Two Algorithms for Transfer Learning

Zvika Marx[1], Michael T. Rosenstein[1], Thomas G. Dietterich[2], and Leslie Pack Kaelbling[1]

[1] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA {zvim,mtr,lpk}@csail.mit.edu
[2] School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR tgd@cs.orst.edu

Summary. Transfer learning aims at improving the performance on a target task given some degree of learning on one or more source tasks. This chapter introduces two transfer learning algorithms that can be employed when the source and target domains share the same feature space and class labels. The first algorithm is a hierarchical Bayesian extension of naive Bayes; the second is a version of logistic regression in which the prior distribution over the weight values is learned from an ensemble of source tasks. The methods are tested on a real-world task of predicting whether a person will accept or decline a meeting invitation. The results demonstrate consistent successful transfer of learning when there is an ensemble of source tasks.

1 Introduction

Transferring knowledge from a familiar domain or task (call it task A) to an unfamiliar or newly encountered one (task B) is a fundamental and fascinating aspect of human learning. Transfer of learning takes place in many contexts: someone starting a new job is likely to employ knowledge and skills aquired in previous jobs, athletes practice a variety of exercises to improve performance during competition, and so on. Although this motivating notion is intuitive, there exists no straightforward and general approach to transfer learning.

In broad terms, the challenge for a transfer learning system is to learn what knowledge should be transferred and how. For simplicity, let us assume that the tasks at hand can be represented as classification problems within the same feature space. Even under this assumption, the decision boundaries of A and B will generally not lie in exactly the same places. Hence treating them as identical and pooling their training data together will not work well and, in fact, may hinder performance if they are too dissimilar. The source A-task information must be used, therefore, in a more sophisticated manner to bias learning and improve performance on the target B task.

Previous work has demonstrated that learning for some target B task can be effectively influenced by inductive bias learned from one or more source A

tasks e.g., [1, 3, 13, 15]. Even for the restricted class of problems addressed by supervised learning, transfer can be realized in many different ways. For instance, Caruana [3] trained a neural network on several tasks simultaneously as a way to induce efficient internal representations for the target task. Wu and Dietterich [15] transferred source training examples either as support vectors or as constraints (or both) and demonstrated improved image classification by SVMs. Sutton and McCallum [12] demonstrated effective transfer by "cascading" a class of graphical models, with the predictions from one classifier serving as features for the next one in the cascade.

The rest of this chapter is organized as follows: Section 2 describes "transfer-aware" versions of the naive Bayes and logistic regression algorithms. Section 3 describes our application problem, which involves predicting whether a person will accept or decline a request for a meeting. We describe two experiments that evaluate our transfer algorithms on variants of this task. Finally, the chapter concludes with a discussion of the results and lessons for future research.

2 Two Transfer Learning Algorithms

We now describe our two transfer learning algorithms: hierarchical naive Bayes and prior logistic regression.

2.1 Hierarchical Naive Bayes

The standard naive Bayes algorithm—which we call here *flat naive Bayes*—has proven to be effective for learning classifiers in *non-transfer* settings [6]. 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 provide a methodology for combining data from multiple heterogeneous sources [7].

To simplify our presentation, we assume that just two tasks, A and B, provide sources of data. 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 capture the fact that two parameters are very similar by decreasing the variance of the hyperprior. Alternatively, it can represent the fact 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 [11].

This method is easily extended to handle multiple source tasks simply by asserting that corresponding parameter values for each naive Bayes classifier are all drawn from a common hyperprior distribution (a detailed description are provided in a technical report).

2.2 Logistic Regression with Learned Priors

Our second transfer learning algorithm requires an ensemble of source tasks. It works by fitting a separate logistic regression model to each source task and then analyzing the parameter values of the learned models to identify their joint probability distribution. This joint distribution is then applied as a prior distribution in the target task.

Logistic regression is one of the best known and most-effective methods for classification [10]. The logistic regression model has the following form:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp\left\{-(w_0 + \sum_{j=1}^{n} w_j x_j)\right\}}\,, \tag{1}$$

where y is the class label, $\mathbf{x}$ is a vector of n features, the w_j are real-valued weights, and $P(y = -1|\mathbf{x}) = 1 - P(y = 1|\mathbf{x})$. A standard way of fitting this model is to assume an independent Gaussian prior on the weight values. That is, each weight is drawn from a Gaussian distribution: $w_j \sim \mathcal{N}(\mu_j, \sigma_j)$. The model is fit by maximizing the following objective function:

$$\sum_{i=1}^{N} \log P(y_i|\mathbf{x}_i) - \sum_{j=0}^{n} \frac{1}{2}\left(\frac{w_j - \mu_j}{\sigma_j}\right)^2\,, \tag{2}$$

where $i = 1, \ldots, N$ indexes the training examples and $j = 0, \ldots, n$ indexes the features.

