Knowledge Transfer via Multiple Model Local Structure Mapping[*]

Jing Gao[†], Wei Fan[‡], Jing Jiang[†], and Jiawei Han[†]
[†]Dept. of Computer Science, University of Illinois at Urbana-Champaign, IL USA
[‡]IBM T. J. Watson Research Center, Hawthorn, NY USA
jinggao3@uiuc.edu, weifan@ibm.com, jiang4@uiuc.edu, hanj@cs.uiuc.edu

ABSTRACT

The effectiveness of knowledge transfer using classification algorithms depends on the difference between the distribution that generates the training examples and the one from which test examples are to be drawn. The task can be especially difficult when the training examples are from one or several domains different from the test domain. In this paper, we propose a locally weighted ensemble framework to combine multiple models for transfer learning, where the weights are dynamically assigned according to a model's predictive power on each test example. It can integrate the advantages of various learning algorithms and the labeled information from multiple training domains into one unified classification model, which can then be applied on a different domain. Importantly, different from many previously proposed methods, none of the base learning method is required to be specifically designed for transfer learning. We show the optimality of a locally weighted ensemble framework as a general approach to combine multiple models for domain transfer. We then propose an implementation of the local weight assignments by mapping the structures of a model onto the structures of the test domain, and then weighting each model locally according to its consistency with the neighborhood structure around the test example. Experimental results on text classification, spam filtering and intrusion detection data sets demonstrate significant improvements in classification accuracy gained by the framework. On a transfer learning task of newsgroup message categorization, the proposed locally weighted ensemble framework achieves 97% accuracy when the best single model predicts correctly only on 73% of the test examples. In summary, the improvement in accuracy is over 10% and up to 30% across different problems.

[*]The work was supported in part by the U.S. National Science Foundation grants IIS-05-13678/06-42771 and BDI-05-15813. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms

1. INTRODUCTION

We are interested in transfer learning scenarios where we learn from one or several training domains and make predictions in a different but related test domain. Such knowledge transfer is possible when the training domain(s) and the test domain have the same set of categories or class labels. We further assume that we are only exposed to some labeled examples from the training domains but do not have any labeled example from the test domain. The study of transfer learning is motivated by the fact that people often exploit knowledge gained from related domains where labeled data are abundant to classify examples in a new domain. Unfortunately, traditional supervised learning techniques usually fail to transfer knowledge in this scenario because it requires the training and the test data to be i.i.d. samples from the same distribution.

There are a few important observations about this problem. We notice that there are usually several classification models available from the training domains. For example, the classifiers can be trained from several relevant domains or built using different learning algorithms on the same domain. Different models usually contain different knowledge and thus have different advantages, due to the inductive bias of the specific learning technique as well as the distributional differences among the training domains. Therefore, different models may be effective at different regions or structures in the new and different test domain, and no single model can perform well in all regions. We refer to these different models as base models. Ideally, we may wish to combine the knowledge from these base models rather than using any single model alone to more effectively transfer the useful knowledge to the new domain. For this task, one would naturally consider model averaging that additively combines the predictions of multiple models. However, the existing model averaging methods in traditional supervised learning usually assign global weights to models, which are either uniform (e.g., in Bagging), or proportional to the training accuracy (e.g., in Boosting), or fixed by favoring certain model (e.g., in single-model classification). Such a global weighting scheme may not perform well in transfer learning because different test examples may favor predictions from different base models. For example, when the base models carry conflicting concepts at a test example, it is essential to select the model that better represents the true target distribution underlying the example. In fact, based on principles of risk minimization, we can derive that there exists a solution to assign per model and per example weights to combine multiple base models to maximize their combined accuracy on the new domain, and the combined accuracy is higher than any single model acting alone. However, it is impossible to dynamically assign the optimal model weights for each example precisely because $P(y|\mathbf{x})$, the true conditional probability of class label y given a test example $\mathbf{x}$, is not known a priori. Past practice of cross-validation based weight assignment is inapplicable since the weights would be assigned based on labels in the given training domain(s) whose $P(y|\mathbf{x})$ could be different from that of the test domain. Therefore the focus of this paper is to find an approximation to this optimal local weight assignment for each test example.

