{z_{1,m-1} \in \mathcal{A}^*} 2^{-D_m(x, z_{1,m-1})} \leq 1 \text{ and } \sum_{w \in \mathcal{A}^*} 2^{-D_m(w, y_{1,m-1})} \leq 1 \ .$$

Let $\mathcal{D}_m$ be the set of m fold admissible distances. A $D \in \mathcal{D}$ is **universal** *in $\mathcal{D}$ if $\forall D' \in \mathcal{D}, \forall x_{1,m} \in \mathcal{A}^*$, $D(x_{1,m}) \stackrel{+}{\leq} D'(x_{1,m})$.* ■

Theorem 3.5. *E_1^m has the following properties:*

1. *E_1^m satisfies the above version of the Kraft inequality.*
2. *E_1^m is universal in the class of admissible distances for m strings.*

Proof. Let $x, y_{1,m-1} \in \mathcal{A}^*$. Part 1 follows because by definition

$$E_1(x, \langle y_{1,m-1} \rangle^{m-1}) \leq E_1^m(x, y_{1,m-1}) \ .$$

and $E_1(x, \langle y_{1,m-1} \rangle^{m-1})$ satisfies the Kraft inequality. For part 2, by Lemma 3.2 and admissibility of D_m:

$$K(x|y_{1,m-1}), K(y_{1,m-1}|x) \stackrel{+}{\leq} D_m(x, y_{1,m-1}) \ .$$

The desired result now follows because by definition,

$$E_1^m(x, y_{1,m-1}) \leq K(x|y_{1,m-1}), K(y_{1,m-1}|x) \ .$$

□

3.3.5 Proof of Theorem 3.4

Proof. The proof is similar to the proof of Theorem 3.3 - we assume m is fixed and treat it as a constant. Otherwise the theorem holds upto additive $m \log m$ terms.

Fix $\Lambda := \{\lambda_{1,m}\}$. We will first show

$$E_1^m(\lambda_{1,m}) \stackrel{+}{\leq} E_0^m(\lambda_{1,m}) \ .$$

Let p be a program such that $\forall m_1, i, \lambda,\ 0 < m_1 < m,\ 1 \leq i \leq \binom{m}{m_1},\ \lambda \in \sigma(\Lambda_i^{m_1}),\ p(\langle\langle\lambda\rangle^{m_1}, m_1\rangle) = \langle y\rangle^{m-m_1}$, where $y \in \sigma(\Lambda\backslash\Lambda_i^{m_1})$. Fix and $i \in \mathbb{N}_m$ and $\eta \in \sigma(\Lambda\backslash\Lambda_i^{m_1})$. Then we can construct 1) a program q that given any λ_i outputs η and 2) a program q' that given η outputs λ_i and $l(p) \stackrel{+}{=} l(q) \stackrel{+}{=} l(q')$. The program q operates as follows. Given input x, it runs $p(\langle x, 1\rangle)$ and if $x = \lambda_i$, gets a $y \in \sigma(\Lambda\backslash\Lambda_i^{m-1})$. q also has encoded in it as $2m\bar{m}$ in $< 4m\log m$ bits the order in which $\lambda_j, j \neq i$ appears in y, and in which they should appear in η as (for definition of $\bar{m}$ see Sect. 3.1). It then uses that to decode y, and output η. The program q' operates as follows - given input x, it runs $p(\langle x, m-1\rangle)$, needing $< 2\log m$ bits to encode $m-1$ as $\overline{m-1}$. If $x := \eta\ q'$ gets λ_i and just outputs it. By construction $l(p) \stackrel{+}{=} l(q) \stackrel{+}{=} l(q')$; furthermore $\arg E_0^m(\lambda_{1,m})$ is a program satisfying the properties of p, while, since λ_i and η were chosen arbitrarily, $\arg E_1^m(\lambda_{1,m})$ is a program satisfying either the properties of q or q'. Hence $l(\arg E_1^m(\lambda_{1,m}))$ is at most $l(\arg E_0^m(\lambda_{1,m}))$ and so we have the above inequality.

Now we prove

$$E_0^m(\lambda_{1,m}) \stackrel{+}{\leq} E_1^m(\lambda_{1,m})\ .$$

Let $E1m = E_1^m(\lambda_{1,m})$. We will construct a program q_{E1m} with $l(q_{E1m}) \stackrel{+}{=} E1m$ that will have the same outputs as $\arg E_0^m(\lambda_{1,m})$ on $\langle\langle y\rangle^{m_1}, m_1\rangle$, $y \in \sigma(\Lambda_i^{m_1}), 0 < m_1 < m, 1 \leq i \leq \binom{m}{m_1}$. For this, we need the set L the sets A_x

$$\begin{aligned} L &:= \{\{x_{1,m}\} : E_1^m(x_{1,m}) \leq E1m\} \\ A_x &:= \{\{z_{1,m-1}\} : \{x, z_{1,m-1}\} \in L\}\ . \end{aligned}$$

and colors $\mathcal{C}_{E1m}$, defined using the set B.

$$\begin{aligned} &\mathcal{C}_{E1m} := \{pj : p \in B, j \leq m\}, \text{ where,} \\ &B := \{p : p(x) = \langle y_{1,m-1}\rangle^{m-1}, \{y_{1,m-1}\} \in A_x, l(p) \leq E1m\}\ . \end{aligned}$$

By using $E1m$ and m, q_{E1m} will construct L dynamically and color each element of L using colors from $\mathcal{C}_{E1m}$, so that if a string x_i appears in multiple m tuples in L, then each m tuple will have a different color from the m tuples - this is stated more precisely below.

To perform the coloring as above, we run all programs p with $l(p) \leq E1m$ on all $x \in \mathcal{A}^*$ in parallel. If we find $p(x) = y$, we record the tuples $(p, (w_{1,m-1}), y)$ and $(p, x, (z_{1,m-1}))$, where $x = \langle w_{1,m-1}\rangle^{m-1}$ and $y = \langle z_{1,m-1}\rangle^{m-1}$. If we find a $x_{1,m}$ such that we have recorded $(p_{x_i,y}, x_i, y)$ and (p_{y,x_i}, y, x_i) for each x_i and $\forall y \in \sigma(\{x_{1,m}\}\backslash\{x_i\})$, then we add each of the $p_{x_i,y}, p_{y,x_i}$s to B and add the corresponding colors to $\mathcal{C}_{E1m}$. We add $X := \{x_{1,m}\}$ to L and color it using a

variation of First-Fit in Theorem 3.3 as follows. Denote by $\mathcal{C}(X)$ the color assigned to X. Then $\mathcal{C}(X)$ is set to the first $\gamma \in \mathcal{C}_{E1m}$ such that $\forall x \in X$, if $x \in X'$, $X' \in L$, then $\gamma \neq \mathcal{C}(X')$. *So given any $x \in X$, and $\mathcal{C}(X)$, q_{E1m} can reconstruct and color L as above and hence find X.*

To see that $\mathcal{C}_{E1m}$ has enough colors: Let $\Delta_L := \max_x |A_x|$. For each $\kappa \in A_x$, $\exists p_\kappa \in B$, $p_\kappa(x) = \langle y \rangle^{m-1}, y \in \sigma(\kappa)$ and $p_{\kappa'} \neq p_\kappa \; \forall \kappa' \in A_x, \kappa' \neq \kappa$. Therefore $|\mathcal{C}_{E1m}| \geq m\Delta_L$. Also, from the construction method for L above, $|\mathcal{C}^t_{E1m}| \geq m\Delta^t_L$ for the estimates at each step t of the construction process. When coloring X at step t, each $x \in X$ has used $\leq \Delta^t_L - 1$ colors previously. So, as $|X| = m$, First-Fit will require at most $m(\Delta^t_L - 1) + 1$ colors to assign a valid color to X.

Now $\max_{\gamma \in \mathcal{C}_{E1m}} l(\gamma) \leq E1m + l(m)$ ($l(m) = \lfloor \log(m+1) \rfloor$ Li & Vitanyi, 1997), and with m as a constant, this becomes $E1m + c$. Like q_{E1} from Theorem 3.3, q_{E1m} can encode $E1m$, m, and the color $\gamma_\Lambda = pj$ for Λ in itself as $\bar{pj}\bar{m}0^{E1m-l(p)}1$. Using this, q_{E1m} can dynamically construct L, $\mathcal{C}_{E1m}$ and color L. For input $\langle x, m_1 \rangle$, $0 < m_1 < m$, q_{E1m} decodes x with $\beta_j := \rangle x \langle^{m_1}_j$, $0 < j < m_1$. By construction of L, using any β_j and γ_Λ, q_{E1m} can find Λ in L, and output $\langle y \rangle^{m-m_1}$, $y \in \sigma(\Lambda \backslash \{\beta_{1,m_1}\})$, which is what is required. This proves, with m as a constant $E^m_0(\lambda_{1,m}) \stackrel{+}{\leq} E^m_1(\lambda_{1,m})$ and $E^m_0(\lambda_{1,m}) \stackrel{+}{\leq} E^m_1(\lambda_{1,m}) + 3\lceil \log m \rceil$ otherwise. This and the first inequality completes the proof. □

3.4 Discussion

In this chapter we defined our universal measures of task relatedness and set the stage for use of these for developing formally optimal Bayesian transfer learning scheme in the next chapter. The capability of AIT theoretic distance measures to measure amount of constructive information that individual objects contain about each other made it possible for us to solve one of the long standing problems in transfer of how to measure relatedness between tasks. By proving the Conversion Theorem for both the 2 and m string case (very interesting results in and of themselves) we were able to give a simple to understand characterization of task relatedness. Future work for this should involve developing finitely computable versions of these distance functions. A first, and seemingly effective, attempt at such a distance is presented in Chap. 6. We believe more sophisticated distance functions can be constructed by restricting ourselves to group of specific machine learning domains and then deriving compression based distance functions suitable for measuring relatedness between hypothesis that are suitable for the group. For an impressive example of such an approach, we refer the reader to Cilibrasi & Vitanyi, 2005.

Chapter 4

Universal Bayesian Transfer Learning

In Chapter 3 we defined our transfer learning distance and established that it is formally optimal in the sense that in the limit of infinite resources, no other reasonable distance function can uncover more similarity. While this is a significant result from a conceptual point of view, we did not describe how it may be used for performing actual transfer learning. In this section we focus on this issue and present Bayesian transfer learning schemes that are *universally optimal*. We assume notation and results described in the preceding chapter and then proceed as follows.

First we describe a very general Bayesian framework and associated powerful convergence results, which together constitute Solomonoff Induction (Solomonoff, 1978; Hutter, 2003). We then use results from the previous chapter to define universally optimal transfer learning schemes for this setting. We consider two types of transfer learning frameworks, sequential and parallel transfer, and show that our methods for each framework are universally optimal. We also show that these methods are always justified from a formal perspective – i.e. our transfer method never performs much worse than a non-transfer method. An interesting result that falls out of this work is that while current practical transfer learning methods are considered to be parallel transfer methods, they are in fact sequential transfer methods in disguise. Finally, we further strengthen the classical universal optimality of our priors by showing that they are also optimal in a competitive setting. All the results and discussion in this chapter are for the squared loss function, and the case for arbitrary bounded loss function is handled in Sect. 5.1.

4.1 Solomonoff Induction and Bayesian Convergence Results

We first recall from Sect. 3.2 the set $\mathcal{V}$ of lower semi-computable semimeasures and the learning problem defined on this space. We consider as our task spaces enumerable subsets $\mathcal{V}' \subset \mathcal{V}$; and without loss of generality, we fix some $\mathcal{V}'$ for the sequel. Given some $x_{1:t}$, generated by some task $\mu \in \mathcal{V}'$, the prediction problem

is to predict the next letter a. When μ is known, the prediction is made according to $\mu(a|x_{1:t})$ (see Sect.5.1). When μ is not known, in the Bayesian setting, this prediction is made using a conditional Bayes Mixture $\mathbf{M}_W(a|x_{1:t})$ for a prior W over $\mathcal{V}'$:

Definition 4.1. *For any $x \in \mathcal{A}^*$ the Bayes mixture over $\mathcal{V}'$ is defined as follows:*

$$\mathbf{M}_W(x) := \sum_{\mu \in \mathcal{V}'} \mu_i(x) W(\mu) \text{ where } \forall \mu, W(\mu) \geq 0 \text{ and } \sum_{\mu \in \mathcal{V}'} W(\mu_i) \leq 1 \ . \quad (4.1)$$

The conditional probability, according to $\mathbf{M}_W$, of the next letter being a is now given by:

$$\mathbf{M}_W(a|x_{1:t}) := \frac{\mathbf{M}_W(x_{1:t}a)}{\mathbf{M}_W(x_{1:t})} \ . \quad (4.2)$$

■

For our purposes it is sufficient that the prior W satisfy the density inequality rather than the equality. So $\mathbf{M}_W$ is a weighted sum of the elements of $\mathcal{V}'$. As mentioned in Chap. 3, we wish to investigate transfer in the limit, and so **we will only consider lower semicomputable priors – and in the sequel all priors will be assumed to be such**. In this case, as $\mathcal{V}'$ is enumerable and each $\mu \in \mathcal{V}'$ is lower semicomputable, we can lower semicompute $\mathbf{M}_W$ by enumerating $\mathcal{V}$ and lower semicomputing μs in parallel. Hence, $\mathbf{M}_W \in \mathcal{V}$.

We will now state a well-known extraordinary convergence result for $\mathbf{M}_W(.|.)$ (Solomonoff, 1978; Hutter, 2003):

Theorem 4.1 (Solomonoff, Hutter)**.** *$\forall \mu \in \mathcal{V}'$:*

$$\sum_{t=0}^{\infty} \sum_{x_{1:t}} \mu(x_{1:t}) \left(\sum_{a \in \mathcal{A}} [\mathbf{M}_W(a|x_{1:t}) - \mu(a|x_{1:t})]^2 \right) \leq -\ln W(\mu) \ . \quad (4.3)$$

That is, for any target probability measure μ, the μ expected error of $\mathbf{M}_W$ goes to zero very rapidly as long as $-\ln W(\mu)$ finite. That is, as long as the target μ is not assigned 0 probability by W, we have:

- the expected number of times t that $|\mathbf{M}_W(a|x) - \mu(a|x)| > \epsilon$ is $\leq -\ln W(\mu)/\epsilon^2$, and
- the probability that the number of ϵ deviations $> -\ln W(\mu)/\epsilon^2 \delta$ is $< \delta$.

Hence by predicting using the conditional $\mathbf{M}_W(.|.)$, we are guaranteed rapid convergence to the target probability measure. Theorem 4.1 was first proved in Solomonoff, 1978 for $\mathcal{V}' = \mathcal{V}$ and $\mathcal{A} = \mathcal{B}$, and was then extended to arbitrary finite alphabets, $\mathcal{V}'$s and bounded loss functions (Hutter 2003; 2004). In Hutter, 2003 it was also

shown that Bayes mixtures are Pareto optimal, and that if $\mu \notin \mathcal{V}'$, but there is a $\rho \in \mathcal{V}'$ such that $\forall t \in \mathbb{N}$, the t^{th} order KL divergence between ρ and $\mu \leq k$, i.e. $\forall t \in \mathbb{N}$,

$$\sum_{x_{1:t}} \mu(x_{1:t}) \ln \left(\frac{\mu(x_{1:t})}{\rho(x_{1:t})} \right) \leq k \ .$$

then

$$\mathsf{Eb}_W(\mu) = -\ln W(\rho) + k \ .$$

That is, $\mathbf{M}_W$ will still converge to μ even if μ itself $\notin \mathcal{V}'$, but has an acceptable approximation in terms of the KL divergence.

For all the above reasons, we use Theorem 4.1 as the main tool to establish our own optimality results. To that end we define:

Definition 4.2. *For a prior W, the* **error bound** *under Theorem 4.1 is defined as*

$$\mathsf{Eb}_W(\mu) := -\ln W(\mu) \ .$$

A prior W is said to be **universally optimal** *in some class C if for all priors $W' \in C$, $\forall \mu \in \mathcal{V}'$:*

$$\mathsf{Eb}_W(\mu) \stackrel{+}{\leq} \mathsf{Eb}_{W'}(\mu) \ . \tag{4.4}$$

■

We end this section by looking at the Solomonoff-Levin prior :

Definition 4.3. *The* **Solomonoff-Levin prior** *is defined by:*

$$\xi_{\mathrm{SL}}(\mu) := 2^{-K(\mu)} \ .$$

■

ξ_{SL} is lower semicomputable as K is upper semicomputable. For this prior we have $\mathsf{Eb}_{\xi_{\mathrm{SL}}}(\mu) = K(\mu) \ln 2$. This is intuitively appealing because it shows the smaller the code for μ, the smaller the bound, which is a instantiation of Occam's razor. In addition, for any other lower semicomputable prior W, the error bound $-\ln W(\mu)$ is upper semicomputable, and $-\ln W / \ln 2$ satisfies the conditions for Lemma 3.2 (with $y = \varepsilon$ and $W(x)$ undefined if $x \notin \mathcal{V}'$), so:

$$K(\mu) \ln 2 \stackrel{+}{\leq} -\ln W(\mu) \ . \tag{4.5}$$

and therefore we have:

Theorem 4.2. *The Solomonoff-Levin prior is universally optimal in the class of lower semicomputable priors.*

Indeed, our universally optimal transfer learning priors will be transfer learning versions of $\xi_{\rm SL}$, and furthermore, we will use Theorem 4.2 to show that our transfer learning methods are always justified.

4.2 Universal Sequential Transfer Learning

In this section we will look at sequential transfer learning in the setting of Solomonoff Induction and derive transfer learning methods that are universally optimal and are, in a formal sense, always justified. We assume that we are given tasks $\varphi_1, \varphi_2, \cdots, \varphi_{m-1} \in \mathcal{V}$, as previously learned tasks. We should stress here that the transfer method we present here is representation agnostic – that is, we do not care about how these were learned and our method will simply try to do the best it can given these previous tasks. For instance each φ_i may be a weighted sum of elements of $\mathcal{V}'$ after having observed a finite sequence $x^{(i)}$ (Hutter, 2003, Sect. 2.4) or each φ_i may be given by the user. Let $\varphi := \langle \varphi_1, \varphi_2, \ldots, \varphi_{m-1} \rangle^{m-1}$. The aim of transfer learning is to use φ as prior knowledge when predicting for the m^{th} task with some unknown generating semimeasure $\mu \in \mathcal{V}'$. Given this, a transfer learning scheme is just a conditional prior over $\mathcal{V}'$, and it may or may not be based on a distance function. So,

Definition 4.4. *A* **transfer learning scheme** *is a lower semicomputable prior* $W(\mu_i|\varphi)$ *with*

$$\sum_{\mu_i \in \mathcal{V}'} W(\mu_i|\varphi) \leq 1 \ .$$

and $W(x|\varphi)$ *undefined for* $x \notin \mathcal{V}'$. *A* **symmetric distance** D **based transfer learning scheme** *is a transfer learning scheme* $W_D(\mu_i|\varphi)$ *with*

$$W_D(\mu_i|\varphi) := g(D(\mu_i, \varphi)) \ .$$

for a symmetric function $D : \mathcal{A}^* \times \mathcal{A}^* \to \mathbb{R}$ *and an arbitrary function* $g : \mathbb{R} \to [0, 1]$. ■

W_D is defined in terms of g because we do not want to put restrictions on how the distance function D may be used to induce a prior, or even what constraints D must satisfy other than being symmetric.

Definition 4.5. *Our* **universal transfer learning scheme** *is the prior*

$$\xi_{\rm TL}(\mu_i|\varphi) := 2^{-K(\mu_i|\varphi)} \ .$$

Our **TL distance based universal transfer learning scheme** *for Bayes mixtures*

over $\mathcal{V}'$ is the prior

$$\xi_{\mathrm{DTL}}(\mu_i|\varphi) := 2^{-E_1(\mu_i,\varphi)} \ .$$

■

For ξ_{DTL} we use E_1 instead of E_1^m because E_1 measures amount of information between the m^{th} task and previous $m-1$ tasks, which is what we want, whereas E_1^m measures amount of information between all possible disjoint groupings of tasks, and hence it measures more information than we are interested in. ξ_{DTL} is a prior since

$$\sum_{\mu_i\in\mathcal{V}'} 2^{-E_1(\mu_i,\varphi)} \leq \sum_{\mu_i\in\mathcal{V}'} 2^{-K(\mu_i|\varphi)} \leq 1 \ .$$

where the inequality holds for $2^{-K(\mu_i|\varphi)}$ because $K(\mu_i|\varphi)$, being lengths of programs, satisfies the Kraft inequality. As $E_1(\cdot,\varphi)$ and $K(\cdot|\varphi)$ are upper semicomputable, ξ_{DTL} and ξ_{TL} are lower semicomputable.

So in the Bayesian framework ξ_{DTL} automatically transfers the right amount of information from previous tasks to a potential new task by weighing it according to how related it is to older tasks. ξ_{TL} is less conceptually pleasing as $K(\mu_i|\varphi)$ is not a distance, and a goal of TL has been to define transfer learning scheme using TL distance functions. But as we see below, ξ_{TL} is actually more generally applicable for sequential transfer.

Theorem 4.3. *ξ_{TL} and ξ_{DTL} are universally optimal in the class of transfer learning schemes and distance based transfer learning schemes respectively.*

Proof. Let W be a transfer learning scheme, then

$$\mathsf{Eb}_{\xi_{\mathrm{TL}}}(\mu) = K(\mu|\varphi)\ln 2 \text{ and } \mathsf{Eb}_W(\mu) = -\ln W(\mu|\varphi) \ .$$

W is lower semicomputable, which implies $-\ln W$ is upper semicomputable; $-\ln W/\ln 2$, restricted to $\mathcal{V}'$, satisfies the requisite conditions for Lemma 3.2 with $y=\varphi$, and so

$$\mathsf{Eb}_{\xi_{\mathrm{TL}}}(\mu) \overset{+}{\leq} \mathsf{Eb}_W(\mu) \ .$$

Let W_D be a distance based transfer learning scheme. Then:

$$\mathsf{Eb}_{\xi_{\mathrm{DTL}}}(\mu) = E_1(\mu,\varphi)\ln 2 \text{ and } \mathsf{Eb}_{W_D}(\mu) = -\ln W_D(\mu|\varphi) \ .$$

$-\ln W_D$ is upper semicomputable as W_D is lower semicomputable; $-\ln W_D$ is symmetric, and restricted to $\mathcal{V}'$, $-\ln W_D/\ln 2$ satisfies the Kraft inequality condi-

tion in Def. 3.8; therefore $-\ln W_D/\ln 2 \in \mathcal{D}$. Now by Theorem 3.2

$$\mathsf{Eb}_{\xi_{\mathrm{DTL}}}(\mu) \stackrel{+}{\leq} \mathsf{Eb}_{W_D}(\mu) \ .$$

□

Note that for W_D the error bound is given by $-\ln W_D \ / \ln 2$ which is $\in \mathcal{D}$, and so whether D itself is admissible or not is irrelevant. This further justifies considering only admissible distances. So from the theorem and discussion above, our method formally solves the problem of sequential transfer. It is universally optimal, and it automatically determines how much information to transfer. Additionally, ξ_{TL} does not transfer information when the tasks are not related in the following sense. By (4.5), the non-transfer universally optimal prior is $2^{-K(.)}$, with error bound $K(\mu)\ln 2$. As $K(\mu|\varphi) \stackrel{+}{\leq} K(\mu)$ by Lemma 3.1, we have

Theorem 4.4. *ξ_{TL} is universally optimal in the class of non-transfer priors.*

The above implies, that, from a *formal perspective*, sequential transfer is always justified - i.e. it never hurts to transfer (see Sect. 4.3.3).