Typically, the values $\mu_j = 0$ and $\sigma_j = \sigma$ are employed, with σ (a constant, positive value) set by holdout or cross-validation methods [5]. The variance for the intercept weight, w_0, is typically set to be relatively large to avoid excessive penalties for deviations from μ_0. The model can be fit via iteratively reweighted least squares [9], boosting [8], or improved iterative scaling [2].

Following Chelba and Acero [5], we adjust this scheme for transfer learning as follows. Let K be the number of source A tasks and n be the number of features in the (common) feature space. First, we fit K independent logistic regression models (with prior mean 0 and $\sigma = 1$) and obtain fitted weights $\left\{w_j^k\right\}$, $k = 1, \ldots, K$ (the superscript k indicates the training task k). From

these, we then estimate the prior mean μ_j and standard deviation σ_j for parameter $j = 0, \ldots, n$ as follows:

$$\mu_j = \frac{1}{K} \sum_{k=1}^{K} w_j^k \quad \text{for } j = 1, \ldots, n$$

$$\sigma_j = \sqrt{\frac{1}{K-1} \sum_{k=1}^{K} (w_j^k - \mu_j^k)^2} \quad \text{for } j = 1, \ldots, n\,. \tag{3}$$

Finally, we fit a logistic regression model to the target B data using μ_j and σ_j to specify the prior distribution over the weights. The effect of this is that w_j values that are similar across all A tasks get a small standard deviation σ_j and so are tightly constrained in task B to have a value close to μ_j. Conversely, weights w_j that are highly variable across the A tasks will have large values for σ_j and will not be very constrained at all. This is slightly different from Chelba and Acero [5]: they fit μ_j to the data from a single A task and then set the σ_j values to a constant σ tuned with holdout data on the target task B. Such a strategy, because it relies on holdout methods, requires a substantial amount of B data. In our application, and in most transfer learning settings, there is not enough task B data to employ holdout methods.

3 Experimental Tests of These Algorithms

We now report on experimental tests of these two algorithms. We first describe the application problem that served as the test domain, and then we describe two experiments and their results.

3.1 The Meeting Acceptance Domain

We tested our transfer-aware versions of naive Bayes and logistic regression on data from a meeting acceptance task. In 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 participant provided two months of calendar data (after removing sensitive events) and also populated a relational database that described their projects, the people working on those projects, and their relationship to each person. From this information, we wrote a program that generated synthetic meeting requests from these people and placed them in the final two week period of the two-month calendar. Each participant then labeled each request (independently) according to whether they would accept or reject the request.

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. The features are described in Table 1.

Table 1. The features used in our experiments

Name	Values
meeting start time	morning*, afternoon*, early, mid-day, late (* denotes specific time was not indicated)
meeting duration	<16 min., 16–30 min., 31–60 min., >60 min.
urgency indicated	unimportant, possibly important, important, very important, critical
topic importance	values as above
inviter job	military officer, military planner, student, faculty, administrator, funding agent
inviter-recipient rel.	subordinate, peer, supervisor, family member, other
recent meeting	same day, 2–3 days ago, same week, a week ago or more
next meeting	same day, in 2–3 days, same week, in a week or more
day load	six values incorporating mixture of schedule and background-activity load
relevant half-day load	values as above
available free slot	<16 min., 16–30 min., 31–60 min., 61–90 min., 90–135 min., >135 min.
conflicting activity	exist, not exist
confl. act. importance	values as for **urgency indicated** and **topic importance** above
confl. act. frequency	none, one-time, occasional, regular
confl. act. location	six values combining locations of suggested meeting and conflicting activity, if exists, where location is one of "local" and "away"

3.2 Experiment 1: Hierarchical Naive Bayes on Task Pairs

In the first experiment, we considered transferring from one source domain (i.e., one person) to one target domain (i.e., another person). We evaluated all pairs of people. When a person serves as the target B task, we would first take 100 of his or her examples and set them aside as the test set. Then, from the remaining examples, we chose (at random, stratified by class), nested sets of 32, 16, 8, 4, and 2 training examples. For example, if a person had 228 total examples available, 100 were employed as the test set, and the remaining 128 examples provided four disjoint training sets of size 32; eight disjoint training sets of size 16; sixteen disjoint training sets of size 8; thirty-two disjoint training sets of size 4; and sixty-four disjoint training sets of size 2. These training

sets were employed to measure a learning curve. When a person serves as the source A task, all of his or her data is employed for training.

