To Transfer or Not To Transfer

Michael T. Rosenstein, Zvika Marx, Leslie Pack Kaelbling
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
{mtr,zvim,lpk}@csail.mit.edu

Thomas G. Dietterich
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97331
tgd@cs.orst.edu

Abstract

With transfer learning, one set of tasks is used to bias learning and improve performance on another task. However, transfer learning may actually hinder performance if the tasks are too dissimilar. As described in this paper, one challenge for transfer learning research is to develop approaches that detect and avoid negative transfer using *very little data* from the target task.

1 Introduction

Transfer learning involves two interrelated learning problems with the goal of using knowledge about one set of tasks to improve performance on a related task. In particular, learning for some target task—the task on which performance is ultimately measured—is influenced by inductive bias learned from one or more auxiliary tasks, e.g., [1, 2, 8, 9]. For example, athletes make use of transfer learning when they practice fundamental skills to improve training in a more competitive setting.

Even for the restricted class of problems addressed by supervised learning, transfer can be realized in many different ways. For instance, Caruana [2] trained a neural network on several tasks simultaneously as a way to induce efficient internal representations for the target task. Wu and Dietterich [9] showed improved image classification by SVMs when trained on a large set of related images but relatively few target images. Sutton and McCallum [7] demonstrated effective transfer by "cascading" a class of graphical models, with the prediction from one classifier serving as a feature for the next one in the cascade. In this paper we focus on transfer using hierarchical Bayesian methods, and elsewhere we report on transfer using learned prior distributions over classifier parameters [5].

In broad terms, the challenge for a transfer learning system is to learn what knowledge should be transferred and how. The emphasis of this paper is the more specific problem of deciding when transfer should be attempted for a particular class of learning algorithms. With no prior guarantee that the auxiliary and target tasks are sufficiently similar, an algorithm must use the available data to guide transfer learning. We are particularly interested in the situation where an algorithm must detect, perhaps implicitly, that the inductive bias learned from the auxiliary tasks will actually hurt performance on the target task.

In the next section, we describe a "transfer-aware" version of the naive Bayes classification algorithm. We then illustrate that the benefits of transfer learning depend, not surprisingly, on the similarity of the auxiliary and target tasks. The key challenge is to identify harmful transfer with *very few training examples* from the target task. With larger amounts of "target" data, the need for auxiliary training becomes diminished and transfer learning becomes unnecessary.

2 Hierarchical Naive Bayes

The standard naive Bayes algorithm—which we call *flat naive Bayes* in this paper—has proven to be effective for learning classifiers in *non-transfer* settings [3]. The flat naive Bayes algorithm constructs a separate probabilistic model for each output class, under the "naive" assumption that each feature has an independent impact on the probability of the class. We chose naive Bayes not only for its effectiveness but also for its relative simplicity, which facilitates analysis of our hierarchical version of the algorithm. Hierarchical Bayesian models, in turn, are well suited for transfer learning because they effectively combine data from multiple sources, e.g., [4].

To simplify our presentation we assume that just two tasks, *A* and *B*, provide sources of data, although the methods extend easily to multiple *A* data sources. The flat version of naive Bayes merges all the data without distinction, whereas the hierarchical version constructs two ordinary naive Bayes models that are coupled together. Let θ_i^A and θ_i^B denote the i-th parameter in the two models. Transfer is achieved by encouraging θ_i^A and θ_i^B to have similar values during learning. This is implemented by assuming that θ_i^A and θ_i^B are both drawn from a common hyperprior distribution, P_i, that is designed to have unknown mean but small variance. Consequently, at the start of learning, the values of θ_i^A and θ_i^B are unknown, but they are constrained to be similar.

As with any Bayesian learning method, learning consists of computing posterior distributions for all of the parameters in the two models, including the hyperprior parameters. The overall model can "decide" that two parameters are very similar (by decreasing the variance of the hyperprior) or that two other parameters are very different (by increasing the variance of the hyperprior). To compute the posterior distributions, we developed an extension of the "slice sampling" method introduced by Neal [6].

3 Experiments

We tested the hierarchical naive Bayes algorithm on data from a meeting acceptance task. For this task, the goal is to learn to predict whether a person will accept an invitation to a meeting given information about (a) the current state of the person's calendar, (b) the person's roles and relationships to other people and projects in his or her world, and (c) a description of the meeting request including time, place, topic, importance, and expected duration.

Twenty-one individuals participated in the experiment: eight from a military exercise and 13 from an academic setting. Each individual supplied between 99 and 400 labeled examples (3966 total examples). Each example was represented as a 15-dimensional feature vector that captured relational information about the inviter, the proposed meeting, and any conflicting meetings. The features were designed with the meeting acceptance task in mind but were not tailored to the algorithms studied. For each experiment, a single person was