4.3 Universal Parallel Transfer Learning

Multitask learning methods used in practice are considered to be 'parallel transfer' methods where we learn m different tasks simultaneously and transfer across all the tasks as we learn them. In the following two sections, we will explore this type of transfer. There are two different possible interpretations of parallel transfer, which we term joint-parallel transfer and online-parallel transfer respectively. We will show that although current transfer methods are conceived of as implementing joint-parallel transfer is not really a transfer method, while the online-parallel transfer is a genuine transfer method. We also show in Sect. 4.3.3, that current transfer methods are in fact just sequential transfer methods, but in disguise.

4.3.1 Joint-parallel Transfer

In joint-parallel transfer we learn m related tasks in parallel. There are m generating semimeasures $\mu_1, \mu_2, \cdots, \mu_m \in \mathcal{V}$ generating sequences $x^{(1)}, x^{(2)}, \cdots, x^{(m)}$ respectively. At step t, μ_i generates the t^{th} bit of sequence $x^{(i)}$ in the usual way. To apply Theorem 4.1 in this scenario, we assume that our semimeasures are defined over an alphabet $\mathcal{A}_m$ of size $|\mathcal{A}|^m$, i.e. we use an m vector of $\mathcal{A}$ to represent each element of $\mathcal{A}_m$. So given a sequence x of elements of $\mathcal{A}_m$, i.e. $x \in \mathcal{A}_m^*$, $x^{(i)}$ will be the i^{th} components of vectors in x, for $1 \leq i \leq m$. A semimeasure over $\mathcal{A}_m$ is now defined as in Definition 3.2.

Definition 4.6. *The* **measure space $\mathcal{V}_m$ for joint-parallel transfer** *is now defined by:*

$$\mathcal{V}_m := \{\rho : \forall t \in \mathbb{N},\, \forall x_{1:t},\, \rho(x_{1:t}) = \prod_i \rho_i^m(x_{1:t}^{(i)}) \text{ where } \rho_i^m \in \mathcal{V}\} \ .$$

We denote the m different components of $\rho \in \mathcal{V}_m$ by ρ_i^m. ■

It is easy to see that $\mathcal{V}_m$ is enumerable: as we enumerate $\mathcal{V}$, we use the $\langle\rangle^m$ map to determine the elements of $\mathcal{V}$ that will be the components of a particular $\rho \in \mathcal{V}_m$. We will consider as our task spaces enumerable subsets $\mathcal{V}'_m$ of $\mathcal{V}_m$. As before we define:

Definition 4.7. *A* **joint-parallel transfer learning scheme** *W_m is a lower semi-computable prior over $\mathcal{V}'_m$ ($W_m(x)$ undefined for $x \notin \mathcal{V}'_m$):*

$$W_m(\rho) := W_m(\rho_{1,m}^m) \text{ with } \sum_{\rho_i \in \mathcal{V}'_m} W_m(\rho_i) \leq 1 \ .$$

A Bayes mixture is now given by:

$$\mathbf{M}_W^m(x) := \sum_{\rho_i \in \mathcal{V}'_m} \rho_i(x) W(\rho_i) \ .$$

■

Definition 4.8. *The* **universal joint-parallel transfer learning scheme** *is defined as the prior:*

$$\xi_{\mathrm{JPTL}}(\rho) := \xi_{\mathrm{JPTL}}(\rho_{1,m}^m) := 2^{-K(\rho_{1,m}^m)} \ .$$

■

Theorem 4.5. *ξ_{JPTL} is universally optimal in the class of joint-parallel transfer learning schemes.*

Proof. Let the generating semimeasure be $\mu = \mu_{1,m}^m$ and W_m be any joint-parallel transfer scheme. Then,

$$\mathsf{Eb}_{\xi_{\mathrm{JPTL}}}(\mu) = K(\mu_{1,m}^m) \ln 2 \text{ and } \mathsf{Eb}_{W_m}(\mu) = -\ln W_m(\mu_{1,m}^m) \ .$$

By Lemma 3.2, and reasoning similar to the first part in the proof of Theorem 4.3,

$$K(\mu_{1,m}^m) \ln 2 \overset{+}{\leq} -\ln W_m(\mu_{1,m}^m) \ .$$

Hence the prior ξ_{JPTL} is universally optimal. □

Indeed, $K(\rho^m_{1,m})$ is the measure of similarity that was used in Juba, 2006 to analyze multitask learning in a PAC setting (as mentioned in the Chaps. 1 and 2). However, ξ_{JPTL} is also the non-transfer Solomonoff-Levin prior for the space $\mathcal{V}_m$. Therefore, it seems that in this interpretation of multitask transfer, in contrast to sequential transfer, no actual transfer of information is occurring. Plain single task learning is taking place, but in a product space. The benefit of this is not clear from a formal perspective as $K(x) \stackrel{+}{\leq} K(x, y_{1,m-1})$, and so this type of 'transfer', in general, should not help learning.

4.3.2 Online-parallel Transfer

In this section, we consider an 'online' version of parallel transfer learning, where we have m different target tasks, μ_{a_k}, $1 \leq k \leq m$ with each task $\mu_{a_k} \in \mathcal{V}'$ as in sequential transfer learning. However, unlike in sequential transfer, we assume that at each step t the t^{th} letter of each sequence is generated simultaneously, and so we predict these sequences in parallel. When learning the i^{th} task, the idea now will now be to use a prior that is conditioned on the sequences generated by other $m-1$ tasks, and hence in this way we effect parallel transfer. Note that, since the information that the prior is being conditioned on is changing at each step, we will have a *sequence of time-varying priors* instead of a single static prior. At first glance this seems like quite a novel setup, that would require a tool more powerful than Theorem 4.1 to handle. But one key interesting result we uncover is that it is quite easy to show that in the general case Theorem 4.1 suffice. We will begin the analysis by defining the time-varying priors.

Definition 4.9. *For target task μ_{a_i}, let $x_t^{(-i)} := \langle x_{<t}^{(j)} : j \in \mathbb{N}_m, j \neq i \rangle$ where $x_{<t}^{(j)}$ is the sequence generated by target task μ_{a_j} in the previous $t-1$ steps*[1]*. We define a* **online-parallel transfer learning scheme** *for task μ_{a_i} as the sequence of lower semicomputable conditional priors $\{W_t^i\}$ over $\mathcal{V}'$, $t \in \mathbb{N}$ such that for each t:*

$$\sum_{\mu \in \mathcal{V}'} W_t^i(\mu | x_t^{(-i)}) \leq 1 \ .$$

We define the corresponding sequence of mixtures as $\{\mathbf{M}_t^i\}$ where for each $t, t \in \mathbb{N}$:

$$\mathbf{M}_t^i(x) := \sum_{\mu \in \mathcal{V}'} \mu(x) W_t^i(\mu | x_t^{(-i)}) \ .$$

■

[1] It is not necessary for what follows that each target measure μ_{a_j} emit their letters simultaneously. Indeed, we can define $x_t^{(-i)} := x_{<t_j}^{(j)}$ where t_j is the number of letters of the j^{th} sequence seen at the time of seeing t^{th} letter of μ_{a_i}; now everything in the sequel will still hold.

These TL schemes are different from typical priors in that they are allowed to change at each step. It is straightforward to show that these priors can behave in a way so that the mixture never converges.

Lemma 4.1. *There exists $\mathcal{A}$, $\mathcal{V}'$, $\mu' \in \mathcal{V}'$ and $\{W_t^i\}$ such that $W_t^1(\mu') > 0$ for all finite t but:*

$$\sum_{t=0}^{\infty} \sum_{x_{1:t}} \mu'(x_{1:t}) \left(\sum_{a \in \mathcal{A}} [\mathbf{M}_t^i(a|x_{1:t}) - \mu'(a|x_{1:t})]^2 \right) = \infty \ .$$

Proof. Let $\mathcal{A} = \mathcal{B}$ and set $\mathcal{V}' := \{\mu', \mu''\}$ such that $\mu'(0_{1:t}) = 1$ and $\mu''(x_{1:t}) = 2^{-t}$ for all t (i.e. μ' always predicts 0, i.e. corresponds to the the sequence $0000\ldots$). Let $W_t^1(\mu') = 2^{-t}$ and so the μ'-expected squared error at step t is given by:

$$\begin{aligned}
&\to \sum_{x_{1:t-1}} \mu'(x_{1:t-1}) \left(\sum_{a \in \{0,1\}} [\mathbf{M}_t^i(a|x_{1:t-1}) - \mu'(a|x_{1:t-1})]^2 \right) \\
&= \mu'(0_{1:t-1}) \left(\sum_{a \in \{0,1\}} [\mathbf{M}_t^i(a|0_{1:t-1}) - \mu'(a|0_{1:t-1})]^2 \right) \\
&= [\mathbf{M}_t^i(0|0_{1:t-1}) - \mu'(0|0_{1:t-1})]^2 + [\mathbf{M}_t^i(1|0_{1:t-1}) - \mu'(1|0_{1:t-1})]^2 \\
&> [\mathbf{M}_t^i(1|0_{1:t-1}) - \mu'(1|0_{1:t-1})]^2 = [\mathbf{M}_t^i(1|0_{1:t-1})]^2 = \left[\frac{\mathbf{M}_t^i(10_{1:t-1})}{\mathbf{M}_t^i(0_{1:t-1})} \right]^2 \\
&= \left[\frac{2^{-t}\mu'(10_{1:t-1}) + (1-2^{-t})\mu''(10_{1:t-1})}{2^{-t}\mu'(0_{1:t-1}) + (1-2^{-t})\mu''(0_{1:t-1})} \right]^2 = \left[\frac{(1-2^{-t})2^{-t}}{2^{-t} + (1-2^{-t})2^{-t+1}} \right]^2 \\
&= \left[\frac{1-2^{-t}}{2+2(1-2^{-t})} \right]^2
\end{aligned}$$

The last term $\to 1/16$ for $t \to \infty$. So for the total expected squared error upto step T we have:

$$> \sum_{t=1}^{T} \left[\frac{1-2^{-t}}{1+(1-2^{-t})} \right]^2$$

which goes to ∞ for $T \to \infty$, from which the claim follows immediately. □

Hence we can have online-transfer schemes that never converge, while never assigning 0 prior to the target measure in any finite step t. To fix this problem we need to restrict the types of sequences of priors that we should consider. For this purpose, we define the type of online-parallel transfer priors that are allowed:

Definition 4.10. *We define a* **admissible online-parallel (AOP) transfer learning scheme** *for task μ_{a_i} as the sequence of lower semicomputable conditional priors*

$\{W_t^i\}$ *over* $\mathcal{V}'$*, such that for each* $t \in \mathbb{N}$*:*

$$\sum_{\mu \in \mathcal{V}'} W_t^i(\mu | x_t^{(-i)}) \leq 1 \textit{ such that}$$
$$\textit{if } W_1^i(\mu | x_t^{(-i)}) > 0, \textit{ then } \inf_t W_t^i(\mu | x_t^{(-i)}) > 0\} \ .$$

We define the corresponding **online-parallel admissible mixtures** *as in Definition 4.9.* ■

That is, a sequence of priors is admissible if it does not assign 0 probability to any measure in the limit, a restriction which is eminently reasonable given Lemma 4.1. We can now define the the error bound of such schemes and a universal member as follows:

Definition 4.11. *We define the error bound for the sequence* $\{W_t^i\}$ *as:*

$$\mathsf{Eb}_{\{W_t^i\}}(\mu) := \sup_t \mathsf{Eb}_{W_t^i}(\mu) = \sup_t -\ln W_t^i(\mu | x_t^{(-i)}) \ .$$

An admissible online-parallel transfer scheme $\{P_t^i\}$ *is said to be* **universally optimal** *in some class* C *of AOP transfer schemes if* $\forall \{W_t^i\} \in C$*,* $\forall \mu \in \mathcal{V}'$*,*

$$\mathsf{Eb}_{\{P_t^i\}}(\mu) \stackrel{+}{\leq} \mathsf{Eb}_{\{W_t^i\}}(\mu) \ .$$

■

That the error bound definition is tight in a certain sense can also be shown quite easily:

Lemma 4.2. *There exists* $\mathcal{V}' \subset \mathcal{V}$*,* $\mu \in \mathcal{V}'$ *and* W*, such that the* μ*-expected squared error obtained in the first step is* $(-\ln W(\mu))/1.3$*.*

Proof. Set $\mathcal{A} := \mathcal{B}$ and $\mathcal{V}' = \{\mu_1, \mu_2\}$ such that μ_1 always predicts 0 and μ_2 always predicts 1. Let $\alpha = \mu_1(0) = 1, \beta = \mu_2(0) = 0$ and let $W(\mu_1) = w = 0.3$ and $W(\mu_2) = v = 1 - w$. Then, assuming μ_1 is the target measure, the expected squared error at $t = 1$ becomes

$$\begin{aligned}
&\to [\alpha - (w.\alpha + v\beta)]^2 + [1 - \alpha - (w(1-\alpha) + v(1-\beta))]^2 \\
&= [\alpha - w.\alpha + (1-w)\beta)]^2 + [1 - \alpha - w + w\alpha + v - v\beta]^2 \\
&= [(1-w)(\alpha - \beta)]^2 + [v - \alpha + w\alpha + v - v\beta]^2 \\
&= [(1-w)(\alpha - \beta)]^2 + [(1-w)(\alpha - \beta)]^2 \\
&= 2(\alpha - \beta)^2(1-w)^2 = 2(1-w)^2 = 0.98 \ .
\end{aligned}$$

while $-\ln w = 1.204$ and $1.204/0.98 = 1.3$.

□

Hence, for the $\mathcal{V}'$ and W in the statement of the theorem, with $W_1^i(\mu|x_t^{(-i)}) = W$, and then $W_t^i(\mu_1|x_t^{(-i)}) = 1$ for all $t > 1$ (i.e. all the prior is put on the target measure), the final total expected squared error cannot be much better than the Theorem 4.1 error bound $\sup \mathsf{Eb}_{W_t^i}(\mu) = -\ln W_1^i(\mu|x_t^{(-i)})$. So this shows that the definition of $\mathsf{Eb}_{\{W_t^i\}}$ is not unreasonable. So now, as before we can define universal AOP transfer learning method:

Definition 4.12. *Our* **universal online-parallel transfer learning scheme** *is the sequence of priors* $\{\xi_{\mathrm{OPTL}_t}^{\ i}\}$ *where*

$$\xi_{\mathrm{OPTL}_t}^{\ i}(\mu|x_t^{(-i)}) := 2^{-K(\mu|x_t^{(-i)})} \ .$$

■

Theorem 4.6. $\{\xi_{\mathrm{OPTL}_t}^{\ i}\}$ *is universally optimal in the class of AOP transfer learning schemes.*

Proof. Let $\{W_t^i\}$ be a AOP transfer learning scheme. Now,

$$\mathsf{Eb}_{\xi_{\mathrm{OPTL}_t}^{\ i}}(\mu) = K(\mu|x_t^{(-i)}) \ln 2$$

and for each t by Lemma 3.2, and reasoning similar to the first part in the proof of Theorem 4.3,

$$\mathsf{Eb}_{\xi_{\mathrm{OPTL}_t}^{\ i}}(\mu) \stackrel{+}{\leq} -\ln W_t^i(\mu|x_t^{(-i)}) = \mathsf{Eb}_{W_t^i}(\mu) \ .$$

and hence

$$\mathsf{Eb}_{\{\xi_{\mathrm{OPTL}_t}^{\ i}\}}(\mu) = \sup_t \mathsf{Eb}_{\xi_{\mathrm{OPTL}_t}^{\ i}}(\mu) \stackrel{+}{\leq} \sup_t -\ln W_t^i(\mu|x_t^{(-i)}) = \mathsf{Eb}_{\{W_t^i\}}(\mu) \ .$$

□

Since any non-transfer prior W corresponds to the AOP $\{W_t^i\}$, with $W_t^i = W$ for all t, we have (unlike the joint-parallel universal transfer prior):

Theorem 4.7. ξ_{OPTL} *is universally optimal in the class of non-transfer priors.*

Finally, just as in Sect. 4.2, we can also define distance based AOP transfer schemes and its universal element:

Definition 4.13. *We define a* **sequence of symmetric distances $\{D_t^i\}$ based AOP transfer learning scheme** *for task μ_{a_i} as the sequence of conditional priors $\{W_{D_t^i}\}$ over $\mathcal{V}'$ such that for each $t \in \mathbb{N}$:*

$$\sum_{\mu \in \mathcal{V}'} W_{D_t^i}(\mu | x_t^{(-i)}) \leq 1 \text{ such that}$$

$$\text{if } W_{D_1^i}(\mu | x_t^{(-i)}) > 0, \text{ then } \inf_t W_{D_t^i}(\mu | x_t^{(-i)}) > 0 \ .$$

where $W_{D_t^i}(\mu | x_t^{(-i)}) := g(D_t^i(\mu, x_t^{(-i)}))$ for a symmetric function $D_t^i : \mathcal{A}^* \times \mathcal{A}^* \to \mathbb{R}$ and $g : \mathbb{R} \to [0, 1]$. We define the corresponding **TL distance based universal AOP transfer learning scheme** *by the sequence of priors $\{\xi_{\mathrm{OPDTL}_t^i}\}$ where*

$$\xi_{\mathrm{OPDTL}_t^i}(\mu | x_t^{(-i)}) := 2^{-E1(\mu, x_t^{(-i)})} \ . \tag{4.6}$$

■

Theorem 4.8. *$\{\xi_{\mathrm{OPDTL}_t^i}\}$ is universally optimal in the class of symmetric distance based AOP transfer learning schemes.*

Proof. Let $\{W_{D_t^i}\}$ be a symmetric distance based AOP transfer learning scheme. Then:

$$\mathsf{Eb}_{W_{D_t^i}}(\mu) := -\ln W_{D_t^i}(\mu) \ .$$

$-\ln W_{D_t^i}(\mu)/\ln 2$ is upper semicomputable as $W_{D_t^i}$ is lower semicomputable; $-\ln W_{D_t^i}$ is symmetric, and restricted to $\mathcal{V}'$, $-\ln W_{D_t^i}/\ln 2$ satisfies the Kraft inequality condition in Definition 3.8; therefore $-\ln W_{D_t^i}/\ln 2 \in \mathcal{D}$. Now:

$$\mathsf{Eb}_{\xi_{\mathrm{OPDTL}_t^i}}(\mu) = E_1(\mu, x_t^{(-i)}) \ln 2 \ .$$

and for each t by Theorem 3.2:

$$\mathsf{Eb}_{\xi_{\mathrm{OPDTL}_t^i}}(\mu) \stackrel{+}{\leq} \mathsf{Eb}_{W_{D_t^i}}(\mu) \ .$$

and so:

$$\mathsf{Eb}_{\{\xi_{\mathrm{OPDTL}_t^i}\}}(\mu) = \sup_t \mathsf{Eb}_{\xi_{\mathrm{OPDTL}_t^i}}(\mu) \leq \sup_t -\ln W_{D_t^i}(\mu | x_t^{(-i)}) = \mathsf{Eb}_{\{W_{D_t^i}\}}(\mu)$$

□

4.3.3 Parallel Transfer in Practice

In the majority of multitask learning methods used in practice, given m tasks, each $x^{(i)}$ corresponds to training samples for task i. In a Bayesian setting, for each

task i, $x^{(j)}, j \neq i$ now function as prior knowledge, and we have priors of the form : $W(\mu|x^{(j)}, 1 \leq j \leq m, j \neq i)$. So current multitask learning methods seem to be performing m sequential transfers in parallel. Note that this is different from online-parallel transfer because the $x^{(i)}$ are static rather than being generated online in parallel. It has been observed that transferring from unrelated tasks hurts generalization (Caruana, 1997), which, given Theorem 4.4, seems to contradict the above conclusion. Nonetheless, our own empirical investigations in Chap. 6 and in Mahmud & Ray, 2007 lead us to believe that this is not because of parallel transfer but use of improper algorithms.

4.4 Competitive Optimality of the Universal Priors

In the universal optimality results of the Kolmogorov complexity based transfer and non-transfer based schemes described above, the inequalities hold upto a constant that depends on the complexity $K(W)$ of the prior W that the universal priors are competing against. In this competitive instance, we can actually define a modified version of our universal priors such that this constant is now independent of W and depends only on U, the reference universal Turing machine. First, we need the following extension of Lemma 3.2:

Lemma 4.3. *For any partial, non-negative, upper semicomputable function* $f : \mathcal{A}^* \times \mathcal{A}^* \to \mathbb{R}$ *and* $y \in \mathcal{A}^*$, *if*

$$\sum_x 2^{-f(x,y)} \leq 1$$ *(taking* $f(x,y) = \infty$ *when it is undefined)*

then we have,

$$K(x|y, f) \overset{+}{\leq} f(x, y) \ .$$

where the constant in $\overset{+}{\leq}$ *depends only on the reference universal Turing machine* U *and is small (Li & Vitanyi, 1997, Chap. 4).*

Proof. (Sketch) Define:

$$\mathbf{m}_f(x|y) := \sum_{g \in \mathcal{V}_\mathcal{D}} g(x|y) 2^{-K(g|f)} \ .$$

where $\mathcal{V}_\mathcal{D}$ is the enumerable set of conditional *discrete* lower semicomputable semi-measures – that is each $g \in \mathcal{V}_\mathcal{D}$, for each $y \in \mathcal{A}^*$, satisfies:

$$\sum_{x \in \mathcal{A}^*} g(x|y) \leq 1 \ .$$

Note that since f is upper semicomputable, and the elements of $\mathcal{V}_{\mathcal{D}}$ are lower semicomputable, there is a function $g_f \in \mathcal{V}_{\mathcal{D}}$ such that $g_f(x|y) = 2^{-f(x,y)}$. Now we have, by definition of $\mathbf{m}_f$:

$$-\log \mathbf{m}_f(x|y) \leq -\log g_f(x|y) + K(g_f|f) = -\log g_f(x|y) + O(1) \ .$$

then by using essentially the same proof as in Li & Vitanyi, 1997, Lemma 4.3.3., we have

$$K(x|y,f) \stackrel{+}{=} -\log \mathbf{m}_f(x|y) \leq -\log g_f(x|y) + O(1) = f(x,y) + O(1) \ .$$

□

We also need the definition of a conditional version of E_1:

Definition 4.14. *Define the* **Cognitive Distance conditioned on** $z \in \mathcal{A}^*$ *for all* $x, y \in \mathcal{A}^*$ *to be:*

$$E_1(x,y|z) := \max\{K(x|y,z), K(y|x,z)\}$$

■

We can now prove the following conditional version of Theorem 3.2:

Lemma 4.4. $\forall D \in \mathcal{D}, \forall x, y \in \mathcal{A}^*$

$$E_1(x,y|D) \stackrel{+}{\leq} D(x,y) \ . \tag{4.7}$$

where the constant in the inequality depends only on the reference Universal Turing machine U.