We evaluated three learning configurations: (i) our hierarchical naive Bayes method, (ii) "flat" naive Bayes in which the task A and task B training data were merged into a single training set, and (iii) B-only in which only the B training data was given to the naive Bayes algorithm.

Statistical significance was evaluated by performing an analysis of deviance in the statistical package R based on a logistic regression model in which there is an effect for each algorithm and a random effect for each training set. The statistical test assesses whether the effect due to the choice of learning algorithm is significant while controlling for random variation due to the different training sets (more details are provided in a technical report).

Figure 1 summarizes the results of these statistical tests. The bar chart in Figure 1a shows that for most cases, the hierarchical method is either statistically significantly superior to the flat method or else indistinguishable from it. There are only a few cases (at sample sizes 2 and 4) where the flat method is statistically significantly superior to the hierarchical naive Bayes method. The bar chart in Figure 1b compares hierarchical naive Bayes to B-only naive Bayes. Here the results are mixed. At a sample size of 2, the hierarchical method wins about half the time and loses about 40% of the time, so it has only a slight advantage over B-only. However, at a sample size of 32, the hierarchical method wins about 40% of the time, B-only wins about 10% of the time, and 50% of the time the methods are tied. Hence we see that there is often positive transfer, but sometimes there is also negative transfer. In data not shown, we find that the error rate of the hierarchical and B-only methods is about the same when averaged across all pairs of individuals.

3.3 Experiment 2: Logistic Regression on Task Ensembles

The second experiment tested transfer learning via logistic regression. For this experiment, we employed a leave-one-person-out cross-validation design. In each of the 21 iterations, a single person was chosen as the target (B) data source, 100 of his or her examples were set aside as the test set, and from the remaining examples 2, 4, 8, 16, or 32 examples were used for training following the same scheme as in experiment 1. The examples from each of the 20 remaining individuals served as 20 source training sets.

To apply logistic regression to the features described in Table 1, it is necessary to convert the features into boolean indicator functions, because, unlike naive Bayes, features with more than 2 values cannot be modeled directly in logistic regression. For features with a discrete set of values (e.g., urgency, importance, inviter job, etc.), we created a separate indicator function for each feature-value combination. For features that represent intervals (e.g., meeting duration, available free slot, etc.), we employed a so-called "thermometer representation" with one indicator function for each threshold value (e.g., one indicator for meeting-duration >15 min., meeting-duration >30 min., and

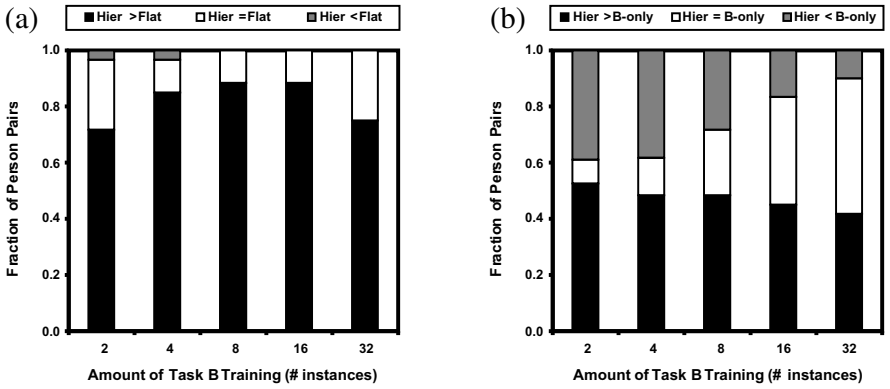

Fig. 1. Effects of B training set size on performance of the hierarchical naive Bayes algorithm versus (a) flat naive Bayes and (b) training with no source data. Shown are the fraction of tested A-B pairs with a statistically significant transfer effect ($p < 0.05$). Black and gray respectively denote positive and negative transfer, and white indicates no statistically significant difference. Performance scores were quantified using the log odds of making the correct prediction.

so on). After this conversion, the 15 original features were represented by 64 boolean indicator functions. The logistic regression algorithm learned a weight for each of these. We did not employ an intercept term.