We propose a graph-based approach to approximate the optimal model weights where the local weight for a base model is computed by first mapping and then measuring the similarity between the model and the test domain's local structure around the test example. This similarity is measured by comparing neighborhood graphs, and quantified in the weight assignment equation. Intuitively, it favors classifiers whose mapped local structure is similar to the local structure around the test example. For a particular example, if none of the mapped local structures is similar to the original local structure in the target domain, the predicted label will be obtained by voting among its neighbors inside the same local structure of the test set. This strategy ensures that the maximum amount of predictive powers of the labeled information are extracted and transferred to the test domain to make the predictions consistent with its underlying manifold structure.

Our main contributions to the task of transfer learning include the following: (1) We propose a locally weighted ensemble framework to address the transfer learning problem, and demonstrate its superiority over single models in terms of risk minimization when the weights are set optimally. (2) None of the base models is required to be specifically designed for transfer learning, thus providing great flexibility and freedom on what models to use. (3) We propose to approximate the model weights based on the local manifold structures in the test domain, and provide neighborhood graph-based estimation. (4) We provide a prediction adjustment step to propagate labels from nearby examples when all base models are inconsistent with certain test examples.

We evaluated the proposed framework on three real tasks: spam filtering, text categorization, and network intrusion detection. In each task, the test examples come from a different domain than the training set. Our experiment results show that the locally weighted ensemble framework significantly improved the performance over a number of baseline methods on all three data sets, which shows the effectiveness of the proposed framework for transfer learning.

2. LOCALLY WEIGHTED ENSEMBLE

Let us first look at a toy learning problem with two training sets and a test set shown in Figure 1. The two training sets have partially conflicting concepts and their decision boundaries are the straight lines. For the test set, however,

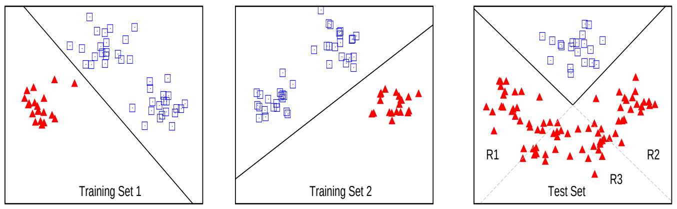

Figure 1: A Motivating Example

the optimal decision boundary is the V-shape solid line. As can be seen, the regions R_1 and R_2 are "uncertain," because the two training sets are conflicting there. If we either simply collapse the two data sets and try to train a classifier on the merged examples, or combine the two linear classifiers M_1 and M_2 trained from the training set 1 and set 2 respectively, then those negative examples in R_1 and R_2 will be hard to predict. Those semi-supervised learning algorithms do not work either because they only propagate the labels of the training examples to the unlabeled examples. In this case, there are conflicting labels in R_1 and R_2, causing ambiguous and incorrect information to be propagated. But it is obvious that, if M_1 is used for predicting test examples in R_1 and M_2 used for examples in R_2, then we can label all test examples correctly. Therefore, ideally, one wish to have a "locally weighted" ensemble framework that combines the two models, and weighs M_1 higher at R_1 and M_2 higher at R_2. We also observe that this data set has a property that neighbors along the same "clustering-manifold structure" share the same class labels, which is a commonly-held assumption for reasonable problems. Below, we first introduce a locally weighted ensemble framework with weights dynamically adjusted according to the model behaviors at each test example. We then present an effective way of approximating the model weights via local structure mapping around each example. The success of the proposed method on this toy data set is demonstrated in Section 4.2.