Proof. Use definition of $E1(.,.|D)$ and Lemma 4.3 with $f = D$ (D satisfies the requisite conditions due to its admissibility). □

Now we define competitively optimal (CO) version of each of the universally optimal prior we have defined, which are same as before but now conditioned on the prior W it is competing against:

Definition 4.15. *Fix any prior* W. *Then the* **CO non-transfer universal prior** *over* $\mathcal{V}'$ *is now defined by:*

$$\xi(\mu|W) := 2^{-K(\mu|W)} \ .$$

Given any sequential transfer scheme W *and distance based sequential transfer scheme* W_D *and previous tasks* φ, *the* **CO sequential transfer and symmetric**

distance based sequential universal transfer schemes *over $\mathcal{V}$ are now defined, respectively, by:*

$$\xi_{\mathrm{TL}}(\mu|\varphi, W) := 2^{-K(\mu|\varphi,W)}, \;\; \xi_{\mathrm{DTL}}(\mu|\varphi, W_D) := 2^{-E1(\mu,\varphi|W_D)} \;\; .$$

Given any AOP transfer scheme $\{W_t^i\}$ and distance based sequential transfer scheme $\{W_D{}_t^i\}$ and sequences $x_t^{(-i)}$, the **CO AOP and AOP symmetric distance-based universal transfer schemes** *over $\mathcal{V}$ are now defined, respectively, by $\{\xi_{\mathrm{OPTL}}{}_t^i(.|W_t^i)\}$ and $\{\xi_{\mathrm{OPDTL}}{}_t^i(.|W_D{}_t^i)\}$ where:*

$$\xi_{\mathrm{OPTL}}{}_t^i(\mu|x_t^{(-i)}, W_t^i) := 2^{-K(\mu|x_t^{(-i)},W_t^i)}$$
$$\xi_{\mathrm{OPDTL}}{}_t^i(\mu|x_t^{(-i)}, W_D{}_t^i) := 2^{-E1(\mu,x_t^{(-i)}|W_D{}_t^i)} \;\; .$$

■

We now have the following:

Theorem 4.9. *1. For any prior W and $\mu \in \mathcal{V}$, the following holds*

$$\mathsf{Eb}_{\xi(.|W)}(\mu) \stackrel{+}{\leq} \mathsf{Eb}_W(\mu) \;\; .$$

where the constant in the inequality depends only on the fixed universal Turing machine U.

2. *Given any sequential transfer scheme W and distance based sequential transfer scheme W_D and previous tasks φ, the following holds*

$$\mathsf{Eb}_{\xi_{\mathrm{TL}}(.|W)}(\mu) \stackrel{+}{\leq} \mathsf{Eb}_W(\mu), \;\; \mathsf{Eb}_{\xi_{\mathrm{DTL}}(.|W_D)}(\mu) \stackrel{+}{\leq} \mathsf{Eb}_{W_D}(\mu) \;\; .$$

where the constants in the inequalities depend only on the fixed universal Turing machine U.

3. *Given any AOP transfer scheme $\{W_t^i\}$ and distance based sequential transfer scheme $\{W_D{}_t^i\}$ and sequences $x_t^{(-i)}$, the following holds*

$$\mathsf{Eb}_{\{\xi_{\mathrm{OPTL}}{}_t^i(.|W_t^i)\}}(\mu) \stackrel{+}{\leq} \mathsf{Eb}_{\{W_t^i\}}(\mu), \;\; \mathsf{Eb}_{\{\xi_{\mathrm{OPDTL}}{}_t^i(.|W_D{}_t^i)\}}(\mu) \stackrel{+}{\leq} \mathsf{Eb}_{\{W_D{}_t^i\}}(\mu) \;\; .$$

where the constants in the inequalities depend only on the fixed universal Turing machine U.

Proof. The theorem follows from the same methods as Theorems 4.2, 4.3, 4.4, 4.6, 4.8 but using Lemma 4.3 instead of Lemma 3.2 and Lemma 4.4 instead of Theorem 3.2. □

Note. This actually induces a different interpretation of exactly what the universal methods are achieving. Now instead of being optimal, these methods are viewed as ones that are powerful 'base' methods that may be used any time, and are also ways to enhance any other high complexity method W that we may choose to use for a particular problem. So even if our prior knowledge W is wrong, the universal priors are guaranteed to not do too badly.

4.5 Discussion

In this chapter we defined our Bayesian transfer learning methods and established their universal optimality. We analyzed both sequential and parallel transfer learning, and showed that practical transfer learning methods are in fact sequential transfer learning methods. The methods we derived automatically transfer the right amount of information and are never much worse than any non-transfer learning scheme. So our optimal Bayesian priors *formally* solves the problem of when to transfer information, when not to, and how much information to transfer. We also introduced a different notion of optimality in the competitive setting wherein our methods are powerful 'base' transfer algorithms that can be used in applications where we do not know which transfer method to use; and at the same time can also be used to improve any high complexity transfer method that we suspect is useful in the given application.

Future work will involve deriving practical versions of these methods using approximations to the distance functions (as mentioned at the end of Chap. 3) and Markov Chain Monte Carlo methods to sample from the Bayes mixtures. We have already done a battery of successful experiments in this setting in Chap. 6, but we believe it is possible to construct much more sophisticated and powerful approximations by constraining ourselves to specific groups of machine learning domains. In this case we attain tractability by exploiting the peculiarities unique to the specific group of domains.

Chapter 5

Universal Transfer Learning: Extensions

In this chapter we will extend the transfer paradigm developed in the previous two chapters. We will first extend the results in Chap. 4 to arbitrary bounded loss functions. Then we will look at how the transfer methods may be applied in the Prediction with Expert Advice setting. We will extend our result to the reinforcement learning setting. We will then end this chapter with a look at Kolmogorov complexity of functions to head off one possible objection to the theoretical framework developed in the previous two chapters.

5.1 Transfer Convergence Rates for Arbitrary Bounded Loss

We now use the convergence results in Hutter, 2003 to extend the Bayesian transfer learning results to arbitrary bounded loss functions.

Definition 5.1. *A* **bounded loss function** $\ell : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}$ *is a function such that if the observed letter is* a *and the letter predicted by some predictor is* b*, then the loss suffered by the predictor is given by* $\ell(a, b)$*.* ■

Definition 5.2. *The* **prediction scheme** Λ_π **defined by a measure** $\pi \in \mathcal{V}$ **at step** t *is given by:*

$$\Lambda(x_{<t}) := \arg\min_{y \in \mathcal{B}} \sum_{a \in \mathcal{A}} l(a, y)\pi(a|x_{<t}) \ .$$

For target measure $\mu \in \mathcal{V}$*,* **for** Λ_π **the** μ**-expected loss at step** t **is given by** *:*

$$\ell_t^{\Lambda_\pi}(x_{<t}) := \sum_{a \in \mathcal{A}} l(a, \Lambda_\pi(x_{<t}))\mu(a|x_{<t}) \ .$$

And **for** Λ_π **the total** μ **expected loss in** n **steps** *is given by:*

$$\ell_{1:n}^{\Lambda_\pi} := \sum_{t=1}^{n} \sum_{x_{<t}} \mu(x_{<t})\ell_t^{\Lambda_\pi}(x_{<t}) \ .$$

■

The above is a fairly standard definition of loss functionals in a Bayesian setup. The equivalent of Theorem 4.1 in this setup is as follows.

Theorem 5.1 (Hutter)**.** *For any mixture $\mathbf{M}_W$, the following holds true for the loss bound:*

$$0 \leq \ell_{1:n}^{\Lambda_{\mathbf{M}_W}} - \ell_{1:n}^{\Lambda_\mu} \leq -\ln W(\mu) + 2\sqrt{\ell_{1:n}^{\Lambda_\mu}(-\ln W(\mu))} \ .$$

So we may define the loss bounds and universally optimal prior for each of the transfer learning priors we considered in chapter 4.

Definition 5.3. *The ℓ-**loss bound** for the mixture $\mathbf{M}_W$ corresponding to the sequential transfer learning scheme W (of any type) is defined as:*

$$\mathsf{Lb}_W(\mu) := -\ln W(\mu) + 2\sqrt{\ell_{1:n}^{\Lambda_\mu}(-\ln W(\mu))} \ .$$

*A prior W in any class C of sequential transfer learning scheme is said to be ℓ-**universally optimal** if for all $P \in C$*

$$\mathsf{Lb}_W(\mu) \overset{+}{\leq} \mathsf{Lb}_P(\mu) + 2\sqrt{c\ell_{1:n}^{\Lambda_\mu}} \ . \tag{5.1}$$

where the constant c is equal to the constant in $\overset{+}{\leq}$.

*The ℓ-**loss bound** for the mixture $\mathbf{M}_{\{W_t^i\}}$ corresponding to the AOP transfer learning scheme $\{W_t^i\}$ (of any type) is defined as:*

$$\mathsf{Lb}_{\{W_t^i\}}(\mu) := \sup_t \left[-\ln W_t^i(\mu|x_t^{(-i)}) + 2\sqrt{\ell_{1:n}^{\Lambda_\mu}(-\ln W(\mu|x_t^{(-i)}))} \right] \ .$$

*A prior $\{W_t^i\}$ in any class C of AOP transfer learning scheme is said to be ℓ-**universally optimal** if for all $\{P_t^i\} \in C$*

$$\mathsf{Lb}_{\{W_t^i\}}(\mu) \overset{+}{\leq} \mathsf{Lb}_{\{P_t^i\}}(\mu) + 2\sqrt{c\ell_{1:n}^{\Lambda_\mu}} \ . \tag{5.2}$$

where the constant c is equal to the constant in $\overset{+}{\leq}$. ■

And now we get:

Theorem 5.2. *We have the following:*

1. *ξ is ℓ-universally optimal in the class of all lower semicomputable priors W.*
2. *ξ_{TL} and ξ_{DTL} are ℓ-universally optimal in the class of sequential and symmetric-distance based sequential transfer learning schemes.*
3. *ξ_{OPTL} and ξ_{OPDTL} are ℓ-universally optimal in the class of lower semi-computable AOP transfer and and symmetric-distance based AOP transfer schemes.*

The above also holds in the competitive setting of Sect. 4.4 where the constant in (5.2) now depend only on the reference universal Turing machine U.

Proof. In the non-competitive setting, the theorem follows from the same methods as Theorems 4.2, 4.3, 4.4, 4.6, 4.8. In the competitive setting the theorem also holds with constant of inequality depending on U by using Lemma 4.3 instead of Lemma 3.2 in the theorems just listed. □

5.2 Transfer in the PEA Setting

A very interesting class of learning algorithms are called the Prediction with Expert Advice (PEA) algorithms which are a generalization of the Bayesian setting (see Cesa-Bianchi & Lugosi, 2006 for a comprehensive introduction). In this section we briefly describe a prominent algorithm, the Aggregating Algorithm (AA) (Vovk, 2001), give its convergence bound and then show how easily our transfer prior ξ_{TL} applies here. In future work we will look at if and how the idea in this section may be applied in other PEA algorithms and in transfer settings other than the sequential transfer case.

The basic setup in which AA/Learner operates is defined using the following elements:

Definition 5.4. PEA Setup *Let Θ be a set of experts, Γ a set of decisions, and a loss function $\ell : \Gamma \times \Gamma \to \mathbb{R}^+$. Let Lea be the learner and let Env be the Environment with which Lea plays the game in algorithm 5.1 for T steps.* ■

Algorithm 5.1 The PEA Setting

1: **for** t = 1 to T **do**
2: Env chooses a decision $\gamma^t_{Env} \in \Gamma$.
3: Each $\theta \in \Theta$ chooses a decision $\gamma^t_\theta \in \Gamma$.
4: Lea chooses a decision $\gamma^t_{Lea} \in \Gamma$ without knowing γ_{Env}.
5: Each $\theta \in \Theta$ suffers loss $\ell(\gamma^t_{Env}, \gamma^t_\theta)$.
6: Lea suffers loss $\ell(\gamma^t_{Env}, \gamma^t_{Lea})$.
7: **end for**

The aim of the Learner is to suffer loss as close as possible to

$$\min_{\theta\in\Theta} L_T(\theta) \text{ where } L_T(\theta) := \sum_{t=1}^{T} \ell(\gamma^t_{Env}, \gamma^t_\theta) \ .$$

i.e. do as well as the best expert. Given prior $P_0 := W$, $\beta = e^{-\eta}, \eta > 1$ where η is the learning rate, the learner Aggregating Algorithm operates as in algorithm

Algorithm 5.2 The Aggregating Algorithm

1: **for** t = 1 to T **do**

2: AA chooses decision:

$$\gamma_{AA}^t := \arg\min_{\gamma_b \in \Gamma} \log_\beta \int_\Theta \beta^{\ell(\gamma_b, \gamma_\theta^t)} P_{t-1}(d\theta)$$

3: Receive all losses for both itself and Θ.

4: Set the weights for the $t + 1^{th}$ step as:

$$P_t(d\theta) = \beta^{\ell(\gamma_{Env}^t, \gamma_\theta^t)} P_{t-1}(d\theta)$$

5: **end for**

5.2 (we are ignoring some subtelties, that are not relevant to us, regarding how the decision is chosen by AA – see Vovk, 2001 for details).

Now we state the convergence bound

Theorem 5.3 (Vovk)**.** *The following holds true:*

$$L_T(AA(\eta, P_0)) \leq \log_\beta \int_\Theta \beta^{L_T(\theta)} P_0(d\theta) \ .$$

for countable set of experts, the following is true:

$$L_T(AA(\eta, P_0)) \leq \log_\beta \sum_i \beta^{L_T(\theta)} P_0(\theta_i) \ .$$

Theorem 5.4. *Let $\varphi := \{\varphi_1, \varphi_2, \cdots, \varphi_m\}$ be a set of m previously learned tasks (as long as φ_i are strings, we do not care how they represent previously learned tasks – e.g. each φ_i may be a program computing measure P_{T_i} over Θ, learned from m previous games of T_i steps each). Then $\xi_{\mathrm{TL}}(\cdot|\varphi)$ is universally optimal – i.e. for any transfer learning prior $W(\cdot|\varphi)$, the following holds:*

$$\log_\beta \sum_i \beta^{L_T(\theta_i)} \xi_{\mathrm{TL}}(\theta_i|\varphi) \stackrel{+}{\leq} \log_\beta \sum_i \beta^{L_T(\theta_i)} W(\theta_i|\varphi) \ .$$

The competitively optimal version of the above, where the constant depends on reference universal Turing machine U also holds.

Proof. Recall that $\xi_{\mathrm{TL}}(\cdot|\varphi) := 2^{-K(\cdot|\varphi)}$ and note that $\log_\beta x = \frac{\log(x)}{\log \beta} = -\frac{\log(x)}{\eta \log e}$.

Setting $k := \frac{1}{\eta \log(e)}$ we get,

$$\begin{aligned}
&\Rightarrow \quad \log_\beta \sum_i \beta^{L_T(\theta_i)} 2^{-K(\theta_i|\varphi)} \\
&= \quad -k \log \sum_i \beta^{L_T(\theta_i)} 2^{-K(\theta_i|\varphi)} \\
&\leq \quad -k \log \sum_i \beta^{L_T(\theta_i)} e^{-c} W(\theta_i|\varphi) \\
&= \quad kc \log(e) - k \log \sum_i \beta^{L_T(\theta_i)} W(\theta_i|\varphi) \\
&= \quad \frac{c}{\eta} + \log_\beta \sum_i \beta^{L_T(\theta_i)} W(\theta_i|\varphi)
\end{aligned}$$

Where c is the constant from the $\overset{+}{\leq}$ inequality in Lemma 3.2. The competitively optimal version of the above, where the constant depends on reference universal Turing machine U also holds by using Lemma 4.3 instead of Lemma 3.2. So this proves the theorem. □

5.3 Transfer in Bayesian Reinforcement Learning Agents

In the previous sections we have primarily considered transfer learning in sequence prediction problems. However transfer learning can also be applied in artificial agents interacting with an environment – see Sect. 2.1.2 for a discussion of such methods. In this section, we apply the ideas of task similarity and optimal transfer learning prior to solve transfer problems in a very general model of Bayesian reinforcement learning agent developed in Hutter, 2004. In the following, we do not consider discounted version of reinforcement learning as the development is quite similar and does not add anything from the transfer learning perspective we are pursuing. We refer the reader to Hutter, 2004 for the full details. In what follows, for reasons of clarity we will use x_t etc. to denote the t^{th} letter in some sequence instead of a string.

5.3.1 The AI Agent Model

The **AI agent model** we consider consists of a set of (*agent*, *environment*) pairs $\{p, q\}$ where each p and q are partially recursive **chronological Turing machines**. That is, $p : \mathcal{Y}^* \to \mathcal{X}^*$ and $q : \mathcal{X}^* \to \mathcal{Y}^*$. The model runs from step $t = 1$ to some step $T \in \mathbb{N} \cup \{\infty\}$ starting from some initial input x_0. Afterward, at each step t (discrete time) the agent generates output $p(yx_{<t}) := y_t$ and the environment generates input $q(yx_{<t}y_t) := x_t$ (where $yx_{<t} := x_0y_1x_1y_2x_2 \cdots y_{t-1}x_{t-1}$).

We assume that $\mathcal{X} \subset \mathcal{S} \times \mathbb{R}$ – i.e. each output x_t is broken up into two parts the *state* s_t at time t, and the *reward* r_t at time t. The aim of the agent is to, at each point in time t, generate the output y_t such that the *value function* is maximized. Before defining the value function we will weaken our assumption about the environment: we will assume that instead of a single function q, it is described by a lower semicomputable chronological semimeasure μ:

Definition 5.5. *A* **chronological semimeasure** μ *is defined as a function that satisfies* $\forall y_{1:t} \in \mathcal{Y}^t$ *:*

$$\sum_{a \in \mathcal{X}} \mu(yx_{<t}y_t a) \leq \mu(yx_{<t}) \text{ and } \mu(\varepsilon) \leq 1$$

■

That is, we assume that at each step t, the next output is drawn according to $\mu(x_t|yx_{<t}y_t)$ where, as usual,

$$\mu(x_t|yx_{<t}y_t) := \frac{\mu(yx_{1:t})}{\mu(yx_{<t})}$$

We set as our task space enumerable subsets $\mathcal{M}'$ of the set of chronological semimeasures $\mathcal{M}$ will be our task space in the reinforcement learning setting. It can be shown that $\mathcal{M}$ is enumerable by using a method very similar to the one used in Zvonkin & Levin, 1970 to show that $\mathcal{V}$ is enumerable (Hutter, 2004, Sect. 5.10). Without loss of generality we set a $\mathcal{M}'$ as our task space.

Definition 5.6. *The* **value function** *for environment* $\mu \in \mathcal{M}'$ *for policy/agent* p *is now defined as :*

$$V_{km}^{p\mu}(yx_{<k}) = p_{y_k} \sum_{x_k} p_{y_{k+1}} \sum_{x_{k+1}} \cdots p_{y_m} \sum_{x_m} (r_k + r_{k+1} + \cdots + r_m) \mu(x_{k:m}|yx_{<k}p_{y_k:y_m}) \ .$$

where p_{y_k} *is the action chosen by the agent on the* k^{th} *step based on the output-input sequence* $yx_{<k}$ *seen so far.*

We also use p^μ *to denote the optimal policy with respect to enviornment* μ*.*

■

Hence $V_{km}^{p\mu}(yx_{<k})$ is the expected value w.r.t. probability distribution μ of the actions of the agent when the environment is μ. The subscript km in $V_{km}^{p\mu}(yx_{<k})$ is the expectation for steps k to m, where m is the lifetime of the agent.