Figure 2 plots the results compared to two baselines: a logistic regression classifier trained only on the B data ("B only"), with a fixed prior vaiance set to 1, and a logistic regression model trained on the union of all available A training instances and the B training examples ("flat"). In addition, the figure plots the mean, over the data for all individuals, of the proportion of the more prevalent class. The figure shows that logistic regression with a learned prior is substantially more accurate at all sample sizes than the B-only classifier. The flat classifier performs very poorly, because the non-B training examples overwhelm the small number of task B examples.

We conducted further experiments to measure transfer within and between the military and research domains. Recall that 8 of our participants were military personnel performing a training exercise, while the remaining 13 participants were academic or industrial researchers. Figure 3 plots the same information as Figure 2, but with the source and target tasks restricted to the military or research domains. For example, Figure 3a reports a 13-fold cross-validation in which there are 12 source researchers and one target researcher; Figure 3c reports a 13-fold cross-validation in which there are 8 source military personnel (in all cases) and one target researcher; and so on. The main conclusion once again is that logistic regression with learned priors outperforms the other three methods in all cases. When the target domain is a military person, the learning curves do not rise as steeply or as high as when

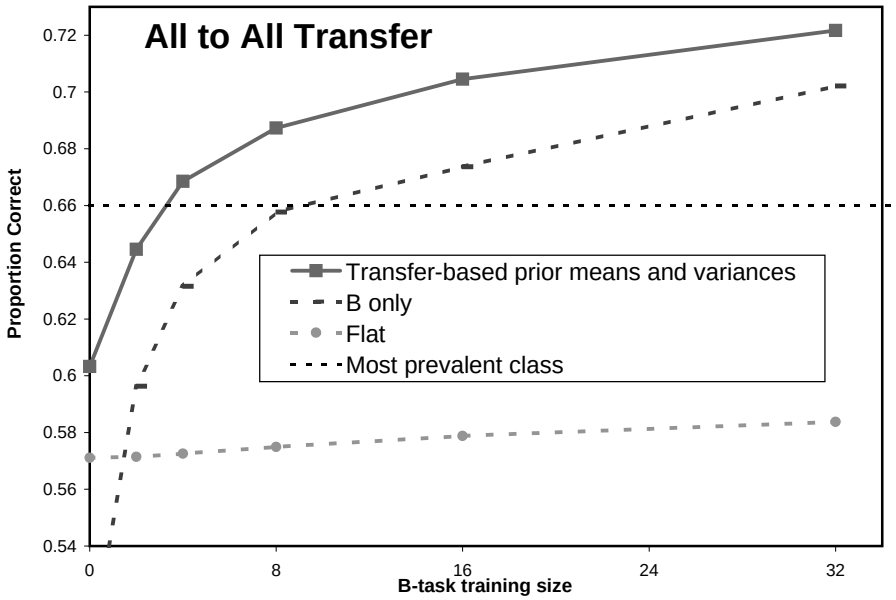

Fig. 2. Mean proportion of correct predictions averaged over all 21 B tasks. The logistic regression method improves over the B-only (transfer-unaware) and flat baselines. Each B task consists of the meeting acceptance data for one person. The ensemble of task As consists of the remaining 20 individuals. The differences relative to the B-only baseline are all statistically significant (one-tailed paired t test, $p < .05$; the significant differences are marked by solid squares).

the target domain is a researcher. This may signal that the military meeting acceptance task was harder to predict.

One interesting observation is that the flat method performs better *across* domains than *within* domains. This contradicts the intuition that instances collected from sources similar to the target task should enhance learning under the flat method compared to sources that are different from the target. One possible explanation is based on recalling that logistic regression learns a separate weight for each feature-value combination. Suppose there are feature value combinations that are only observed in research or only in the military. In such cases, when the source domains are all research and the target domain is military (or vice versa), there will be feature values that are always zero in the source domains but non-zero in the target domain. In such cases, only the training examples in the target domain contribute to learning the weight for those feature-value combinations, so the flat and B-only methods can both learn well.