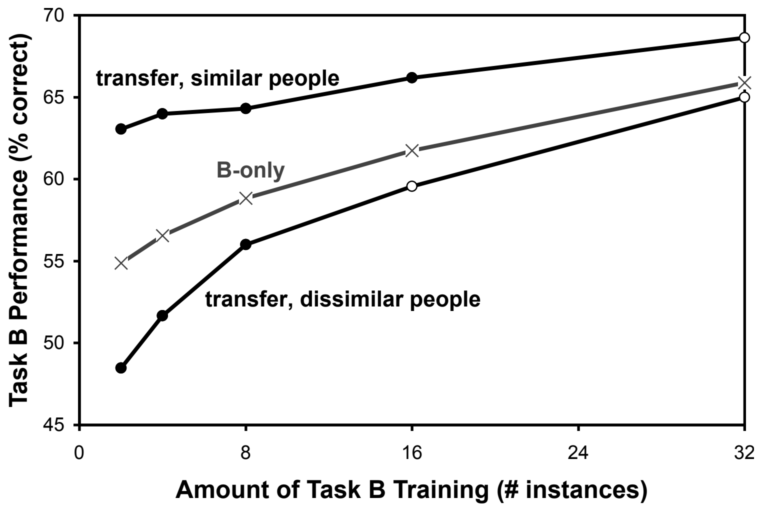

Figure 1: Effects of *B* training set size on performance of the hierarchical naive Bayes algorithm for three cases: no transfer ("B-only") and transfer between similar and dissimilar individuals. In each case, the same person served as the *B* data source. Filled circles denote statistically significant differences ($p<0.05$) between the corresponding transfer and B-only conditions.

chosen as the target (*B*) data source; 100 of his or her examples were set aside as a holdout test set, and from the remaining examples either 2, 4, 8, 16, or 32 were used for training. These training and test sets were disjoint and stratified by class. All of the examples from one or more other individuals served as the auxiliary (*A*) data source.

Figure 1 illustrates the performance of the hierarchical naive Bayes algorithm for a single *B* data source and two representative *A* data sources. Also shown is the performance for the standard algorithm that ignores the auxiliary data (denoted "B-only" in the figure). Transfer learning has a clear advantage over the B-only approach when the *A* and *B* data sources are similar, but the effect is reversed when *A* and *B* are too dissimilar.

Figure 2a demonstrates that the hierarchical naive Bayes algorithm almost always performs at least as well as flat naive Bayes, which simply merges all the available data. Figure 2b shows the more interesting comparison between the hierarchical and B-only algorithms. The hierarchical algorithm performs well, although the large gray regions depict the many pairs of dissimilar individuals that lead to negative transfer. This effect diminishes—along with the positive transfer effect—as the amount of *B* training data increases. We also observed qualitatively similar results using a transfer-aware version of the logistic regression classification algorithm [5].

4 Conclusions

Our experiments with the meeting acceptance task demonstrate that transfer learning often helps, but can also hurt performance if the sources of data are too dissimilar. The hierarchical naive Bayes algorithm was designed to avoid negative transfer, and indeed it does so quite well compared to the flat algorithm. Compared to the standard B-only approach, however, there is still room for improvement. As part of ongoing work we are exploring the use of clustering techniques, e.g., [8], to represent more explicitly that some sources of data may be better candidates for transfer than others.

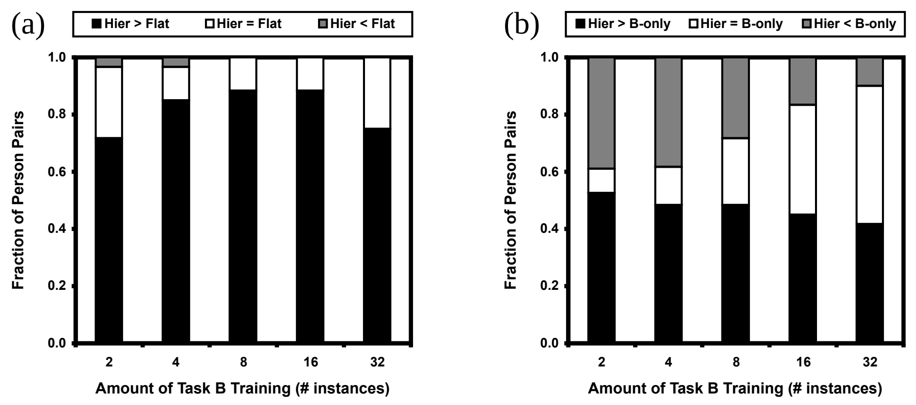

Figure 2: Effects of *B* training set size on performance of the hierarchical naive Bayes algorithm versus (a) flat naive Bayes and (b) training with no auxiliary data. Shown are the fraction of tested *A*-*B* pairs with a statistically significant transfer effect ($p<0.05$). Black and gray respectively denote positive and negative transfer, and white indicates no statistically significant difference. Performance scores were quantified using the log odds of making the correct prediction.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

[1] J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

[2] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–70, 1997.

[3] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2–3):103–130, 1997.

[4] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis, Second Edition*. Chapman and Hall/CRC, Boca Raton, FL, 2004.

[5] Z. Marx, M. T. Rosenstein, L. P. Kaelbling, and T. G. Dietterich. Transfer learning with an ensemble of background tasks. Submitted to this workshop.

[6] R. Neal. Slice sampling. *Annals of Statistics*, 31(3):705–767, 2003.

[7] C. Sutton and A. McCallum. Composition of conditional random fields for transfer learning. In *Proceedings of the Human Language Technologies / Emprical Methods in Natural Language Processing Conference (HLT/EMNLP)*, 2005.

[8] S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: the TC algorithm. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 489–497. Morgan Kaufmann, 1996.

[9] P. Wu and T. G. Dietterich. Improving SVM accuracy by training on auxiliary data sources. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 871–878. Morgan Kaufmann, 2004.