The agent would like to select a policy that maximizes the expected reward or value function. This maximum value is:

$$V_{km}^{*\mu}(yx_{<k}) = \max_{y_k} \sum_{x_k} \max_{y_{k+1}} \sum_{x_{k+1}} \cdots \max_{y_m} \sum_{x_m} [(r_k + r_{k+1} + \cdots + r_m) \mu(x_{k:m}|yx_{<k}y_{k:m})] \ .$$

and the corresponding optimal action is:

$$y_k^{\mu} = \arg\max_{y_k} \sum_{x_k} \max_{y_{k+1}} \sum_{x_{k+1}} \cdots \max_{y_m} \sum_{x_m} [(r_k + r_{k+1} + \cdots + r_m) \mu(x_{k:m}|yx_{<k}y_{k:m})]$$

Since μ is unknown, the idea behind Hutter's *universal AI agent* Hutter, 2004 is to use a mixture $\mathbf{M}_W$ to approximate μ and hence get the AI-$\mathbf{M}$ agent:

$$\mathbf{M}_W(yx_{<t}) := \sum_{\mu \in \mathcal{M}'} \mu(yx_{<t}) W(\mu) \text{ and } \sum_{\mu \in \mathcal{M}'} W(\mu) \leq 1$$

where W is a lower semicomputable prior. Please note that the AI-$\mathbf{M}$ is actually called the AIξ agent in Hutter's work, but as a we use ξ to denote priors, we use AI-$\mathbf{M}$ to avoid confusion. This agent which chooses the following action $y_k^{\mathbf{M}_W}$ at step k:

$$V_{km}^{*\mathbf{M}_W}(yx_{<k}) = \max_{y_k} \sum_{x_k} \max_{y_{k+1}} \sum_{x_{k+1}} \cdots \max_{y_m} \sum_{x_m} [(r_k + r_{k+1} + \cdots + r_m) \mathbf{M}_W(yx_{<k}y\underline{x}_{k:m})]$$

$$y_k^{\mathbf{M}_W} = \arg\max_{y_k} \sum_{x_k} \max_{y_{k+1}} \sum_{x_{k+1}} \cdots \max_{y_m} \sum_{x_m} [(r_k + r_{k+1} + \cdots + r_m) \mathbf{M}_W(yx_{<k}y\underline{x}_{k:m})]$$

5.3.2 Convergence Bounds for the AI Agent Model

The first convergence result is given below; this result parallels Theorem 4.1 and has similar implications:

Theorem 5.5 (Hutter)**.** $\forall \mu \in \mathcal{V}', y_{1:t} \in \mathcal{Y}^t, t \in \mathbb{N}$*:*

$$\sum_{t=0}^{\infty} \sum_{yx_{1:t}} \mu(yx_{1:t}) \left(\sum_{a \in \mathcal{X}} [\mathbf{M}_W(a|yx_{1:t}) - \mu(a|yx_{1:t})]^2 \right) \leq -\ln W(\mu) \ . \quad (5.1)$$

And the following now holds trivially:

Theorem 5.6. *Theorems 4.2, 4.3, 4.4, 4.6, 4.8 and 4.9 now hold but with lower semicomputable semimeasure replaced by lower semicomputable chronological semimeasure.*

So this may suggest that the agent using $V^{*\mathbf{M}_W}$ should converge to the $V^{*\mu}$ as $k \to \infty$ where μ is the chronological semimeasure defining the environment. This is correct but only in the following sense:

Theorem 5.7 (Hutter)**.** *If $\forall \mu \in \mathcal{M}'$ there exists a sequence of policies p_m such that*

$$V_{1m}^{*\mu}(yx_{<k}) - V_{1m}^{p_m \mu}(yx_{<k}) < \Delta(m)$$

then

$$V_{1m}^{*\mu}(yx_{<k}) - V_{1m}^{p_m^{\mathbf{M}_W} \mu}(yx_{<k}) < \frac{\Delta(m)}{W(\mu)}$$

So the above theorem shows that if there exists a policy p_m, for each lifetime m, whose value function comes to within $\Delta(m)$ of the optimal policy, then there is a *universal* policy $p_m^{\mathbf{M}_W}$ that comes to within $\Delta(m)/W(m)$. Using the above result we can now derive:

Theorem 5.8 (Hutter)**.** *If $\forall \mu \in \mathcal{M}'$ there exists a sequence of policies p_m such that*

$$\frac{1}{m} V_{1m}^{p_m \mu}(yx_{<k}) \stackrel{m \to \infty}{\longrightarrow} \frac{1}{m} V_{1m}^{*\mu}(yx_{<k})$$

then,

$$\frac{1}{m} V_{1m}^{p_m^{\mathbf{M}_W} \mu}(yx_{<k}) \stackrel{m \to \infty}{\longrightarrow} \frac{1}{m} V_{1m}^{*\mu}(yx_{<k})$$

The above shows that if the average value of a sequence of policies, with the average taken over the lifetime m of the agent, converges to the optimal, then so does the value of the sequence of universal policies. The speed of convergence is at most $1/W(\mu)$ slower.

The transfer learning setup described in Chaps. 3 and 4 can easily be extended to cover the AI agent setting. This is done by defining our task space to be $\mathcal{M}$ instead of $\mathcal{V}$, and then we obtain the same optimality results for both non-transfer and transfer learning case for learning a target chronological semimeasure (Theorem 5.6). Unfortunately, as we have seen above, these convergence rates *do not* translate to convergence rate of the value function which is what we are interested

in. The convergence results are in fact much weaker. So for instance, in the sequential transfer learning setting, we assume we have seen a sequence of tasks $\varphi := \varphi_{1,m-1} \in \mathcal{M}$. When learning the m^{th} task, we use the conditional prior $W(\mu|\varphi)$ and hence from Theorem 5.7 we get the 'convergence rate' for $\mathbf{M}_W$ of $\Delta(m)/W(\mu|\varphi)$. If we use the transfer prior $2^{-K(\mu|\varphi)}$, the convergence rate is $\Delta(m)2^{K(\mu|\varphi)}$. From Lemma 3.2, we have

$$2^{K(\mu|\varphi)} \stackrel{\times}{\leq} W(\mu|\varphi)$$

Hence for the artificial intelligent agent case, our transfer learning priors are better only upto a multiplicative constant. So optimality conditions are not as interesting. This requires further study and will be done so in future work.

5.4 Kolmogorov Complexity of Functions

One natural definition of Kolmogorov complexity of a function f given string q is $K'(f|q)$, the length of the shortest program that computes f given q as extra information (Hutter, 2002, Sect. 7), (Grunwald & Vitanyi, 2004, Sect 2.2.3) . So one objection to the definitions in chapter 3 may be that, since we are interested in $\mu \in \mathcal{V}$ as semimeasures (i.e. functions), perhaps we should define the complexity of $\mu \in \mathcal{V}$ as $K'(\mu|q)$. However K' is not computable in the limit, so to address this concern, we establish another reasonable definition of complexity of elements of $\mathcal{V}$, $K_{\mathcal{P}}$. We then deflate this possible objection by showing that $K_{\mathcal{P}}$ is in fact, upto a constant, equal to K. To motivate the definition of $K_{\mathcal{P}}$, we will begin by looking at a slight adaptation of a definition of Kolmogorov complexity of functions, K''. This was introduced and used in Hutter, 2002, and was defined primarily to counter the noncomputability in the limit of K'.

To define K'', we need a formal system $\mathcal{F}$ (Shoenfield, 1967) with axioms and inference rules in which we can formalize the notions of provability and Turing machines. We enclose formulas in $\mathcal{F}$ in $\S\ \S$ and the proof of a formula s is a sequence of formulas such that, each formula in the sequence is either an axiom or derived from a previous formula in the sequence via the inference rules, and the last formula in sequence is $\S\, s\, \S$. The properties of $\mathcal{F}$ we use are:

- The set of correct proofs in $\mathcal{F}$ is enumerable.
- We can formulate the formula $\S\, \forall x : \mu(x) \Uparrow \alpha(x)\, \S$ which is true if and only if $\forall x, U(\mu, x) \Uparrow U(\alpha, x)$.

Now we define $K''(\mu|q)$

Definition 5.7. *For $\mu \in \mathcal{V}, q \in \mathcal{A}^*$,*

$$K''(\mu|q) := \min_r \{ l(r) : r(q) = \alpha \textit{ and } \exists \textit{ proof } \S\, \forall x : \mu(x) \Uparrow \alpha(x) \,\S \} \ .$$

■

The above definition means $K''(\mu|q)$ is the length of the shortest program that given q as input, outputs a program α that *provably* lower semicomputes (denoted by $\Uparrow$) the same semimeasure as μ. This definition is slightly different from the one used in Hutter, 2002, which is:

$$K''_H(\mu) := \min_r \{ l(r) : \exists \text{ proof } \S\, \forall x : \mu(x) \Uparrow r(x) \,\S \} \ .$$

This is the length of the shortest program that provably lower semicomputes μ. However, now it is not entirely clear how the conditional should be defined. Intuition suggests we define it as

$$K''_H(\mu|q) := \min_r \{ l(r) : \exists \text{ proof } \S\, \forall x : \mu(x) \Uparrow r(\langle x, q \rangle) \,\S \} \ .$$

which is a little awkward. Hence, we adapt Hutter's definition to our K'' given above. It is easy to show, using standard methods, that $K''_H \stackrel{+}{=} K''$ for a small constant of equality. That is: Given r that is a witness in Def. 5.7 we can construct $r'(x) := (r(q))(x)$ to get $K''_H(\mu|q) \stackrel{+}{\leq} K''(\mu|q)$. Similarly, given r that is a witness in the definition of K''_H, we can define $r'(q) := r(\langle q, \cdot \rangle)$ to show $K''(\mu|q) \stackrel{+}{\leq} K''_H(\mu|q)$, which proves the equality. The constant of equality is small because the definition of r' in each case requires just a little bit of extra code.

Note that both K'' and K''_H are upper semicomputable because K is upper semicomputable and the set of correct proofs is enumerable. Now we have:

Lemma 5.1. *Let $\arg K''(\mu|q)$ denote the α that is the witness in Definition 5.7. Then,*

1) $\forall \mu \in \mathcal{V}, q \in \mathcal{A}^*, K''(\mu|q) \leq K(\mu|q)$

2) $\forall n \in \mathbb{N}, q \in \mathcal{A}^* \exists \mu \in \mathcal{V}$ *such that* $K(\mu|q) - K''(\mu|q) \stackrel{+}{\geq} n$

3) $\forall \mu \in \mathcal{V}, q \in \mathcal{A}^*, K(\arg K''(\mu|q)) \stackrel{+}{=} K''(\arg K''(\mu|q))$

4) $\forall q \in \mathcal{A}^*, \sum_{\mu \in \mathcal{V}} 2^{-K''(\mu|q)} = \infty \ .$

Proof. The results are mostly self-evident.

Part 1**.** This follows from definition since each $\mu \in \mathcal{V}$ provably computes the same

function as itself.

Part 2**.** Fix $q \in \mathcal{A}^*$, $\varphi \in \mathcal{V}$, and $n \in \mathbb{N}$. Now by the theory of random strings (see Li & Vitanyi, 1997), there exists infinitely many incompressible strings - i.e. strings s such that $K(s|\varphi, K(\varphi|q), q) \geq l(s)$. Let $l(s) = n$, and construct a $\mu \in \mathcal{V}$ which is just φ with s encoded in it at a fixed position t. Now $K(\mu|q) \stackrel{+}{=} K(s, \varphi|q)$, since, using t, given a program to generate μ given q, we can recover φ and s from it, and given a program to generate $\langle s, \varphi \rangle$ given q, we can construct μ. By Lemma 3.1 part 5, we get

$$K(s, \varphi|q) \stackrel{+}{=} K(s|\varphi, K(\varphi|q), q) + K(\varphi|q) \ .$$

By definition $K''(\mu|q) \leq K(\varphi|q)$, so we get:

$$K(\mu|q) - K''(\mu|q) \stackrel{+}{=} K(\varphi, s|q) - K''(\mu|q) \stackrel{+}{=} K(s|\varphi, K(\varphi|q), q) + K(\varphi|q) -$$
$$K''(\mu|q) \geq n + K(\varphi|q) - K''(\mu|q) \geq n + K(\varphi|q) - K(\varphi|q) = n \ .$$

Part 3**.** Follows from definition.

Part 4**.** for each $\varphi \in \mathcal{V}$, by the method in part 2, there are infinitely many $\mu \in \mathcal{V}$ such that $\forall x, \varphi(x) \Uparrow \mu(x)$ provably . So $\sum_{\mu_i \in \mathcal{V}} 2^{-K''(\mu_i|q)} = \infty$, as infinitely many μ_is have the same $K''(\mu_i|q)$ value.

□

Parts 1 and 2 in the lemma show that the K''s can uncover much more similarity between tasks than K. However, there is no advantage to using K'' for Bayesian transfer learning, as for any enumerable set $\mathcal{V}'$, the set of programs $\mathcal{V}'_{proof}$ that are provably equal to the elements of $\mathcal{V}'$ is also enumerable (because the set of correct proofs in $\mathcal{F}$ are enumerable). Therefore we get that for any $\mu \in \mathcal{V}'$, $\arg K''(\mu|q)$ is in $\mathcal{V}'_{proof}$. Since the error bound in Bayes mixtures depends only on the weight assigned to the generating semimeasure , from part 3 of the above lemma, substituting $\mathcal{V}'$ with $\mathcal{V}'_{proof}$ counteracts the benefit of using K''. Part 2 in the lemma seems to suggest that K'' deserves further study and this will be done in future.

However for now, we note that in the definition of K'' we require the witness to output a program that provably lower semicomputes the target function, but we do not require it to actually output the proof. To keep the witness honest, we will now look at a slightly altered version of K'' where the witness is also required to output this proof. It will then turn out that this new function and K are in fact equal upto a constant. We first define this altered version of K'':

Definition 5.8. *The* **provable Kolmogorov complexity** $K_{\mathcal{P}}$ *of* $\mu \in \mathcal{V}$ *is defined as*

follows:

$$K_{\mathcal{P}}(\mu) := \min_{p}\{l(p) : p(\varepsilon) = \langle \gamma, \pi \rangle \textit{ where } \pi \textit{ is a proof for } \S\, \forall x : \gamma(x) \Uparrow \mu(x)\, \S\} \ .$$

■

So now, in addition to γ that provably computes μ, we also require that the program output the corresponding proof. We can now define the conditional $K_{\mathcal{P}}$ and the information distances. But first we need:

Definition 5.9. *Let $\mathcal{J}_\mu$ be the enumerable set of all $\langle \gamma, \pi \rangle$ such that π is a proof of $\S\, \forall x : \gamma(x) \Uparrow \mu(x)\, \S$ ($\mathcal{J}_\mu$ is enumerable because the set of all correct proofs in $\mathcal{F}$ is enumerable).* ■

Now define:

Definition 5.10. *The* **conditional** $K_{\mathcal{P}}$ *is defined as:*

$$K_{\mathcal{P}}(\mu|\mu') := \min_{p}\{l(p) : \forall \tau' \in \mathcal{J}_{\mu'}, p(\tau') \in \mathcal{J}_\mu\} \ .$$

the **Information Distance** $E_{\mathcal{P}0}$ *is defined as:*

$$E_{\mathcal{P}0}(\mu, \mu') := \min_{p}\{l(p) : \forall \tau \in \mathcal{J}_\mu, \forall \tau' \in \mathcal{J}_{\mu'}\, , p(\tau) \in \mathcal{J}_{\mu'},\ p(\tau') \in \mathcal{J}_\mu\} \ .$$

and **Cognitive Distance** $E_{\mathcal{P}1}$ *is now defined as:*

$$E_{\mathcal{P}1}(\mu, \mu') := \max\{K_{\mathcal{P}}(\mu|\mu'), K_{\mathcal{P}}(\mu'|\mu)\} \ .$$

■

$K_{\mathcal{P}}$ is upper semicomputable by the same reasoning K'' is. However, our definition of $K_{\mathcal{P}}(.|.)$ is a lot stronger than definition of $K''(.|.)$ as we require that the $\arg K_{\mathcal{P}}(\mu|\mu')$ output an element in $\mathcal{J}_\mu$ given *any* element of $\mathcal{J}_{\mu'}$. Because of this last condition $K_{\mathcal{P}}$ (and by similar reasoning $E_{\mathcal{P}1}$ and $E_{\mathcal{P}0}$) is not upper semicomputable. However, we can show the following:

Lemma 5.2. *The following equalities now hold:*

1. $K_{\mathcal{P}}(\mu) \stackrel{+}{=} K(\mu)$.
2. $K_{\mathcal{P}}(\mu_{1,n}) \stackrel{+}{=} K(\mu_{1,n})$.
3. $K_{\mathcal{P}}(\mu|\mu') \stackrel{+}{=} K(\mu|\mu')$.
4. $E_1(\mu, \mu') \stackrel{+}{=} E_{\mathcal{P}1}(\mu, \mu')$.

5. $E_0(\mu, \mu') \stackrel{+}{=} E_{\mathcal{P}0}(\mu, \mu')$.

6. $E_{\mathcal{P}0}(\mu, \mu') \stackrel{+}{=} E_{\mathcal{P}1}(\mu, \mu')$.

7. $K_{\mathcal{P}}(\mu | \arg K_{\mathcal{P}}(\rho)) \stackrel{+}{=} K(\mu | \arg K_{\mathcal{P}}(\rho))$.

8. $K_{\mathcal{P}}(\mu | \arg K_{\mathcal{P}}(\rho)) + K_{\mathcal{P}}(\rho) \stackrel{+}{=} K_{\mathcal{P}}(\mu, \rho) \stackrel{+}{=} K_{\mathcal{P}}(\rho | \arg K_{\mathcal{P}}(\mu)) + K_{\mathcal{P}}(\mu)$.

That is the $K_{\mathcal{P}}$ and K etc. are equal upto a constant, and hence $K_{\mathcal{P}}$ satisfies some of the most interesting inequalities in Kolmogorov complexity theory. Hence this shows that there exists a reasonable definition of Kolmogorov complexity of elements of $\mathcal{V}$ for which it is equivalent to the Kolmogorov complexity of strings. That is, the objection stated in the beginning of the section, that since we are interested in $\mu \in \mathcal{V}$ as semimeasures (i.e. functions), perhaps we should define the complexity of $\mu \in \mathcal{V}$ as $K'(\mu|q)$, is not justified.

Now we prove Lemma 5.2, and we will do so using the following technical lemma:

Lemma 5.3. *Let $\mu_i \in \mathcal{V}, i \in \mathbb{N}_N$ and $\rho_j \in \mathcal{V}, j \in \mathbb{N}_M$ for finite N and M. Then,*

1. *For all program p such that*

   $$\forall X, X := \langle \tau_{1,N} : \tau_i \in \mathcal{J}_{\mu_i} \} \rangle,\ p(X) = Y \text{ where } Y = \langle \kappa_{1,M} : \kappa_i \in \mathcal{J}_{\rho_i} \} \rangle \ .$$

   there exists a program q with $l(p) \stackrel{+}{=} l(q)$ such that

   $$q(\langle \mu_{1,M} \rangle) = \langle \rho_{1,N} \rangle \ .$$

2. *Similarly given any program q' such that,*

   $$q'(\langle \mu_{1,M} \rangle) = \langle \rho_{1,N} \rangle \ .$$

   there exists a program p' with $l(p') \stackrel{+}{=} l(q')$ such that

   $$\forall X, X := \langle \tau_{1,N} : \tau_i \in \mathcal{J}_{\mu_i} \} \rangle,\ p'(X) = Y \text{ where } Y = \langle \kappa_{1,M} : \kappa_i \in \mathcal{J}_{\rho_i} \} \rangle \ .$$

Proof. Let p be a program as above. Then we can construct the requisite program q as follows. Program q given any $\langle \mu_{1,N} \rangle$, constructs $X := \langle \tau_{1,N} : \tau_i \in \mathcal{J}_{\mu_i} \} \rangle$ by setting $\tau_i := \langle \mu_i, \pi_i \rangle$ where π_i is simply the statement $\S\, \forall x : \mu_i(x) \Uparrow \mu_i(x)\, \S$. q then runs $p(X)$ to get $Y := \langle \kappa_{1,M} : \kappa_i \in \mathcal{J}_{\rho_i} \rangle$. We extract ρ_i from $\kappa_i := \langle \gamma, \pi \rangle$ decoding ρ_i in π as the last statement is π is of the form $\S\, \forall x : \gamma(x) \Uparrow \rho_i(x)\, \S$. Then program q outputs $\langle \rho_{1,M} \rangle$. As q has only constant amount of additional code, $l(q) \stackrel{+}{=} l(p)$.

Let q' be a program as above. Then we can construct the requisite program p' as follows. Program p' given any $X := \langle \tau_{1,N} : \tau_i \in \mathcal{J}_{\mu_i} \} \rangle$ extracts μ_i from $\tau_i = \langle \gamma, \pi \rangle$ using π and γ. It then runs $q(\langle \mu_{1,N} \rangle)$ to get $\langle \rho_{1,M} \rangle$. p' then outputs Y where $Y := \langle \kappa_{1,M} \rangle$ where $\kappa_i := \langle \rho_i, \pi \rangle$ and π is simply the statement $\S \forall x : \rho_i(x) \Uparrow \rho_i(x) \S$.

□

Proof of Lemma 5.2. We prove each part in turn.

Part 1. With μ_i set to ε in Lemma 5.3 part 1 we get $K(\mu) \stackrel{+}{\leq} K_{\mathcal{P}}(\mu)$; by part 2 of Lemma 5.3 we get $K_{\mathcal{P}}(\mu) \stackrel{+}{\leq} K(\mu)$. Hence this proves part 1 of this lemma.

Part 2. This follows from Lemma 5.3 with $\mu_i := \epsilon$ via similar reasoning as part 1.

Part 3. This follows from Lemma 5.3 with $\mu_i := \mu'$ via similar reasoning as part 1.

Part 4. This follows from part 3 and definitions of E_1 and $E_{\mathcal{P}1}$.

Part 5. This can be easily proved using the method in Lemma 5.3.

Part 6. This follows from parts 4, 5 and theorem 3.3.

Part 7. This follows from Lemma 5.3 with $\mu_i := \arg K_{\mathcal{P}}(\rho)$ via similar reasoning to parts 1 and 3.

Part 8. This now follows by Lemma 3.1 part 4, and parts 1, 2 and 3. □

So, in our definition of K'', if we include the additional information required in the form of the proof, we immediately get the equivalence between Kolmogorov complexity of functions and bit strings. We should also note that the above applies for $\mu \notin \mathcal{V}$ and notions of computability other than lower semicomputability.

5.5 Discussion

In this chapter we introduced some miscellaneous extensions to the Kolmogorov complexity based transfer learning paradigm. First we rederived our results for arbitrary bounded loss functions using Hutter's result. Then we extended our system to the Prediction with Expert Advice algorithms. Then we extended our result to the reinforcement learning setting. Finally we allayed some possible concerns about our definition of Kolmogorov complexity of functions by showing that other related and reasonable measures of function complexity also reduce to the regular Kolmogorov complexity case.

Chapter 6

Practical Approximations

In this chapter we develop practical approximations to the universally optimal sequential transfer prior ξ_{TL} developed in chapter 4. We consider Bayesian binary decision trees (Breiman et al., 1993) and in this setting develop approximations to our distance measures and universal transfer learning priors. We then apply our approximations in 144 individual transfer experiments using 7 arbitrarily chosen data-sets from the UCI machine learning repository (Newman et al., 1998). The arbitrary choice of data-sets make our experiments are the most general transfer experiments to date.