But now consider what happens when the source and target are all from the military (or all from research). Further suppose that there is considerable diversity within the military (or within research). Then the same feature-value

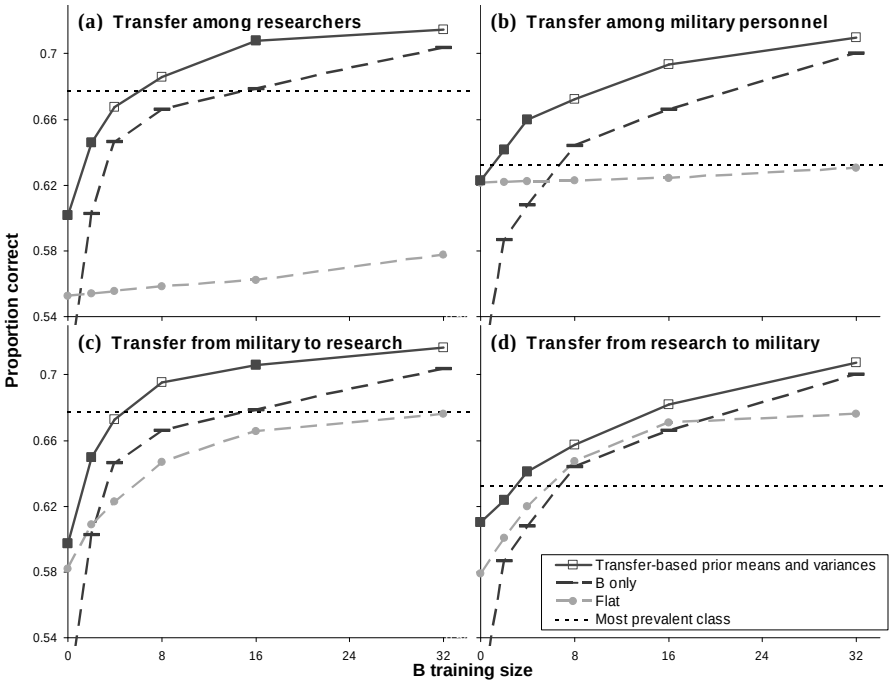

Fig. 3. Transfer within and across domains: (a) transfer among researchers, (b) transfer among military personnel, (c) transfer from military to research, (d) transfer from research to military. In all cases, logistic regression with learned priors gives better results than the other methods, although there are fewer statistically-significant differences. Statistically significant differences between transfer-aware logistic regression and the B-only baseline are marked by solid squares

combinations are observed in both the source and target tasks, so the flat and B-only methods will have difficulty learning the correct weights. In this case, the learned prior will have a large variance, so the learned prior will not tightly constrain the weight learned in task B. Consequently, the learned prior will work much better than either flat or B-only.

An initial inspection of the data for the academic researchers (where the effect is strongest) confirms this explanation. The academic researchers are extremely diverse in their meeting acceptance behavior. Each individual researcher makes decisions based on different features and, in several cases, researcher react differently to the same feature (for instance, accepting versus rejecting early morning meetings).

3.4 Discussion

The experiments show that the logistic regression method with learned priors is a good transfer learning algorithm when applied to an ensemble of source

tasks. It out-performs the simple "flat" transfer learning algorithm, and it also out-performs the no-transfer (B-only) method.

The results for the hierarchical naive Bayes method were less compelling. Hierarchical naive Bayes clearly out-performs the simple "flat" transfer learning method. But its performance against the no-transfer (B-only) method was less clear cut. It was better than B-only a majority of the time, but at small sample sizes there are still many cases where B-only is better.

As our two experiments employ different features and different experiment designs, we have performed some exploratory experiments for comparison. We have checked how well logistic regression, with individual-A-based means and fixed prior variance, works in a setting similar to Experiment 1. The results confirm that on average, it is difficult to see an improvement over B-only when transferring from only one source domain.

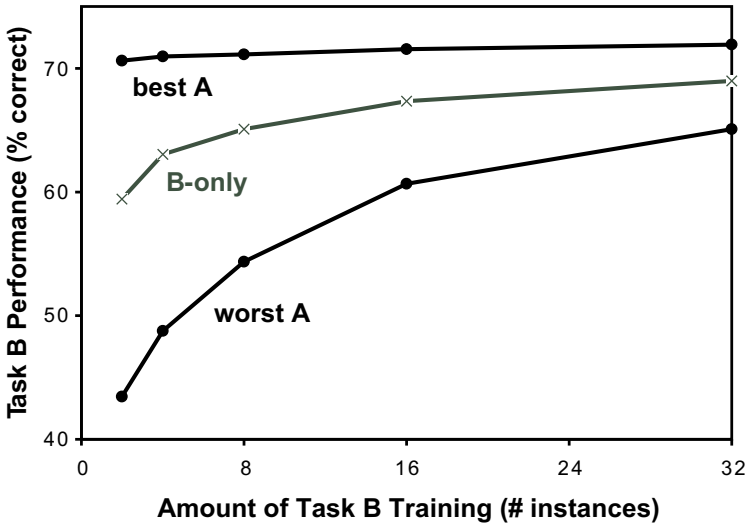

Fig. 4. Effects of B training set size on performance of the hierarchical naive Bayes algorithm for three cases: no transfer ("B-only") and transfer from the best and worst individual, i.e., "best A" and "worst A", respectively. At each B training size, differences between the transfer conditions and the corresponding B-only condition were all statistically significant ($p < 0.05$).