6.1 Bayesian Setting for Finite Spaces

We consider Bayesian transfer learning for input spaces $\mathcal{I}_i$ (possibly infinite) and finite output spaces $\mathcal{O}_i$. We assume finite hypothesis spaces $\mathcal{H}_i$, where each $\mathbf{h} \in \mathcal{H}_i$ is a computable conditional probability measure on $\mathcal{O}_i$, conditioned on elements of $\mathcal{I}_i$. So for $y \in \mathcal{O}_i$ and $x \in \mathcal{I}_i$, $\mathbf{h}(y|x)$ gives the probability of output being y given input x. Given $D_n = ((x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n))$ from $\mathcal{I}_i \times \mathcal{O}_i$, the probability of D_n according to $\mathbf{h} \in \mathcal{H}_i$ is given by:

$$h(D_n) := \prod_{k=1}^{n} h(y_k|x_k) \ .$$

The conditional probability of a new sample $(x_{new}, y_{new}) \in \mathcal{I}_i \times \mathcal{O}_i$ for any conditional probability measure μ is given, as usual, by:

$$\mu(y_{new}|x_{new}, D_n) := \frac{\mu((x_{new}, y_{new}), D_n)}{\mu(D_n)} \ . \tag{6.1}$$

So the learning problem is: given a training sample D_n, where for each $(x_k, y_k) \in D_n$ y_k is assumed to have been chosen according a $\mathbf{h} \in \mathcal{H}_i$, learn $\mathbf{h}$. The prediction problem is to predict the label of new sample x_{new} using (6.1). We ignore the probability of the inputs $P(x_i)$ because in the conditional, they cancel out. That is, assuming that the the x_is are generated according to some measure P, and denot-

ing by $\mu_P := \mu \cdot P$, we have the conditional probability of the new sample given sample (x_{new}, y_{new}) is given by definition to be:

$$\begin{aligned}
\rightarrow \quad & \mu_P((y_{new}, x_{new})|D_n) \\
= \quad & \frac{\mu_P(D_n, (x_{new}, y_{new}))}{\mu_P(D_n)} \\
= \quad & \frac{\mu(y_1, \cdots, y_n, y_{new}|x_1, \cdots, x_n, x_{new})P(x_1, x_2, \cdots, x_n, x_{new})}{\mu(y_1, \cdots, y_n|x_1, \cdots, x_n)P(x_1, x_2, \cdots, x_n)} \\
= \quad & \frac{\mu(y_1, \cdots, y_n, y_{new}|x_1, \cdots, x_n, x_{new})P(x_{new}|x_1, x_2, \cdots, x_n)}{\mu(y_1, \cdots, y_n|x_1, \cdots, x_n)}
\end{aligned}$$

Finally we have, again, by definition :

$$\begin{aligned}
& \mu(y_{new}|x_{new}, D_n)P(x_{new}|x_1, x_2, \cdots, x_n) = \mu_P((y_{new}, x_{new})|D_n) \\
& \Rightarrow \mu(y_{new}|x_{new}, D_n) = \frac{\mu_P((y_{new}, x_{new})|D_n)}{P(x_{new}|x_1, x_2, \cdots, x_n)} \\
& = \frac{\mu(y_{new}, y_1, \cdots, y_n|x_{new}, x_1, \cdots, x_n)P(x_{new}|x_1, x_2, \cdots, x_n)}{\mu(y_1, \cdots, y_n|x_1, \cdots, x_n)P(x_{new}|x_1, x_2, \cdots, x_n)} \\
& = \frac{\mu(y_{new}, y_1, \cdots, y_n|x_{new}, x_1, \cdots, x_n)}{\mu(y_1, \cdots, y_n|x_1, \cdots, x_n)} \\
& = \frac{\mu((x_{new}, y_{new}), D_n)}{\mu(D_n)}
\end{aligned}$$

Turning our attention back to the prediction problem, in this setting the sequence of predictions made over the sequence of outputs to be observed. As before, this is done using a Bayes mixture $\mathbf{M}_W$ over $\mathcal{H}_i$:

$$\mathbf{M}_W(D_n) := \sum_{\mathbf{h} \in \mathcal{H}_i} \mathbf{h}(D_n) W(\mathbf{h}) \text{ with } \sum_{\mathbf{h} \in \mathcal{H}_i} W(\mathbf{h}) \leq 1 \tag{6.2}$$

and hence the convergence bound Theorem 4.1 is now expressed as, $\forall x \in \mathcal{I}_i$:

$$\sum_{n=1}^{\infty} \sum_{D_n} \mathbf{h}_j(D_n) \sum_{y \in \mathcal{O}_i} \left([\mathbf{M}_W(y|x, D_n) - \mathbf{h}_j(y|x, D_n)]^2 \right) \leq -\ln W(\mathbf{h}_j). \tag{6.3}$$

We seek to perform sequential transfer using decision trees. Since the Kolmogorov complexity K is computable only in the limit, to apply the methods and results in Chap. 3 in transferring using Bayesian decision trees, we need to approximate K and hence ξ_{TL}. Furthermore we also need to specify the spaces $\mathcal{H}_i, \mathcal{O}_i, \mathcal{I}_i$ and how to sample from the approximation of ξ_{TL}. We address each issue in turn.

6.2 Brief Primer on Practical Bayesian Learning

In chapter 4 and even in the preceding section we took a somewhat formal view of Bayesian learning where we did not consider whether it is computationally feasible to predict using the Bayes mixture in practice. In this section we address this issue. In majority of cases of interest in practice, it is intractable to compute the mixture $\mathbf{M}_W$ because of difficult summations or integrals involved, and so it is impractical to predict the label of a new sample using $\mathbf{M}_W(y_{new}|x_{new}, D_n)$ directly. So the *posterior* is used to approximate the mixture as described below. Before proceeding, we note that to avoid confusion, we use P and W to denote measures that are given by our assumed model, and Pr to denote densities that arise from the Ps via probability theory.

The posterior of $\mu \in \mathcal{H}_i$ given D_n is given according to Bayes' rule by:

$$Pr(\mu|D_n) := \frac{\mu(D_n)W(\mu)}{Pr(D_n)} = \frac{\mu(D_n)W(\mu)}{\sum_{\mu\in\mathcal{H}_i}\mu(D_n)W(\mu)} \ .$$

We can now rewrite the mixture $\mathbf{M}_W$ by:

$$\begin{aligned}
\mathbf{M}_W(y_{new}|x_{new}, D_n) &:= \frac{\mathbf{M}_W[(x_{new}, y_{new}), D_n]}{\mathbf{M}_W(D_n)} \\
&= \frac{\sum_{\mu\in\mathcal{H}_i}\mu[(x_{new}, y_{new}), D_n]W(\mu)}{\sum_{\mu\in\mathcal{H}_i}\mu(D_n)W(\mu)} \\
&= \frac{\sum_{\mu\in\mathcal{H}_i}\mu(y_{new}|x_{new}, D_n)\mu(D_n)W(\mu)}{\sum_{\mu\in\mathcal{H}_i}\mu(D_n)W(\mu)} \\
&= \sum_{\mu\in\mathcal{H}_i}\mu(y_{new}|x_{new}, D_n)Pr(\mu|D_n) \ .
\end{aligned}$$

Therefore, the posterior is the weight that is assigned to each $\mu \in \mathcal{H}_i$ because of the evidence D_n. The mixture $\mathbf{M}_W$ is now approximated as an averaging of N measures ρ_i that were sampled from $\mathcal{H}_i$ according to the posterior $Pr(\mu|D_n)$:

$$\widehat{\mathbf{M}}_W(x_{new}|y_{new}, D_n) := \frac{1}{N}\sum_{i=1}^{N}\rho_i(y_{new}|x_{new}) \ .$$

Unfortunately, even the posterior is usually difficult to sample from directly. However, if we can compute $\mu(D_n)$ and $W(\mu)$ upto a normalization term, then we can use *Markov Chain Monte Carlo* methods to sample from it. The literature on Markov Chain Monte Carlo is vast, so here we will content ourself with giving the briefest of description. Essentially, the idea is to construct an *irreducible and aperiodic* Markov chain with $\mathcal{H}_i$ as its state space, and that has the posterior as its *stationary distribution*. There are standard ways of constructing such chains, and

one popular method, the Metropolis-Hastings chain that we use in our experiments, is described in Sect. 6.4 and algorithm 6.2. Given this, the chain is simulated using a random number generator, and eventually it is guaranteed to converge to the stationary distribution – i.e. after a certain number of steps, as we simulate the chain we start sampling from the posterior. The issue of determining when a chain converges is an highly active research area. There are methods known as exact-sampling methods, that guarantee that if the algorithm halts, then the samples are generated from the stationary distribution. However, these require that the Markov chain state space satisfy certain special criteria and so we do not use them in our experiments.

For an introduction to the Bayesian approach to machine learning, please see Andrieu et al., 2003; Neal, 2004; Mackay, 2003. For full details on the Bayesian approach to statistics see Bernardo & Smith, 1994, and for more details on Markov chain Monte Carlo please see Gilks et al., 1996; Robert & Casella, 2005. For an introduction Markov chains and fast mixing/convergence to stationary distribution (and additional references), please see Behrends, 2000; Häggström, 2002. For more on exact sampling, please see Propp & Wilson, 1996; Fill, 1998; Häggström, 2002.

6.3 The Bayesian Decision Tree Model

We will consider transfer learning with Bayesian binary decision trees as our hypothesis space $\mathcal{H}_i$s. Of course, decision trees (Breiman et al., 1993) are well known models for classification and regression, but when used in a Bayesian setting, they need some additional explanation. In particular, Bayesian decision trees, whether used for classification or regression, are parameterized by continuous parameters and hence we need to describe how they fit into the finite hypothesis-space case we described above.

We use the standard Bayesian decision tree setting as described in Denison et al., 2005. We assume that $\mathcal{I}_i := [0,1]^{|\mathbf{f}_i|}$, where $\mathbf{f}_i$ is a finite set of features, and finite $\mathcal{O}_i := \mathbb{N}_o$ for some $o \in \mathbb{N}$. As is well known, decision trees partition the input space $\mathcal{I}_i$ into a finite set of hypercubes, defined by axis parallel hyperplanes. In the Bayesian setting, we assume that within each such hypercube h_k, the distribution over o classes is given by a multinomial distribution with parameter $\vec{\theta}_k$, a vector of o elements such that $\sum_{j=1}^{o} \vec{\theta}_k(j) = 1$. So for any sample D_n, the likelihood of h_k is given by:

$$P(D_n|\vec{\theta}_k, h_k) := \prod_{j=1}^{o} \vec{\theta}_k(j)^{m_{kj}}$$

where m_{kj} is the number of pairs in $(x, y) \in D_n$ with $x \in h_k$ and $y = j$. We do not include the $\frac{n!}{\prod_j m_{kj}!}$ term above because we consider D_n to be a sequence of pairs, rather than a representative of any sample with m_{kj} elements of class j. We assume a Dirichlet prior over the parameters $\vec{\theta}_k(j)$ (for details see for instance, Friedman & Singer, 1998), for which the density function is given using hyperparameters α_{kj} as follows:

$$P(\vec{\theta}_k|h_k) := \frac{\Gamma(\sum_j \alpha_{kj})}{\prod_j \Gamma(\alpha_{kj})} \prod_j \vec{\theta}_k(j)^{\alpha_{kj}-1}$$

where the normalization term consists of the well known gamma function, $\Gamma(x) := \int_0^\infty t^{x-1}e^{-t}dt$, with the property that for $\Gamma(x) = (x-1)\Gamma(x-1)$. We set each $\alpha_{kj} = 1$, which corresponds to the uniform prior over the value of $\vec{\theta}_k(j)$. Now, we have

$$\begin{aligned}
Pr(D_n|h_k) &:= \int \prod_{j=1}^{o} \vec{\theta}_k(j)^{m_{kj}} P(\vec{\theta}_k) d\vec{\theta}_k \\
&= \int \prod_{j=1}^{o} \vec{\theta}_k(j)^{m_{kj}} \frac{\Gamma(\sum_j \alpha_{kj})}{\prod_j \Gamma(\alpha_{kj})} \prod_j \vec{\theta}_k(j)^{\alpha_{kj}-1} d\vec{\theta}_k \\
&= \frac{\Gamma(\sum_j \alpha_{kj})}{\prod_j \Gamma(\alpha_{kj})} \int \prod_{j=1}^{o} \vec{\theta}_k(j)^{m_{kj}+\alpha_{kj}-1} d\vec{\theta}_k \\
&= \frac{\Gamma(\sum_j \alpha_{kj})}{\prod_j \Gamma(\alpha_{kj})} \frac{\prod_j \Gamma(m_{kj}+\alpha_{kj})}{\Gamma(\sum_j m_{kj}+\alpha_{kj})} \\
&= \Gamma(o) \frac{\prod_j \Gamma(m_{kj}+1)}{\Gamma(\sum_j m_{kj}+1)} \quad . \qquad (6.4)
\end{aligned}$$

Therefore, the probability of the $y_{new} = i$ for input $x_{new} \in h_k$ is now simply given by,

$$\begin{aligned}
Pr(y_{new} = a|x_{new} \in h_k, D_n) &:= \frac{P((y_{new} = a, x_{new}), D_n)}{P(D_n)} \\
&= \frac{m_{ka} + \alpha_{ka}}{\sum_j m_{kj} + \alpha_{kj}} \\
&= \frac{m_{ka} + 1}{\sum_j m_{kj} + 1} \quad . \qquad (6.5)
\end{aligned}$$

Hence, given h_k, the predictive distribution is determined solely by the sample D_n. And so we are left with the task of choosing the partitions h_k – i.e. the *structure* of the tree. And in Bayesian decision tree learning for classification, this is what we learn and restrict our attention to. So from now on, a $\mathbf{h}$ will refer to the structure of the tree that consists of $M^{\mathbf{h}}$ partitions h_k.

The likelihood of a tree $\mathbf{h}$ is now given by

$$\mathbf{h}(D_n) := P(D_n|\mathbf{h}) := \prod_{k=1}^{M^{\mathbf{h}}} P(D_n|h_k)$$

The posterior for the tree is now given by:

$$\begin{aligned} Pr(\mathbf{h}|D_n) &:= Pr(h_{1,M^{\mathbf{h}}}|D_n) \\ &= \frac{Pr(D_n|h_{1,M^{\mathbf{h}}})P(h_{1,M^{\mathbf{h}}})}{Pr(D_n)} \\ &= \frac{P(h_{1,M^{\mathbf{h}}})\prod_{k=1}^{M^{\mathbf{h}}} Pr(D_n|h_k)}{Pr(D_n)} \end{aligned} \tag{6.6}$$

where $P(h_{1,M^{\mathbf{h}}})$ is the prior over the structure. To complete the definition of any Bayesian decision tree learning algorithm, transfer or otherwise, all we need to do is specify the prior over the structures. In the following, we define the structure more formally and set the ground for describing our approximation to the theoretically optimal Bayesian transfer learning algorithm discussed so far.

A tree $\mathbf{h} \in \mathcal{H}_i$ is defined recursively (see Figure 6.1):

$$\mathbf{h} := \mathbf{n}_{root}$$
$$\mathbf{n}_j := r_j\ \mathbf{C}_j\ \emptyset\ \emptyset \mid r_j\ \mathbf{C}_j\ \mathbf{n}_L^j\ \emptyset \mid r_j\ \mathbf{C}_j\ \emptyset\ \mathbf{n}_R^j \mid r_j\ \mathbf{C}_j\ \mathbf{n}_L^j\ \mathbf{n}_R^j$$

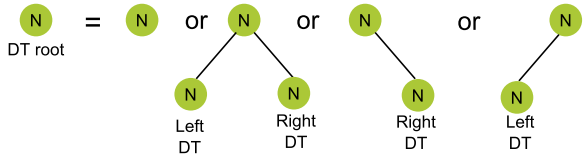

Figure 6.1: Schematic illustration of recursive tree definition.

So each decision tree is defined by its root node $\mathbf{n}_{root}$. Each node is either $\emptyset$, or consists of a rule a r and a vector $\mathbf{C}$ and two sub-decision trees (each of which are defined the same way). Each rule r is of the form $f < v$, where $f \in \mathbf{f}_i$ and v is a value for f. Categorical features are converted to integer valued features for this purpose. $\mathbf{C}$ is a vector of size o, with component i corresponding to the i^{th} class. The vector $\mathbf{C}$ is used during classification only when the corresponding node has one or more $\emptyset$ children, and $\mathbf{n}_i.\mathbf{C}(j)$ contains the value of m_{kj} for all the inputs in D_n that belong to the partition defined by the node $\mathbf{n}_i$ and its parents. We restrict the possible values of v for each feature to the the values observed in the sample D_n, and so this makes the space of possible trees finite and *brings the Bayesian decision tree framework discussed so far in the framework of finite space discussed*

in Sect. 6.1. Classification is performed using algorithm 6.1. So all we need to do now is define the priors.

Algorithm 6.1 Method for classifying input x using decision tree $\mathbf{h}$

Let $\mathbf{n}_{cur} \leftarrow \mathbf{h}.node_{root}$
while Probability is not Output **do**
  if x satisfies $\mathbf{n}_{cur}.r$ **then**
    $\mathbf{n}_{next} \leftarrow \mathbf{n}_{cur}.n_L$
  else
    $\mathbf{n}_{next} \leftarrow \mathbf{n}_{cur}.n_R$
  end if
  if then$\mathbf{n}_{next} \neq \emptyset$
    $\mathbf{n}_{cur} \leftarrow \mathbf{n}_{next}$
  else
    Output probability of class via (6.5) using values m_{kj} stored in $\mathbf{n}_{cur}.\mathbf{C}$.
  end if
end while

6.4 Transfer Learning in Bayesian Decision Trees

Before beginning this section we draw attention to the fact that $\mathbf{h}$ now refers only to the structure. To be able to use approximation of our transfer method in this case, we need to define the approximation to Kolmogorov complexity of each tree. Now, the size of each tree is Sc_0 where S is the number of nodes, and c_0 is a constant, denoting the size of each rule entry, the outgoing pointers, and $\mathbf{C}$. Since c_0 and the length of the program code p_0 for computing the tree output are constants independent of the tree, we define the length/complexity of a tree as $\mathit{Kxt}(\mathbf{h}) := S$. So the approximation of Kolmogorov complexity $K(\mathbf{h})$ of tree $\mathbf{h}$ is given by $\mathit{Kxt}(\mathbf{h})$. Hence, in the single task case, the prior we use is the approximation to the Solomonoff-Levin prior $2^{-K(\mathbf{h})}$ and is given by:

$$P(\mathbf{h}) := \frac{2^{-\mathit{Kxt}(\mathbf{h})}}{Z_{\mathit{Kxt}}} \ .$$

where the Z is a normalization term. The Z exists, here because $\mathcal{H}$s are finite, and in general because $k_i = Sc_0 + l(p_0)$ gives lengths of programs, which are known to satisfy the Kraft inequality $\sum_i 2^{-k_i} \leq 1$.

For the transfer learning case, we need to approximate $K(.|.)$. We are going to consider transferring from $m-1$ previously learned trees, and so without loss of generality, assume that $\mathbf{h} \in \mathcal{H}_{i_m}$ and $\mathbf{h}' \in \mathcal{H}_{i_j}$, $j < m$. We now approximate $K(.|.)$ using a function that is defined for a single previously learned tree as

follows:

$$Kxt_2(\mathbf{h}|\mathbf{h}') := Kxt(\mathbf{h}) - d(\mathbf{h}, \mathbf{h}')$$

where $d(\mathbf{h}, \mathbf{h}')$ is maximum number of overlapping nodes starting from the root nodes (see Figure 6.2):

$$\begin{aligned} d(\mathbf{h}, \mathbf{h}') &:= d(\mathbf{n}_{root}, \mathbf{n}'_{root}) & d(\mathbf{n}, \emptyset) &:= 0 \\ d(\mathbf{n}, \mathbf{n}') &:= 1 + d(\mathbf{n}_L, \mathbf{n}'_L) + d(\mathbf{n}_R, \mathbf{n}'_R) & d(\emptyset, \mathbf{n}') &:= 0 \end{aligned}$$

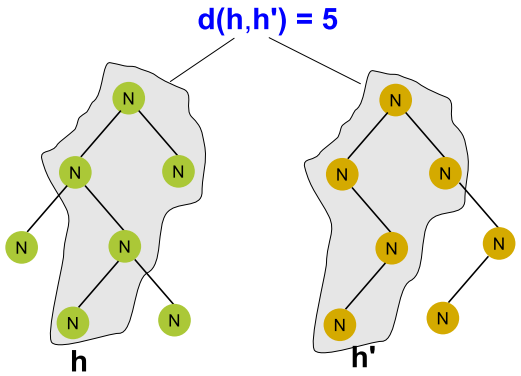

Figure 6.2: Example illustrating d between two decision trees.

and so in the transfer learning case, the prior when there is only one tree learned previously,

$$P(\mathbf{h}|\mathbf{h}') := \frac{2^{-Kxt_2(\mathbf{h}|\mathbf{h}')}}{Z_{Kxt_2}} \quad .$$

In both cases, we can sample from the prior directly by growing the decision tree dynamically. This fact will become useful below when we sample from the posterior using a MCMC algorithm. Call a $\emptyset$ in $\mathbf{h}$ a hole. Then for $P(\mathbf{h})$, during the generation process, we first generate an integer k according to 2^{-t} distribution (easy to do using a pseudo random number generator). Then at each step we select a hole uniformly at random and then create a node there with two more holes and the rule generated randomly. The prior $2^{-Kxt(\mathbf{h})}/Z_{Kxt}$ gives equal probability to all trees of the same complexity k, while giving trees of complexity k half as probability as the trees of complexity $k-1$. So the above procedure samples from the prior as it samples k according to 2^{-t} and gives equal probability to every tree of size $Kxt(\mathbf{h}) = k$ by growing the tree uniformly at random.

In the transfer learning case, for prior $P(\mathbf{h}|\mathbf{h}')$ we first generate an integer k according to 2^{-t} distribution. Then we generate a tree using the above procedure until we get a tree $\mathbf{h}$ with $Kxt_2(\mathbf{h}|\mathbf{h}') = k$. $P(\mathbf{h}|\mathbf{h}')$ gives equal probability to all trees of the same conditional complexity $Kxt_2(\mathbf{h}|\mathbf{h}')$ equal to k, while giving trees of complexity k half as probability as the trees of complexity $k-1$. So the

above procedure samples from the transfer prior as it samples k according to 2^{-t} and gives equal probability to every tree of size $Kxt_2(\mathbf{h}|\mathbf{h}') = k$ by growing the tree uniformly at random.

For $m-1$ previously learned trees $\mathbf{h}_{1,m-1}$, with $\mathbf{h}_j \in \mathcal{H}_{i_j}$, we define Kxt_m as an averaging of the contributions of each $m-1$ previously learned trees:

$$Kxt_m(\mathbf{h}_m|\mathbf{h}_{1,m-1}) = -\log\left(\frac{1}{m-1}\sum_{i=1}^{m-1} 2^{-Kxt_2(\mathbf{h}|\mathbf{h}_i)}\right)$$

In the transfer learning case, the prior, and hence our approximation to ξ_{TL} is

$$P_{TL}(\mathbf{h}|\mathbf{h}_{1,m-1}) := \frac{2^{-Kxt_m(\mathbf{h}|\mathbf{h}_{1,m-1})}}{Z_{Kxt_m}} \quad .$$

which reduces to:

$$\frac{1}{[(m-1)Z_{Kxt_m}]}\sum_{i=1}^{m-1} 2^{-Kxt_2(\mathbf{h}|\mathbf{h}_i)} \quad . \tag{6.7}$$

To sample from this, we can simply select one of the $m-1$ trees at random and then use the procedure for sampling from 2^{-Kxt_2} to get the new tree. So, finally, the approximation of the transfer learning mixture $\mathbf{M}_{\xi_{\mathrm{TL}}}$ is now:

$$\mathbf{M}_{P_{TL}}(D_n) = \sum_{\mathbf{h}\in\mathcal{H}_{i_m}} \frac{\mathbf{h}(D_n)2^{-Kxt_m(\mathbf{h}|\mathbf{h}_{1,m-1})}}{Z_{Kxt_m}}$$

where $\mathbf{h}(D_n)$ is $Pr(D_n|h_{1,M^{\mathbf{h}}})$ from (6.6). So by (6.3), the convergence rate for $\mathbf{M}_{P_{TL}}$ is given by $Kxt_m(\mathbf{h}|\mathbf{h}_{1,m-1})\ln\sqrt{2} + \log Z_{Kxt_m}$ (the $\log Z_{Kxt_m}$ is a constant that is same for all $\mathbf{h} \in \mathcal{H}_i$). In our experiments we actually used the exponent 1.005^{-Kxt_m} instead of 2^{-Kxt_m} above to speed up convergence of our MCMC method.

Algorithm 6.2 Metropolis-Hastings Algorithm

1: Let D_n be the training sample;
2: Select the current tree/state $\mathbf{h}_{cur}$ using the proposal distribution $q(\mathbf{h}_{cur})$.
3: **for** $i = 1$ to J **do**
4: Choose a candidate next state $\mathbf{h}_{prop}$ according to the $q(\mathbf{h}_{prop})$.
5: Draw u uniformly at random from $[0, 1]$
6: Set $\mathbf{h}_{cur} := \mathbf{h}_{prop}$ if $A(\mathbf{h}_{prop}, \mathbf{h}_{cur}) > u$, where A is defined by

$$A(\mathbf{h}, \mathbf{h}') := \min\left\{1, \frac{\mathbf{h}(D_n)2^{-Kxt_m(\mathbf{h}|\mathbf{h}_{1,m-1})}q(\mathbf{h}')}{\mathbf{h}'(D_n)2^{-Kxt_m(\mathbf{h}'|\mathbf{h}_{1,m-1})}q(\mathbf{h})}\right\}$$

7: **end for**

As in standard Bayesian MCMC methods, the idea will be to draw N samples $\mathbf{h}_{m_i}$ from the posterior, $Pr(\mathbf{h}|D_n, \mathbf{h}_{1,m-1})$ which is given by

$$Pr(\mathbf{h}|D_n, \mathbf{h}_{1,m-1}) := \frac{\mathbf{h}(D_n)2^{-Kxt_m(\mathbf{h}|\mathbf{h}_{1,m-1})}}{Z_{Kxt_m}P(D_n)}$$

Then we will approximate $\mathbf{M}_{P_{TL}}$ by

$$\widehat{\mathbf{M}}_{P_{TL}}(y|x) := \frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_{m_i}(y|x)$$

We will use the standard Metropolis-Hastings algorithm to sample from $\mathbf{M}_{P_{TL}}$ (see Sect. 6.2 for details). The algorithm is given in Table 6.2. The algorithm is first run for some $J = T$, to get the Markov chain $q \times A$ to converge, and then starting from the last $\mathbf{h}_{cur}$ in the run, the algorithm is run again for $J = N$ times to get N samples for $\widehat{\mathbf{M}}_{P_{TL}}$. In our experiments we set T to 1000 and $N = 50$. We set q to our prior $2^{-Kxt_m}/Z_{Kxt_m}$, and hence the acceptance probability A is reduced to $\min\{1, \mathbf{h}(D_n)/\mathbf{h}'(D_n)\}$. Note that every time after we generate a tree according to q, we set the $\mathbf{C}$ entries to m_{kj} values obtained from the training sample D_n

The main question that one will ask about the approximations presented in this section is just how good these approximations are. One very meaningful and appropriate way to answer this question is by looking at how well these methods perform in practice. This is done in the next section where we show that our approximations perform quite well and enable us to perform very general and successful transfer experiments.

6.5 Experiments

6.5.1 Setup of the Experiments

In our experiments we used 7 data-sets from the UCI machine learning repository Newman et al., 1998. The data-sets and their summary are given in table 6.1. To show transfer of information, we chose 3 data-sets to transfer to, and for each such data-set we chose 3 other data-sets at random to transfer from. So there are 9 pairs of data-sets we performed transfer experiments for. For each such pair, we divided up the transfer-to and transfer-from data-set into $x/(100-x)$ and $w/(100-w)$ portions respectively, where $x, w \in \{20, 40, 60, 80\}$. We used $x\%$ and $w\%$ of the samples as a training set and $100-x\%$ and $100-w\%$ of the samples as the test set for the corresponding data-sets. For each of the pairs, we first learned the transfer-from data set using the $w/(100-w)$ sample and then learned the transfer-to data-set using $x/(100-x)$ sample and the 50 trees sampled during learning

of the transfer-from data set (as described at the end of the preceeding section). So we performed $3 \times 4 \times 3 \times 4 = 144$ different transfer experiments in total. We present the results below for each transfer-to data-set, where all error rates reported were obtained by averaging over 10 different runs. Within each run we shuffled the samples of each data-set before splitting them up.

Table 6.1: Summary of the data-sets used in the transfer experiments.

Database	**# of Samples**	**# of Features**	**# of Classes**	**Ref. Name**
E-coli	336	7	8	ecoli
Yeast	1484	8	10	yeast
Australian Credit	690	14	2	aus
German Credit	1000	20	2	german
Hepatitis	155	19	2	hep
Breast Cancer,Wisc.	699	9	2	bc-wisc
Heart Disease, Cleve.	303	14	5	heart

6.5.2 Overview and Interpretation of Results

Before we dive into the details of the experiments, we will make some observations about our methods and the results presented below. The key result we observe is that in most cases we see improvement in performance (presented in terms of percentage improvement with respect to the non-transfer case), and in many cases the improvement is significant. Furthermore, when we do see reduction in performance, in most cases it is $< 2\%$ and never $> 10\%$. This seems to give evidence that the approximations of our transfer method in Chap. 4 that we have developed are effective, as they bear out the theoretical justification that transfer should never hurt too much. In addition, we can also give an intuitive explanation of the results in purely Bayesian machine learning terms without reference to our AIT based results.

MCMC methods are essentially stochastic exploration methods with nice convergence guarantees. When we perform transfer learning, we change the prior so that the MCMC algorithm explores certain areas of the hypothesis space with higher probability. Now the base (Non-Transfer) learner with prior $2^{-Kt(\mathbf{h})}$ is simply a Bayesian learner with a Occam prior, assigning higher probability to smaller trees. Use of the 2^{-Kt_m} priors during transfer forces the MCMC algorithm to focus its attention on trees with size possibly larger than recommended by the Occam prior. For this reason, transfer learning improves performance. The reason it does not degrade performance significantly is because MCMC methods, being

stochastic exploration methods, automatically reject larger trees that causes lower performance in the transfer learning case. The tradeoff in the transfer learning case is between possibly improved performance and possibly higher computational cost from testing larger trees. In case the reader is wondering, the computational cost is not twice as much because we assume that the we were going to learn the transfer-from task anyway.

Table 6.2: Non-Transfer error rates for the Data Sets

Data-set	*20/80*	*40/60*	*60/40*	*80/20*
ecoli	$19.5\%, 3.4$	$13.38\%, 4.45$	$9.9\%, 2.11$	$10.89\%, 5.8$
yeast	$15.3\%, 2.77$	$17.88\%, 3.96$	$17.0\%, 3.06$	$14.89\%, 2.73$
aus	$21.93\%, 4.03$	$18.55\%, 2.25$	$20.9\%, 3.08$	$18.9\%, 2.1$
german	$31.6\%, 1.38$	$31.6\%, 1.38$	$29.0\%, 1.27$	$31.1\%, 4.47$
hep	$23.3\%, 1.85$	$20.1\%, 3.7$	$20.8\%, 4.35$	$19.8\%, 1.38$
bc-wisc	$10.8\%, 3.1$	$8.92\%, 1.01$	$8.27\%, 1.93$	$8.99\%, 2.3$
heart	$26.6\%, 4.7$	$27.7\%, 3.9$	$26.6\%, 3.45$	$23.3\%, 1.8$

Finally, to ensure that the improvement in performance is due to transfer and not because our base learner was faulty, we compared the error rate of the base learner to results in previous work. From a survey of literature it seems the error rate for our classifier for the $80/20$ case is always at least a couple of percentage points better than C4.5. As an example, for *ecoli* our classifier outperforms Adaboost and Random Forests in Breiman, 2001, but is a bit worse than these for *German Credit*. These non-transfer results are summarized in table 6.2. Our results for each Transfer-To data-set appear starting on the next page.

6.5.3 Results for the *ecoli* Dataset

Tables 6.4 to 6.7 and Figs. 6.3 to 6.6 shows the results of transfer learning experiments when the transfer-to data-set was ecoli with types $20/80$, $40/60$, $60/40$ and $80/20$ respectively. The transfer from data-sets were yeast, german and bc-wisc. As can be seen by just skimming through the results, particularly the percentage improvement part of the results (the second part of each table and the figures), in almost all cases, transfer learning results in improved performance than the no-transfer case. The tables and figures appear starting in the next page

Table 6.3: **Table Key:** The *From Type* row gives the type of data-set information is being transferred from - *No-Trans* means no transfer is occurring, and $x/(100-x)$ mean the corresponding data-set type (see text for details). Each subsequent row gives the result when information is transferred from the corresponding data-set. The top half of the table gives the actual error rates and standard deviation, and the lower half gives the percentage improvement in each case.

TRANSFER TO $x/(1-x)$ TRANSFER-TO DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
data-set 1	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$
data-set 2	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$
data-set 3	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$
TRANSFER TO $x/(1-x)$ TRANSFER-TO DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/40*	*40/60*	*60/40*	*80/20*
data-set 1	–	9%	9%	9%	9%
data-set 2	–	9%	9%	9%	9%
data-set 3	–	9%	9%	9%	9%

Table 6.4: Results of 12 transfer experiments for the $20/80$ ecoli data set. See Table Key for meaning of the table entries.

TRANSFER TO 20/80 ECOLI DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
yeast	$19.5\%, 6.44$	$11.19\%, 4.27$	$13.54\%, 5.87$	$12.39\%, 4.94$	$13.88\%, 3.21$
german	$19.5\%, 6.44$	$14.7\%, 4.46$	$11.72\%, 4.66$	$14.25\%, 3.26$	$11.12\%, 2.93$
bc-wisc	$19.5\%, 6.44$	$14.63\%, 4.43$	$12.95\%, 4.52$	$12.54\%, 4.77$	$11.31\%, 4.26$
TRANSFER TO 20/80 ECOLI DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
yeast	–	42.62%	30.56%	36.46%	28.82%
german	–	24.62%	39.9%	26.92%	42.97%
bc-wisc	–	24.97%	33.59%	35.69%	42.0%

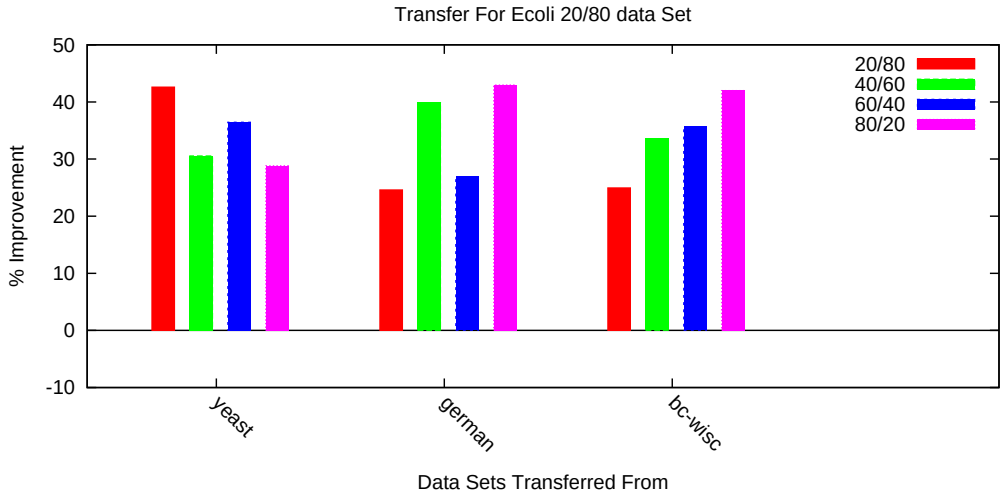

Figure 6.3: Percentage Improvement for the 20/80 ecoli data-set. Each color represents a particular type of the transfer-from data-set.

20/80 ecoli data-set: For the $20/80$ data-set type, in all cases we observe significant improvement in performance, which is intuitively satisfying because in this case the loss in performance due to reduced data will be most severe, and hence opportunities for transfer the most.

Table 6.5: Results of 12 transfer experiments for the $40/60$ ecoli data set. See Table Key for meaning of the table entries.

TRANSFER TO 40/60 ECOLI DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
yeast	$13.38\%, 4.5$	$11.44\%, 3.95$	$11.54\%, 3.86$	$10.05\%, 3.29$	$10.25\%, 2.45$
german	$13.38\%, 4.5$	$10.2\%, 2.73$	$11.64\%, 3.84$	$9.9\%, 4.47$	$10.6\%, 4.16$
bc-wisc	$13.38\%, 4.5$	$11.84\%, 3.26$	$9.45\%, 3.61$	$9.3\%, 2.63$	$9.3\%, 2.73$
TRANSFER TO 40/60 ECOLI DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
yeast	–	14.5%	13.75%	24.89%	23.39%
german	–	23.77%	13.0%	26.01%	20.78%
bc-wisc	–	11.51%	29.37%	30.49%	30.49%

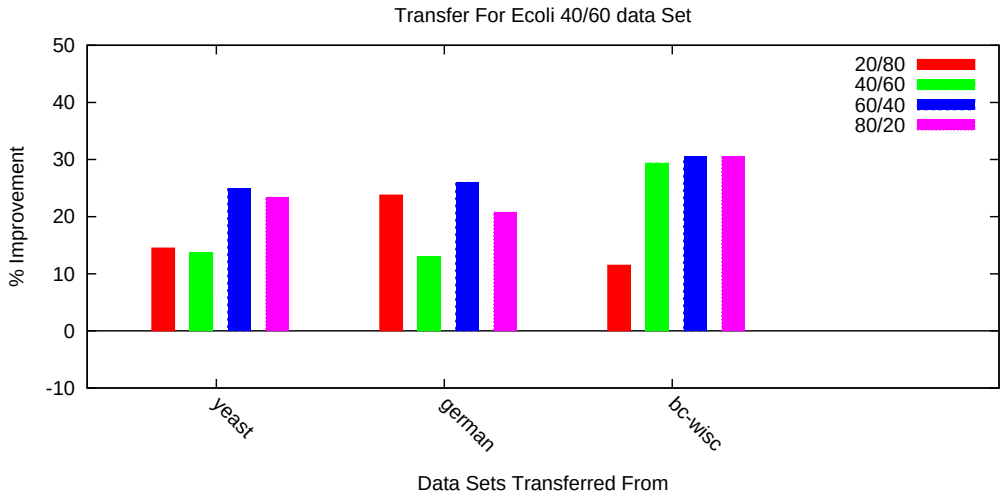

Figure 6.4: Percentage Improvement for the 40/60 ecoli data-set. Each color represents a particular type of the transfer-from data-set.

40/60 ecoli data-set: We see improvement in performance for the $40/60$ data-set type that is similar to those for the $20/80$ type, which is also intuitively satisfying for the same reason as above. In addition, we do not see any adverse effect due to transfer.

Table 6.6: Results of 12 transfer experiments for the $60/40$ ecoli data set. See Table Key for meaning of the table entries.

TRANSFER TO 60/40 ECOLI DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
yeast	$9.9\%, 2.11$	$10.07\%, 3.08$	$8.21\%, 2.97$	$8.66\%, 2.74$	$6.34\%, 2.25$
german	$9.9\%, 2.11$	$9.48\%, 3.61$	$8.06\%, 4.18$	$9.63\%, 3.54$	$10.15\%, 3.82$
bc-wisc	$9.9\%, 2.11$	$9.93\%, 3.6$	$10.82\%, 5.43$	$8.28\%, 1.55$	$10.6\%, 3.14$
TRANSFER TO 60/40 ECOLI DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
yeast	–	-1.72%	17.07%	12.53%	35.96%
german	–	4.24%	18.59%	2.73%	-2.53%
bc-wisc	–	-0.3%	-9.29%	16.36%	-7.07%

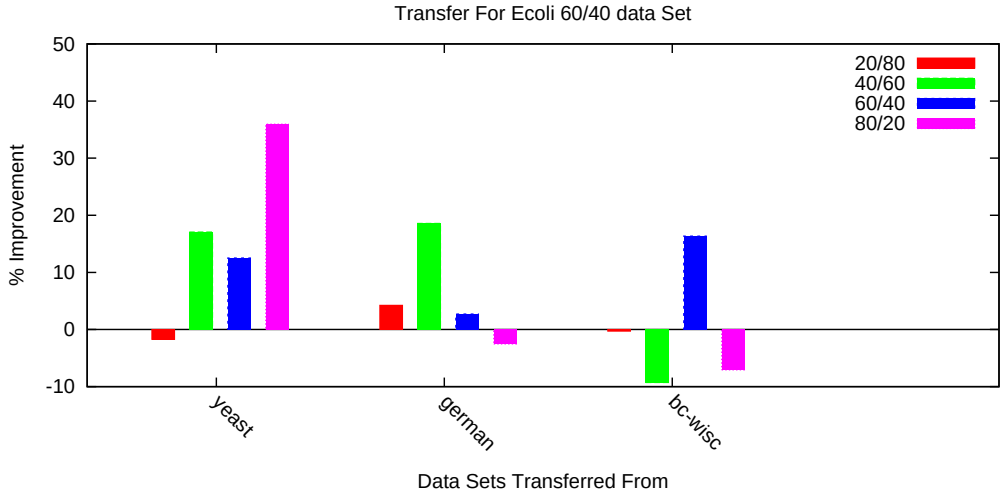

Figure 6.5: Percentage Improvement for the 60/40 ecoli data-set. Each color represents a particular type of the transfer-from data-set.

60/40 ecoli data-set: In the $60/40$ data-set type we for the first time observe ill-effects of transfer, as in half the cases we see reduction in performance. However, except in one case, the negative impact of transfer is not that severe.

Table 6.7: Results of 12 transfer experiments for the $80/20$ ecoli data set. See Table Key for meaning of the table entries.

TRANSFER TO 80/20 ECOLI DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
yeast	$10.89\%, 5.82$	$8.06\%, 3.28$	$7.61\%, 3.74$	$9.55\%, 4.18$	$8.96\%, 4.95$
german	$10.89\%, 5.82$	$10.15\%, 2.57$	$9.55\%, 4.63$	$11.04\%, 4.44$	$9.55\%, 5.14$
bc-wisc	$10.89\%, 5.82$	$8.51\%, 5.82$	$7.61\%, 3.16$	$8.66\%, 6.36$	$10.15\%, 2.89$
TRANSFER TO 80/20 ECOLI DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
yeast	–	25.99%	30.12%	12.3%	17.72%
german	–	6.8%	12.3%	-1.38%	12.3%
bc-wisc	–	21.85%	30.12%	20.48%	6.8%

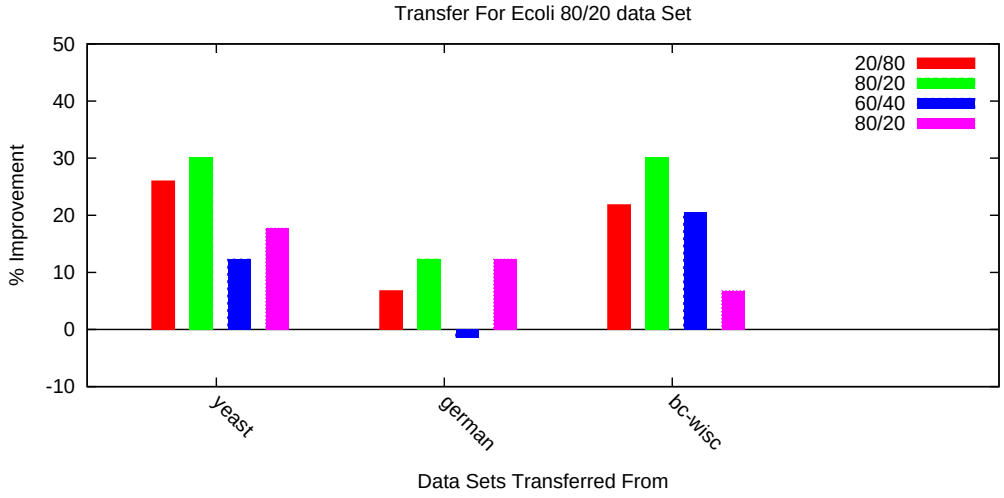

Figure 6.6: Percentage Improvement for the 80/20 ecoli data-set. Each color represents a particular type of the transfer-from data-set.

80/20 ecoli data-set: The results for the $80/20$ case is much better than the result for the $60/40$ case, where we see significant improvement in most cases, and a minor reduction in performance in only one case.

6.5.4 Results for the *bc-wisc* Dataset

Tables 6.9 to 6.12 and Figs. 6.7 to 6.6 shows the results of transfer learning experiments when the transfer-to data-set was ecoli with types $20/80$, $40/60$, $60/40$ and $80/20$ respectively. The transfer from data-sets were heart, aus and ecoli. The performance improvement here is not as notable as for ecoli, and in many cases there is a reduction in performance. However, as mentioned in Sect. 6.5.2, most of these reductions are not that significant. Strangely enough, the most significant improvement is observed for the 80/20 transfer-to data-set. So in this case it seems that the space of tree sizes that the MCMC algorithm is being told to explore in the transfer case is insufficient to overcome the paucity of data in the $x/(1-x)$ transfer-to data-types for $x < 80$. Not only that, the space being suggested seem to somewhat harmful in some cases. The tables and figures appear starting in the next page.

Table 6.8: **Table Key:** The *From Type* row gives the type of data-set information is being transferred from - *No-Trans* means no transfer is occurring, and $x/(100-x)$ mean the corresponding data-set type (see text for details). Each subsequent row gives the result when information is transferred from the corresponding data-set. The top half of the table gives the actual error rates and standard deviation, and the lower half gives the percentage improvement in each case.