We have also explored whether training hierarchical naive Bayes on an ensemble of source domains would improve its performance compared to the B-only case. However, those initial experiments have not shown any particular benefit. Indeed, the main effect is to weaken the transfer effect by increasing the variance of the hyperprior, because the source domains are so diverse. This suggests that we need a two-level hyperprior that can "cluster" the source domains into groups based on similarity and then determine which groups are

most similar to the target domain. Figure 4 shows the results of a simple experiment to see how well hierarchical naive Bayes works when good (or bad) source domains are chosen. For each target domain, we determined which source domain (i.e., which individual) was most similar to the target by measuring the accuracy on the target domain of a classifier trained only on the source domain. We also determined which individual was the worst source domain for each target. The figure plots the average percentage of correct classifications (over the 21 target domains) for three cases: (a) training on the best source domain, (b) training only in the target domain (B-only), and (c) training on the worst source domain. This shows that if a learning algorithm could choose the correct source domain, hierarchical naive Bayes would give excellent results. This, however, depends on a significant yet unsolved part of the transfer problem: estimating the relevance of a source domain using the small amount of B data available in a typical transfer learning situation.

4 Concluding Remarks

Most research in transfer learning has been performed in situations where the source domains were constructed for the purpose of transfer, and therefore they were guaranteed to provide good transfer learning with suitable learning algorithms. This chapter has addressed the problem of transfer learning "in the wild", where it is possible that the source domains are significantly different from the target domain. The results in Figure 4 show that in our experiment, such significant differences do arise.

The mediocre performance of our hierarchical naive Bayes algorithm suggests that it is vulnerable to such significant differences. The logistic regression approach appears to be less vulnerable, although the reasons are unclear. It may be a result of the representational difference between logistic regression coefficients and the probability parameters of the naive Bayes model, or it may be a consequence of the discriminative nature of logistic regression models.

In any case, the use of an ensemble of several source domains—possibly from a different domain pool than the target's—was critical for the successful performance of the logistic regression method.

Future research must deepen our understanding of how transfer learning algorithms can learn about the relationships between different learning tasks and then use this knowledge to perform successful transfer. Future work must also consider situations in which the source and target tasks employ different ontologies and representations, so that transfer also requires understanding how these ontologies and representations are related.

Acknowledgements

The authors gratefully acknowledge the support of the Defense Advanced Research Projects Agency under contract NBCHD030010. Any opinions, findings, and conclusions or recommendations expressed in this material are those

of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency or the Department of Interior-National Business Center.

References

1. Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.
2. Adam Berger. The improved iterative scaling algorithm: A gentle introduction. Technical report, CMU, 1997
3. Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–70, 1997.
4. Stanley Chen and Ronald Rosenfeld. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50, 2000.
5. Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. In Dekang Lin and Dekai Wu, editors, *Proceedings of the 2004 Conference on Empirica Methods in Natural Language Processing*, pages 285–292. Association for Computational Linguistics (ACL), 2004
6. Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2–3):103–130, 1997.
7. Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*, Second Edition. Chapman and Hall/CRC, 2004.
8. Jerome H. Friedman, Trevor Hastie, and Rob Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.
9. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer Verlag, 2001.
10. Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, 2000.
11. Radford Neal. Slice sampling. *Annals of Statistics*, 31(3):705–767, 2003.
12. Charles Sutton and Andrew 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)*, pages 748–754. Association for Computational Linguistics (ACL), 2005.
13. Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: the TC algorithm. In Lorenza Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 489–497. Morgan Kaufmann, 1996.
14. Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances In Neural Information Processing Systems*, volume 8, pages 640–646. The MIT Press, 1996.
15. Pengcheng Wu and Thomas G. Dietterich. Improving SVM accuracy by training on auxiliary data sources. In Carla E. Brodley, editor, *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 871–878. Morgan Kaufmann, 2004.