TRANSFER TO $x/(1-x)$ TRANSFER-TO DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
data-set 1	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$
data-set 2	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$
data-set 3	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$
TRANSFER TO $x/(1-x)$ TRANSFER-TO DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/40*	*40/60*	*60/40*	*80/20*
data-set 1	–	9%	9%	9%	9%
data-set 2	–	9%	9%	9%	9%
data-set 3	–	9%	9%	9%	9%

Table 6.9: Results of 12 transfer experiments for the $20/80$ bc-wisc data set. See Table Key for the meaning of the table entries.

TRANSFER TO 20/80 BC-WISC DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
heart	$10.8\%, 3.1$	$9.52\%, 2.74$	$10.07\%, 2.01$	$10.21\%, 2.27$	$9.95\%, 1.95$
aus	$10.8\%, 3.1$	$10.47\%, 2.24$	$11.27\%, 2.55$	$9.71\%, 1.66$	$11.66\%, 2.13$
ecoli	$10.8\%, 3.1$	$8.91\%, 2.19$	$10.77\%, 2.67$	$9.55\%, 1.44$	$10.27\%, 2.14$
TRANSFER TO 20/80 BC-WISC DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
heart	–	11.85%	6.76%	5.46%	7.87%
aus	–	3.06%	-4.35%	10.09%	-7.96%
ecoli	–	17.5%	0.28%	11.57%	4.91%

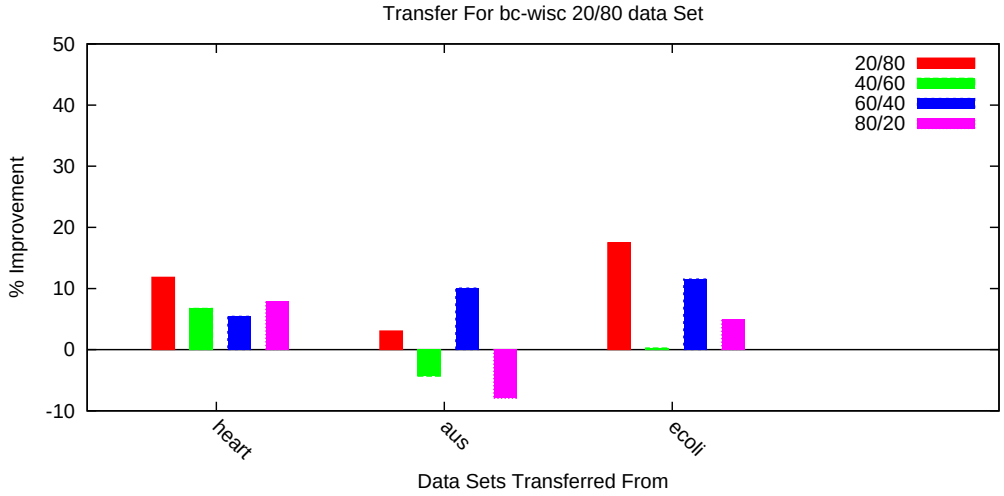

Figure 6.7: Percentage Improvement for the 20/80 bc-wisc data-set. Each color represents a particular type of the transfer-from data-set.

20/80 bc-wisc data-set: We mostly observe improvements in performance, but for the most part it is not that significant, and there are some insignificant reduction in performances.

Table 6.10: Results of 12 transfer experiments for the $40/60$ bc-wisc data set. See Table Key for the meaning of the table entries.

TRANSFER TO 40/60 BC-WISC DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
heart	$8.92\%, 1.01$	$8.97\%, 0.87$	$8.9\%, 2.42$	$9.02\%, 2.02$	$9.59\%, 1.37$
aus	$8.92\%, 1.01$	$9.71\%, 2.49$	$8.93\%, 1.87$	$7.76\%, 1.34$	$7.8\%, 1.13$
ecoli	$8.92\%, 1.01$	$8.59\%, 1.39$	$9.33\%, 3.63$	$9.19\%, 2.46$	$9.14\%, 1.04$
TRANSFER TO 40/60 BC-WISC DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
heart	–	-0.56%	0.22%	-1.12%	-7.51%
aus	–	-8.86%	-0.11%	13.0%	12.56%
ecoli	–	3.7%	-4.6%	-3.03%	-2.47%

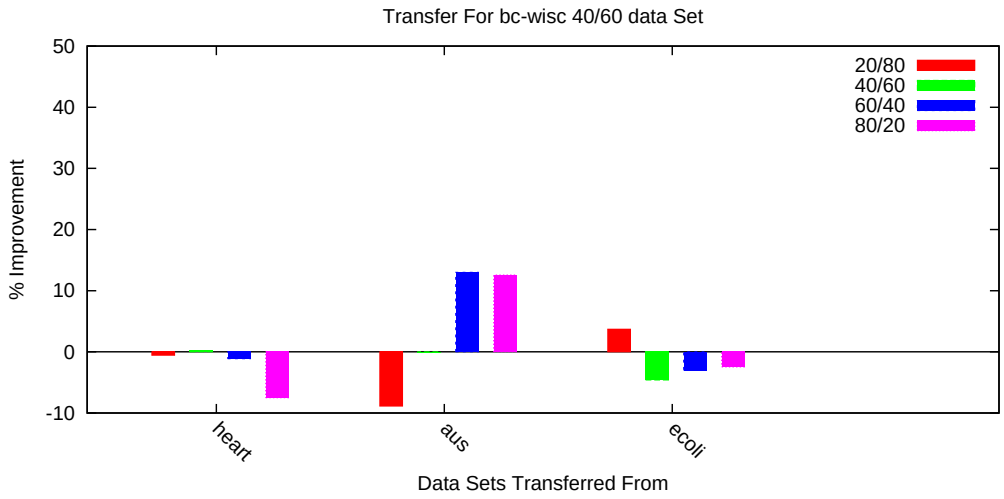

Figure 6.8: Percentage Improvement for the 40/80 bc-wisc data-set. Each color represents a particular type of the transfer-from data-set.

40/60 bc-wisc data-set: In this case, almost all the change in performances are reductions. There are two increase in performance, (aus, 60/40 and 80/20), that are just significant, but also two reductions in performance that are also nearly significant. In our entire collection of experiments this is the worst performing set – and yet there are only 2 nearly bad performances. So this is a good sign for our method.

Table 6.11: Results of 12 transfer experiments for the $60/40$ bc-wisc data set. See Table Key for the meaning of the table entries.

TRANSFER TO 60/40 BC-WISC DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
heart	$8.28\%, 1.93$	$8.32\%, 2.04$	$7.46\%, 1.4$	$7.53\%, 1.71$	$8.42\%, 1.54$
aus	$8.28\%, 1.93$	$7.67\%, 2.15$	$8.57\%, 2.2$	$8.28\%, 1.45$	$8.35\%, 1.59$
ecoli	$8.28\%, 1.93$	$8.49\%, 1.9$	$6.77\%, 3.07$	$7.56\%, 1.73$	$7.99\%, 1.72$
TRANSFER TO 60/40 BC-WISC DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
heart	–	-0.48%	9.9%	9.06%	-1.69%
aus	–	7.37%	-3.5%	0.0%	-0.85%
ecoli	–	-2.54%	18.24%	8.7%	3.5%

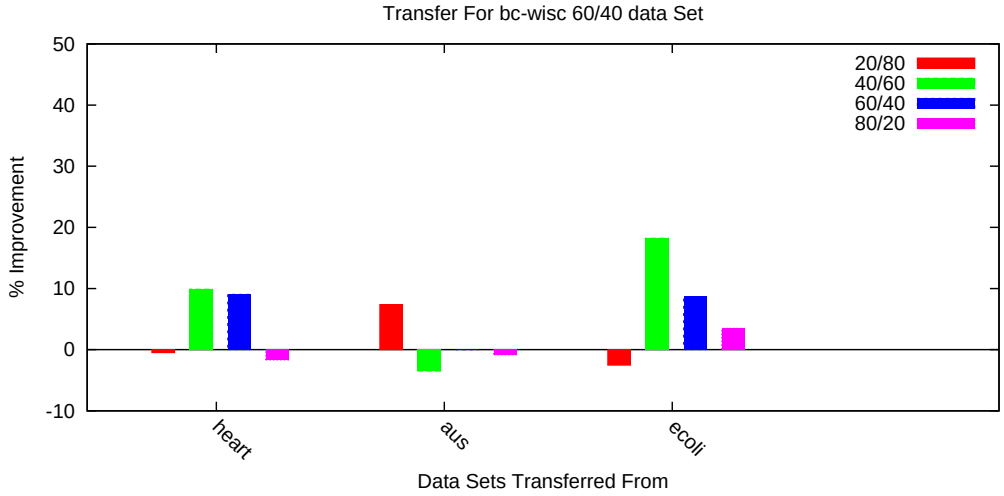

Figure 6.9: Percentage Improvement for the 60/40 bc-wisc data-set. Each color represents a particular type of the transfer-from data-set.

60/40 bc-wisc data-set: The results here are much better than in the preceding case, with no significant reductions in performance, and several significant improvements in performance.

Table 6.12: Results of 12 transfer experiments for the $80/20$ bc-wisc data set.

TRANSFER TO 80/20 BC-WISC DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
heart	$8.99\%, 3.03$	$7.19\%, 2.11$	$5.9\%, 1.81$	$6.76\%, 2.09$	$8.85\%, 2.47$
aus	$8.99\%, 3.03$	$8.2\%, 1.62$	$6.76\%, 1.96$	$7.91\%, 2.09$	$7.77\%, 2.47$
ecoli	$8.99\%, 3.03$	$9.57\%, 2.91$	$8.2\%, 2.85$	$6.12\%, 1.86$	$7.27\%, 2.58$
TRANSFER TO 80/20 BC-WISC DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
heart	–	20.02%	34.37%	24.81%	1.56%
aus	–	8.79%	24.81%	12.01%	13.57%
ecoli	–	-6.45%	8.79%	31.92%	19.13%

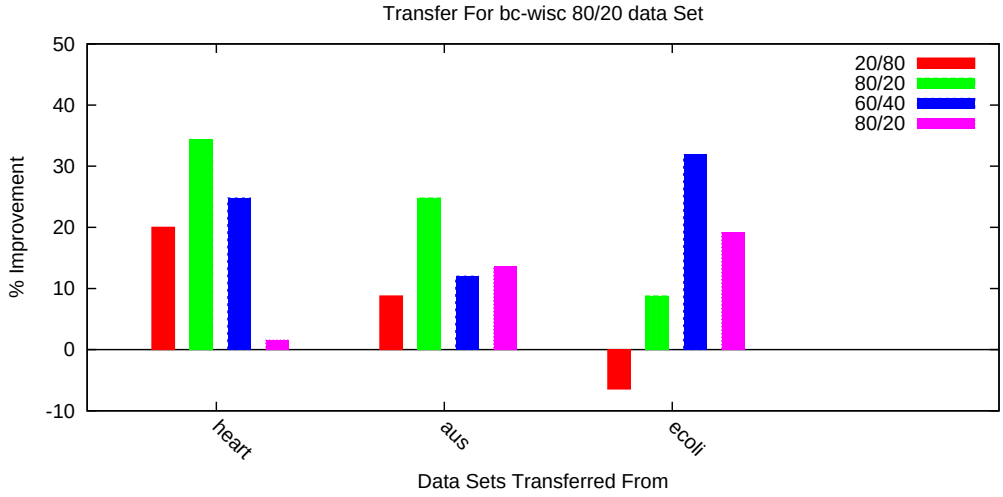

Figure 6.10: Percentage Improvement for the 80/20 bc-wisc data-set. Each color represents a particular type of the transfer-from data-set.

80/20 bc-wisc data-set: These are the best batch of results by far for the bc-wisc data-set with significant performance improvements in all cases, and with only reduction in performance.

6.5.5 Results for the *aus* Dataset

Tables 6.14 to 6.17 and Figs. 6.11 to 6.14 shows the results of transfer learning experiments when the transfer-to data-set was ecoli with types $20/80$, $40/60$, $60/40$ and $80/20$ respectively. The transfer from data-sets were german, ecoli and hep. This is the best performing transfer-to data-set by far, with no reductions in performance, and significant improvement in performance in all cases. The results are also intuitive in that we observe the most improvement in performance when training data is most scarce – i.e. in the transfer-to 20/80 case. The tables and figures appear starting in the next page.

Table 6.13: **Table Key:** The *From Type* row gives the type of data-set information is being transferred from - *No-Trans* means no transfer is occurring, and $x/(100-x)$ mean the corresponding data-set type (see text for details). Each subsequent row gives the result when information is transferred from the corresponding data-set. The top half of the table gives the actual error rates and standard deviation, and the lower half gives the percentage improvement in each case.

TRANSFER TO $x/(1-x)$ TRANSFER-TO DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
data-set 1	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$
data-set 2	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$
data-set 3	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$	$9\%, 9$
TRANSFER TO $x/(1-x)$ TRANSFER-TO DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/40*	*40/60*	*60/40*	*80/20*
data-set 1	–	9%	9%	9%	9%
data-set 2	–	9%	9%	9%	9%
data-set 3	–	9%	9%	9%	9%

Table 6.14: Results of 12 transfer experiments for the $20/80$ aus data set. See Table Key for the meaning of the table entries.

TRANSFER TO 20/80 AUS DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
german	$21.9\%, 4.03$	$15.49\%, 1.47$	$16.9\%, 3.65$	$15.6\%, 1.47$	$15.36\%, 1.27$
ecoli	$21.9\%, 4.03$	$14.8\%, 0.94$	$15.63\%, 1.91$	$15.47\%, 1.32$	$15.54\%, 1.46$
hep	$21.9\%, 4.03$	$14.93\%, 1.23$	$14.91\%, 1.72$	$15.22\%, 1.06$	$14.73\%, 1.0$
TRANSFER TO 20/80 AUS DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
german	–	29.27%	22.83%	28.77%	29.86%
ecoli	–	32.42%	28.63%	29.36%	29.04%
hep	–	31.83%	31.92%	30.5%	32.74%

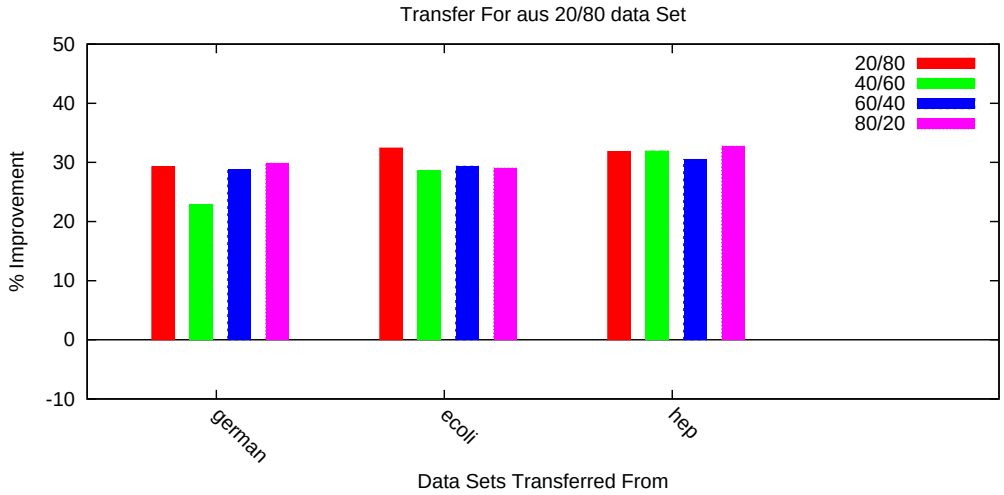

Figure 6.11: Percentage Improvement for the 20/80 aus data-set. Each color represents a particular type of the transfer-from data-set.

20/80 aus data-set: We see significant performance in all cases.

Table 6.15: Results of 12 transfer experiments for the $40/60$ aus data set. See Table Key for the meaning of the table entries.

TRANSFER TO 40/60 AUS DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
german	$18.55\%, 2.25$	$14.76\%, 0.71$	$14.37\%, 1.35$	$14.28\%, 0.96$	$15.07\%, 0.87$
ecoli	$18.55\%, 2.25$	$14.49\%, 0.95$	$14.54\%, 1.29$	$15.07\%, 0.9$	$15.05\%, 0.99$
hep	$18.55\%, 2.25$	$14.15\%, 0.82$	$14.47\%, 0.91$	$15.24\%, 1.16$	$15.8\%, 3.34$
TRANSFER TO 40/60 AUS DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
german	–	20.43%	22.53%	23.02%	18.76%
ecoli	–	21.89%	21.62%	18.76%	18.87%
hep	–	23.72%	21.99%	17.84%	14.82%

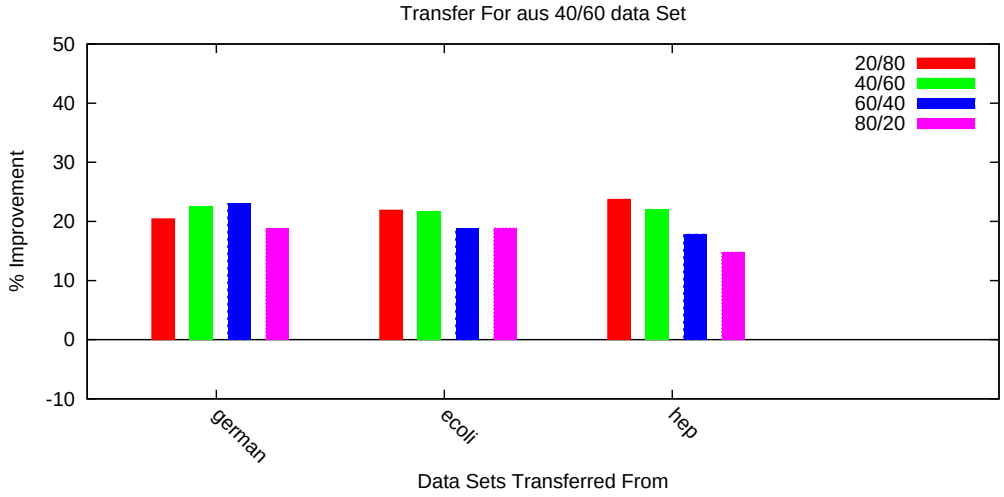

Figure 6.12: Percentage Improvement for the 40/60 aus data-set. Each color represents a particular type of the transfer-from data-set.

40/60 aus data-set: The performance improvement not as significant in the preceding case, but still quite significant.

Table 6.16: Results of 12 transfer experiments for the $60/40$ aus data set. See Table Key for the meaning of the table entries.

TRANSFER TO 60/40 AUS DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
german	$20.9\%, 3.03$	$15.33\%, 2.76$	$14.71\%, 1.87$	$14.93\%, 1.49$	$13.88\%, 1.77$
ecoli	$20.9\%, 3.03$	$14.24\%, 1.47$	$13.55\%, 1.48$	$13.77\%, 1.64$	$14.24\%, 1.76$
hep	$20.9\%, 3.03$	$15.4\%, 1.69$	$14.86\%, 1.87$	$14.75\%, 1.23$	$14.64\%, 1.63$
TRANSFER TO 60/40 AUS DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
german	–	26.65%	29.62%	28.56%	33.59%
ecoli	–	31.87%	35.17%	34.11%	31.87%
hep	–	26.32%	28.9%	29.43%	29.95%

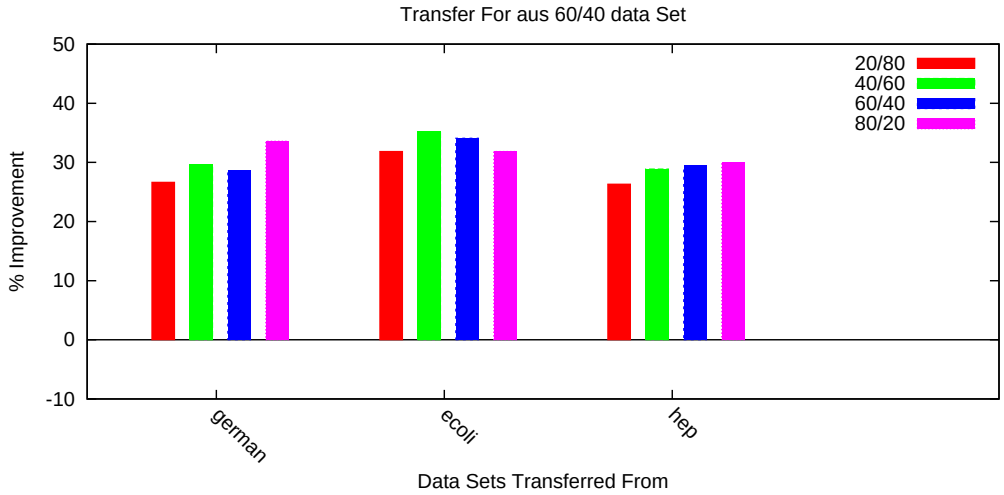

Figure 6.13: Percentage Improvement for the 60/40 aus data-set. Each color represents a particular type of the transfer-from data-set.

60/40 aus data-set: We observe significant performance improvement in all cases.

Table 6.17: Results of 12 transfer experiments for the $80/20$ aus data set. See Table Key for the meaning of the table entries.

TRANSFER TO 80/20 AUS DATA SET – ERROR RATES					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
german	$18.99\%, 2.9$	$13.94\%, 2.22$	$14.45\%, 2.5$	$13.87\%, 1.73$	$14.31\%, 3.4$
ecoli	$18.99\%, 2.9$	$13.43\%, 3.47$	$14.53\%, 2.89$	$15.91\%, 3.34$	$14.31\%, 2.4$
hep	$18.99\%, 2.9$	$15.62\%, 2.6$	$13.5\%, 4.38$	$15.04\%, 2.43$	$15.04\%, 2.1$
TRANSFER TO 80/20 AUS DATA SET – PERCENTAGE IMPROVEMENTS					
From Type	*No-Trans*	*20/80*	*40/60*	*60/40*	*80/20*
german	–	26.59%	23.91%	26.96%	24.64%
ecoli	–	29.28%	23.49%	16.22%	24.64%
hep	–	17.75%	28.91%	20.8%	20.8%

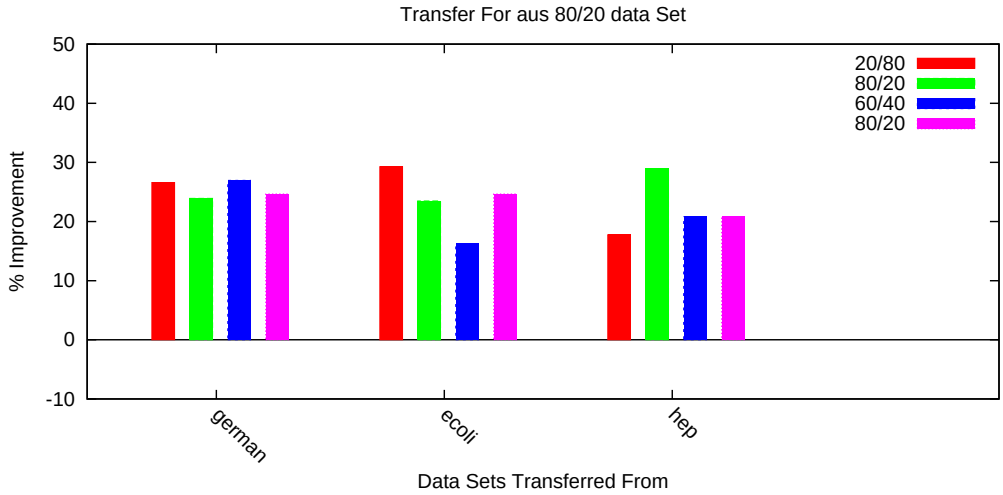

Figure 6.14: Percentage Improvement for the 80/20 aus data-set. Each color represents a particular type of the transfer-from data-set.

80/20 aus data-set: We observe significant performance improvement in all cases.

6.6 Discussion

In this section we showed how we may approximate our optimal sequential transfer prior ξ_{TL} in a practical setting. We approximated the prior for practical Bayesian learning using decision trees and showed that these experiments more or less hew closely to theoretical predictions. That is, in a battery of 144 individual transfer experiments, in most cases we see significant improvement due to transfer and only in a couple of experiments out of 144 we see significant reduction in performance. This shows our approximation, which are admittedly crude, still result in interesting practical performance. We believe that this demonstrates the power of the theory developed in preceding chapters.

While we have performed a whole slew of successful transfer experiments, there are avenues of experimentation we have not yet explored. Due to the general nature of our method, we can perform transfer experiments between any combination of databases in the UCI repository and in the future it will be interesting to perform these experiments. Additionally, our approximations, while effective in practice, are not as sophisticated as they could be, and so in future we also wish to explore transfer using more powerful generalized similarity functions like the gzip compressor as in Cilibrasi & Vitanyi, 2005. A flavor of this approach: if the standard compressor is gzip, then the function $C_{gzip}(xy)$ will give the length of the string xy after compression by gzip. $C_{gzip}(xy) - C_{gzip}(y)$ will be the conditional $C_{gzip}(x|y)$. So $C_{gzip}(\mathbf{h}|\mathbf{h}')$ will give the relatedness between tasks. The most promising avenue of research in this direction seems to be to restrict ourselves to group of specific machine learning domains and then deriving compression based distance functions suitable for measuring relatedness between hypothesis that are suitable for the group.

Chapter 7

Conclusion

We will end this dissertation with a look at the contributions made in this thesis, how this work can be extended in the future, and finally, a brief look at the connection of the ideas in this thesis to work in cognitive science in trying to understand how humans measure similarity.

7.1 Contributions of this Thesis

We began this thesis by pointing out that while transfer learning is an important subfield of machine learning, key problems in it remain formally unsolved. In particular, it was not clear how we should measure similarity between tasks, and this lead to problems of not knowing when to transfer information, how much information to transfer and when not to. In this thesis, with the aid of ideas in Algorithmic Information Theory, we gave a formally/universally optimal measure of task relatedness. We then used this measure to derive universally optimal transfer learning schemes in the Bayesian setting. Universal optimality means that no other reasonable methods can do much better than the schemes we describe, and hence in a very formal sense, our methods solved the key problems in transfer learning that we mentioned above. We further extended our theory to the Artificial Agents setting and Prediction with Expert Advice setting.

As a byproduct of the above investigation, we derived interesting results in Algorithmic Information Theory itself. We extended the theory of Information Distance and gave a new, more robust, interpretation of classic universality results in AIT. Furthermore, we used information measures for strings to measure information content of programs computing distributions. To allay concern that we are not losing something in this process, we also briefly developed the theory of Kolmogorov complexity of functions, and showed that these two are equivalent under certain reasonable conditions.

Interestingly, we were also able to construct a practical approximation of our theoretical methods. Using this, we performed 144 individual transfer experiments to successfully transfer across 7 real-life databases from the UCI repository that

have little to no semantic similarity. This made our experiments the most general transfer experiments to date.

7.2 Future Work

There are many directions for possible future work. As we mentioned in the body of the thesis, the theoretically optimal measures and methods that we introduce are computable only in the limit. So a major thrust of the future theoretical work will be in developing a practical version of the theory. There are couple of different ways to approach this. The first is to focus on specific machine learning domains, such as systems biology, machine vision etc. and develop transfer method tailored to transfer within such a domain, or to transfer across a certain class of domains and so forth. While we expect this to lead to interesting practical applications, from a formal perspective this seems somewhat unsatisfactory as a main point of interest of our research was that we were able to transfer across arbitrary domains.

The other more interesting option is to restrict the class of measures and transfer schemes we consider from computable-in-the-limit to those that are resource boundedly computable. That is, we only consider probability measures and distance functions such that there exists programs that compute them while respecting some given time and memory usage constraint; then we try to find transfer learning distances and transfer methods that are universally optimal with respect to this class. This framework we term resource bounded learning, and this is a very rich area in machine learning that needs to be explored both in a single task learning case and transfer learning case. Current work in this include Feder & Federovski, 1998; Rajwan & Feder, 2000; Meron & Feder, 2004. The results, particularly the last paper cited, are impressive. These papers considers sequence prediction problem and the most impressive results give asymptotic regret bounds for the best K-state finite state machines competing against order L Markov chains. The way we envision extending this work is via considering the Bayesian setting but with resource bounded, computable measures (which is obviously a larger class than those representable by K-state FSMs or order L Markov chains), and deriving t-step bounds rather than asymptotic ones.

Furthermore, we just barely touched on applying our transfer learning scheme to the artificial intelligent agent setting. We plan on exploring this issue further by focusing on Bayesian reinforcement learning agents (Dearden et al., 1998; Sterns, 2000; Ross et al., 2007).

We expect future practical work to largely come out as applications of the theory to be developed. However, the decision tree based transfer method we developed in this thesis is also quite general, and it would be interesting to perform more

experiments with this method to establish its applicability.

7.3 Similarity Measures in Human Cognition

We began this thesis by observing that study of transfer learning was motivated by the fact that when people solve problems they almost always use transfer. So it is appropriate that we end this document by looking at the question of how cognitively plausible our approach to measuring similarity is. That is, do people use something similar to Kolmogorov complexity to measure similarity across tasks ?

While the answer is not known for the general case, there has been some work that postulates (Feldman, 2003) and gives evidence (Hahn et al., 2003) that people do use something similar to Kolmogorov complexity to measure similarity between concepts for the purpose of categorization. In the Cognitive science literature there are two main hypothesis about how people measure similarity; the first is based on a distance function in some psychological space (Shepard, 1957), and the second based on how many features the entities being compared have in common (Tversky, 1977). Both suffer from the severe limitation that objects are represented as points in a space or purely in terms of feature sets (i.e. no notion of structure in the representation is allowed). Due to these limitations, the notion of *transform functions* were proposed as a measure of similarity (Hahn et al., 2003) . That is, given two concepts, the measure of similarity between two concepts is the number of transforms that need to be applied to convert one concept to the other. This is, of course, a practical approximation to the Information Distance and evidence for use of this idea in people was given in Hahn et al., 2003. The connection between this and Kolmogorov complexity was elucidated in Chater & Vitanyi, 2002; Chater & Vitanyi, 2003. This connection is quite gratifying and exciting, as it seems that the ideas that lead us to formal solutions to problems of measuring similarity in machines may also hold the key to mysterious and deep question of how people so successfully measure and similarity between mental concepts – the very fact that initiated this whole work!

References

Abu-Mostafa, Y. S. (1995). Hints. *Neural Computation*, *7*, 639–671.

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*.

Andrieu, C., de Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, *50(1-2)*, 5–43.

Barto, A., Bradtke, S., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, *72(1)*, 81–138.

Baxter, J. (1995). Learning internal representations. *Proceeding of the workshop on Computational Learning Theory*.

Baxter, J. (1998). *Learning to learn*, chapter Theoretical Models of Learning to Learn, 71–94. MA: Kluwer Academic Publishers.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, *12*, 149–198.

Behrends, E. (2000). *Introduction to markov chains: With special emphasis on rapid mixing*. Berlin: Vieweg Verlag.

Ben-David, S., Gehrke, J., & Schuller, R. (2002). A theoretical framework for learning from a pool of disparate data sources. *ACM SIGKDD International conference on Knowledge discovery and data mining*, *8*, 443–449.

Ben-David, S., & Schuller, R. (2003). Exploiting task relatedness for learning multiple tasks. *Proceedings of the* 16^{th} *Annual Conference on Learning Theory*.

Bennett, C., Gacs, P., Li, M., Vitanyi, P., & Zurek, W. (1998). Information distance. *IEEE Transactions on Information Theory*, *44(4)*, 1407–1423.

Bernardo, J. M., & Smith, A. F. M. (1994). *Bayesian theory*. New York: Wiley.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1993). *Classification and regression trees*. New York: Chapman and Hall.

Caruana, R. (1993). Multitask learning: A knowledge-based of source of inductive bias. *Proceedings of the* 10^{th} *International Conference on Machine Learning*.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning and games*. Cambridge University Press. 1st edition.

Chaitin, G. J. (1975). A theory of program size formally identical to information theory. *Journal if the ACM*, *22(3)*, 329–340.

Chater, N., & Vitanyi, P. (2002). Simplicity: A unifying principle in cognitive science? *Trends in Cognitive Sciences*, *7*, 19–22.

Chater, N., & Vitanyi, P. (2003). The generalized universal law of generalization. *Journal of Mathematical Psychology*, *47*, 346–369.

Cilibrasi, R., & Vitanyi, P. (2005). Clustering by compression. *IEEE Transactions on Information theory*, *51(4)*, 1523–1545.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. Wiley-Interscience.

Dearden, R., Friedman, N., & Russell, S. (1998). Bayesian q-learning. *Proceedings of* 15^{th} *National Conference on Artificial Intelligence (AAAI)*. AAAI Press, Menlo Park, CA.

Denison, D. G. T., Holmes, C. C., Mallick, B. K., & Smith, A. F. M. (2005). *Bayesian methods for nonlinear classification and regression*. England: Wiley.

Drummond, C. (2002). Accelerating reinforcement learning by composing solutions of automatically identified subtasks. *Journal of Artificial Intelligence*, *16*, 59–104.

Evgeniou, T., Micchelli, C., , & Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics*, *13*, 1–50.

Evgeniou, T., Micchelli, C., , & Pontil, M. (2004). Learning multiple tasks with kernel methods. *Conference on Knowledge Discovery and Data Mining*, *10*.

Evgeniou, T., Micchelli, C., , & Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, *6*, 615–637.

Feder, M., & Federovski, E. (1998). Prediction of binary sequences using finite memory. *Proceedings of the International Symposium on Information Theory*.

Feldman, J. (2003). The simplicity principle in human concept learning. *Current Directions in Psychological Science*, *12(6)*, 227–232.

Fill, J. A. (1998). An interruptible algorithm for perfect sampling via Markov chains. *The Annals of Applied Probability*, *8(1)*, 131–162.

Fitting, M. (1996). *First order automated theorem proving*. Berlin: Springer. 2nd edition.

Friedman, N., & Singer, Y. (1998). Efficient bayesian paramter estimation in large discrete domains. *Proceedings of* 13^{th} *Neural Information Processing Systems Conference*.

Gacs, P. (1974). On the symmetry of algorithmic information. *Soviet Mathematics Doklady, 15*, 1477–1480.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. (1996). *Markov chain monte carlo in practice*. London: Chapman and Hall.

Grunwald, P., & Vitanyi, P. (2004). Shannon information and Kolmogorov complexity. *Submitted to IEEE Transactions on Information Theory*.

Häggström, O. (2002). *Finite Markov chains and algorithmic applications*. Cambridge University Press.

Hahn, U., Chater, N., & Richardson, L. B. C. (2003). Similarity as transformation. *Cognition, 87*.

Hutter, M. (2002). The fastest and shortest algorithm for all well defined problems. *International Journal of Foundations of Computer Science, 13(3)*, 431–443.

Hutter, M. (2003). Optimality of Bayesian universal prediction for general loss and alphabet. *Journal of Machine Learning Research, 4*, 971–1000.

Hutter, M. (2004). *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Berlin: Springer-Verlag.

Jebara, T. (2004). Multi-task feature and kernel selection for svms. *Proceedings of the* 21^{st} *International Conference on Machine Learning, (11)*.

Juba, B. (2006). Estimating relatedness via data compression. *Proceedings of the* 23^{rd} *International Conference on Machine Learning*.

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems of Information and Transmission, 1(1)*, 1–7.

Langley, P., & Rogers, S. (2004). Cumulative learning of hierarchical skills. *Proceedings of the Third International Conference on Development and Learning*.

Levin, L. A. (1973). Universal sequential search problem. *Problems of Information and Transmission, 9(3)*, 265–266.

Levin, L. A. (1974). Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Problems of Information and Transmission, 10*, 206–210.

Li, M., Chen, X., Ma, B., & Vitanyi, P. (2004). The similarity metric. *IEEE Transactions on Information Theory, 50(12)*, 3250–3264.

Li, M., & Vitanyi, P. (1997). *An introduction to Kolmogorov complexity and its applications*. New York: Springer-Verlag. 2nd edition.

Littlestone, N., & Warmuth, M. K. (1987). The weighted majority algorithm. *Annual Symposium on the Foundations of Computer Science, 30*, 256–261.

Mackay, D. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press. 1st edition.

Mahmud, M. M. H. (2007). On universal transfer learning. *Proceedings of the* 18^{th} *International Conference on Algorithmic Learning Theory*.

Mahmud, M. M. H., & Ray, S. (2007). Transfer learning using Kolmogorov complexity:basic theory and empirical evaluations. *Proceedings of the* 21^{st} *Neural Information Processing Systems Conference*.

McGovern, A. (2002). *Autonmous discovery of temporal abstractions from interactions with an environment*. Doctoral dissertation, University of Massachusetts.

Mehta, N., Natarajan, S., Tadepalli, P., & Fern, A. (2005). Transfer in variable reward hierarchical reinforcement learning. *Workshop on Inductive Transfer,* 19^{th} *Neural Information Processing Systems Conference*.

Meron, E., & Feder, M. (2004). Finite-memory universal prediction of individual sequences. *IEEE Transactions on Information Theory, 50(7)*, 1506–1523.

Mihalkova, L., Huynh, T., & Mooney, R. (2007). Mapping and revising markov logic networks for transfer learning. *Proceedings of the* 22^{nd} *National Conference on Artificial Intelligence (AAAI)*.

Mitchell, M. (1996). *An introduction to genetic algorithmis*. Cambridge, MA: MIT Press.

Mitchell, T. M., & Thrun, S. B. (1993). Explanation-based neural network learning for robot control. *Adavances in Neural Information Processing Systems* (pp. 287–294). San Mateo, CA: Morgan Kaufmann Press.

Neal, R. M. (2004). Bayesian methods for machine learning, NIPS tutorial.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newman, D., Hettich, S., Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

O'Quinn, R., Silver, D., & Poirier, R. (2005). Continued practice and consolidation of a learning task. *Proceedings of the Meta-Learning Workshop, 22nd International Conference on Machine Learning*.

Pratt, L. (1991). Discriminability-based transfer between neural networks. *AAAI* (pp. 204–211). Morgan Kaufmann.

Pratt, L. (1992). Discriminability-based transfer between neural networks. *Advances in Neural Information Processing Systems 5* (pp. 204–211). Morgan Kaufmann.

Propp, J. G., & Wilson, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms, 9(1,2)*, 223–252.

Rajwan, D., & Feder, M. (2000). Universal finite memory machines for coding binary sequences. *Proceedings of the Data Compression Conference*.

Richardson, M., & Domingos, P. (2002). Markov logic networks. *Machine Learning, 62*, 07136.

Robert, C. P., & Casella, G. (2005). *Monte carlo statistical methods*. Berling: Springer.

Ross, S., Chaib-draa, B., & Pineau, J. (2007). Bayes adaptive POMDP. *Proceedings of the* 20^{th} *Conference on Neural Information Processing Systems*.

Russell, S. J., & Norvig, P. (2003). *Artificial intelligence: A modern approach*. Upper Saddle River, NJ: Prentice Hall. 2nd edition.

Schapire, R. E. (1997). Selective transfer of task knowledge using stochastic noise. *Advances in Artificial Intelligence, Conference of the Canadian Society for Computational Studies of Intelligence, 16*, 190–205.

Schmidhuber, J. (1994). *On learning how to learn learning strategies* (Technical Report FKI-198-94). Fakultat Fur Informatik, Technische Universitat Munchen.

Schmidhuber, J. (2004). Optimal ordered problem solver. *Machine Learning, 54*, 211–254.

Schmidhuber, J. (2006). Goedel machines: self-referential universal problem solvers making provably optimal self-improvements. *Aritifical General Intelligence, 54*, 119–226.

Shepard, R. N. (1957). Stimulus and response generalization: a stochastic model relating generalization to distance in psychological space. *Psychometrika, 22*, 325–345.

Shoenfield, J. (1967). *Mathematical logic*. Menlo Park: Addsion-Wesley. 1st edition.

Silver, D., & McCracken, P. (2003). Selective transfer of task knowledge using stochastic noise. *Advances in Artificial Intelligence, 16th Conference of the Canadian Society for Computational Studies of Intelligence*, 190–205.

Silver, D., & McCracken, R. (2001). Selective functional transfer: Inductive bias from related tasks. *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*, 182–189.

Silver, D., & McCracken, R. (2002). The task rehearsal method of life-long learning: Overcoming impoverished data. *Advances in Artificial Intelligence, 15th Conference of the Canadian Society for Computational Studies of Intelligence*, 90–101.

Silver, D., & Mercer, R. (1996). The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science, 8(2)*, 277–294.

Silver, D. L., & Poirier, R. (2004). Sequential consolidation of learned task knowledge. *Proceedings of the Seventeenth Canadian Conference on Artificial Intelligence*.

Simard, P., Victorri, B., LeCun, Y., & Denker, J. (1992). Tangent prop - a formalism for specifying selected invariances in an adaptive network. *Advances in Neural Information Processing Systems, 4)*, 895–903.

Singh, S., Barto, A., & Chentanez, N. (2004a). Intrinsically motivated reinforcement learning. *Advances in Neural Information Processing Systems, 16*.

Singh, S., Barto, A., & Chentanez, N. (2004b). Intrinsically motivated reinforcement learning of a hierarchical collection of skills. *Internation Conference on Developmental Learning, 3*.

Singh, S. P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning, 8*, 323–339.

Solomonoff, R. J. (1964a). A formal theory of inductive inference: Part 1. *Information and Control, 7(1)*, 1–22.

Solomonoff, R. J. (1964b). A formal theory of inductive inference: Part 2. *Information and Control, 7(2)*, 224–254.

Solomonoff, R. J. (1978). Complexity-based induction systems: comparisons and convergence theorems. *IEEE Transactions on Information Theory, 24(4)*, 422–432.

Sterns, M. (2000). A bayesian framework for reinforcement learning. *Proceedings of the* 17^{th} *International Conference on Machine Learning*.

Swarup, S., & Ray, S. R. (2006). Cross domain knowledge transfer using structured representations. *Proceedings of the* 21^{st} *National Conference on Artificial Intelligence (AAAI)*.

Taylor, M., & Stone, P. (2007). Cross-domain transfer for reinforcement learning. *Proceedings of the* 24^{th} *International Conference on Machine Learning*.

Thrun, S. (1995). *Lifelong learning: A case study* (Technical Report CMU-CS-95-208). Computer Science Department, Carnegie Mellon University.

Thrun, S., & Mitchell, T. (1995). Lifelong robot learning. *Robotics and Autonomous Systems, 15*, 25–46.

Thrun, S., & O'Sullivan, J. (1996). Discovering structure in multiple learning tasks:the TC algorithm. *Proceedings of the* 13^{th} *International Conference on Machine Learning*.

Thrun, S., & Pratt, L. Y. (Eds.). (1998). *Learning to learn*. Boston, MA: Kluwer Academic Publishers.

Tversky, A. (1977). Features of similarity. *Psychological Review*, *84*, 327–352.

Utgoff, P. E., & Stracuzzi, D. J. (2002). Many-layered learning. *Neural Computation, 14(10)*.

Veloso, M., Carbonell, J., Perez, A., Borrajo, D., & Fink, E. (1995). Integrating planning and learning: The prodigy architecture. *Journal of Experimental and Theoretical Artificial Intelligence, 7(1)*.

Vilalta, R., & Drissi, Y. (2001). Research directions in meta-learning. *Proceedings of the International Conference on Artificial Intelligence*. Las Vegas, Nevada, USA.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*, 77–95.

Vovk, V. (1990). Aggregating strategies. *Proceedings of the* 3^{rd} *Internation Conference on Computational Learning Theory*.

Vovk, V. (2001). Competitive online statistics. *International Statistics Review, 69*, 213–248.

Wilson, A., Fern, A., Ray, S., & Tadepalli, P. (2007). Multi-task reinforcement learning: A hierarchical bayesain approach. *In Proceedings of the* 24^{th} *International Conference on Machine Learning*.

Zvonkin, A. K., & Levin, L. A. (1970). The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Math. Surveys, 25(6)*, 83–124.

Author's Biography

M. M. Hassan Mahmud was born the same year Solomonoff published his celebrated convergence theorem that is the basis for this very dissertation. He met people, made friends and learned new things, and carried on in this vein, eventually completing his O'Levels and A'Levels. During this period he came across and became fascinated with computers and making them to do things that they previously were not able to do. This resulted in him flying to the opposite side of the world to the USA and pursuing and obtaining a B.S. in Computer Science in 2000 from Stevens Institute of Technology. During his stay at Stevens, he came to see making computers to do things that others have already made them do before, as grunt work. The solution was not to abandon CS, but M.S. – specifically in Artificial Intelligence where one makes computers learn what they need to do, a – dare we say it – paradigm shift from just spelling out what computers should be computing. After completing his M.S., he was sufficiently foolhardy to take the next step and pursue a Ph.D. in AI and he completed this in 2008, the evidence for which is the dissertation you now hold in your hands/see on your screen. He is planning on pursuing a Post-Doc in AI somewhere nice.