2.1 Optimal Domain Transfer Weights

Let $\mathbf{x}$ be the feature vector and y be the class label where $\mathbf{x}$ and y are drawn from feature space $\mathbf{X}$ and label space Y respectively. For a set of k models $M_1, \ldots, M_k$, the general Bayesian model averaging approach computes the posterior distribution of y as $P(y|\mathbf{x}) = \sum_{i=1}^{k} P(y|\mathbf{x}, D, M_i)P(M_i|D)$, where $P(y|\mathbf{x}, D, M_i) = P(y|\mathbf{x}, M_i)$ is the prediction made by each model and $P(M_i|D)$ is the posterior of model M_i after observing the training set D. However, in transfer learning, since training and test domains are different, we may wish to incorporate information about the test domain and update the model prior for $P(M_i|T)$, where T is the test set. So $P(M_i|D)$ should be replaced by $P(M_i|T)$ in the weighted combination of model predictions. By this replacement, we take the difference between training and test domains into consideration during learning. If the true distribution $P(y|\mathbf{x})$ is known, then for predictions on $\mathbf{x}$, the other examples in the test set are irrelevant to the model performance at $\mathbf{x}$. In other words, the model weight $P(M_i|T)$ is actually $P(M_i|\mathbf{x})$ at $\mathbf{x}$ when $P(y|\mathbf{x})$ is available. Different from traditional ensemble approaches, this locally weighted model averaging method weights individual models according to their local behaviors at each test example. The final prediction for $\mathbf{x}$ is:

$$P(y|\mathbf{x}) = \sum_{i=1}^{k} w_{M_i,\mathbf{x}} P(y|\mathbf{x}, M_i), \tag{1}$$

where $w_{M_i,\mathbf{x}} = P(M_i|\mathbf{x})$ is the true model weight that is locally adjusted for $\mathbf{x}$ representing the model's effectiveness on the test domain.

The benefits of this locally weighted model averaging approach can be shown as follows. To simplify the problem, we map the label space Y to $\{1, \ldots, c\}$ where c is the number of classes. We then use a $c \times 1$ vector $\mathbf{f}$ to denote the true conditional probability in the test domain where the i-th element is $f_i = P(y = i|\mathbf{x})$. Supervised learning can output a $c \times 1$ vector $\mathbf{h}$ that is close to $\mathbf{f}$ for $\mathbf{x}$. Let $w_i = w_{M_i,\mathbf{x}}$ denote the weight for model M_i at test example $\mathbf{x}$, and let $\mathbf{w}$ denote the $k \times 1$ weight vector. $\mathbf{h}^i$ represents the predictions made by model M_i at $\mathbf{x}$ and is again a $c \times 1$ vector where the j-th element is $h^i_j = P(y = j|\mathbf{x}, M_i)$. $\mathbf{H}$ is used to represent a $c \times k$ matrix with all the model predictions made for $\mathbf{x}$ where the ij entry is model M_i's predicted $P(y = j|\mathbf{x}, M_i)$, i.e., h^i_j. Then the output of the model averaging framework for $\mathbf{x}$ is a vector $\mathbf{h}^e = \mathbf{H}\mathbf{w}$. Note that $\mathbf{w}$ satisfies the constraints that $w_i \in [0, 1]$ and $\sum_{i=1}^{k} w_i = 1$, and thus the output vector $\mathbf{h}^i$ from a single model M_i is a special case of $\mathbf{h}^e$ when $w_i = 1$ and other weights are zero. But we wish to find a weight vector $\mathbf{w}$ which minimizes the distance between $\mathbf{f}$ and $\mathbf{h}^e$. Under squared-error loss, the following objective function should be minimized to obtain the optimal $\mathbf{w}$:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} (\mathbf{f} - \mathbf{H}\mathbf{w})^T (\mathbf{f} - \mathbf{H}\mathbf{w}) + \lambda(\mathbf{w}^T \mathbf{I} - 1), \tag{2}$$

where $\mathbf{I}$ is a $k \times 1$ vector of 1 and λ is the regularization term. It is obvious that Eq. (2) represents a least-square linear regression problem and the optimal solution is

$$\mathbf{w}^* = (\mathbf{H}^T \mathbf{H})^{-1} (\mathbf{H}^T \mathbf{f} - \frac{1}{2}\lambda \mathbf{I}). \tag{3}$$

λ can be further calculated by substituting the above $\mathbf{w}^*$ to the constraint $(\mathbf{w}^*)^T \mathbf{I} = 1$. Usually w^*_i is a value between 0 and 1 so the weight vector of the optimal ensemble is different from that of the single model. Therefore, the error of the model averaging framework on each test example $\mathbf{x}$ will not be greater than that of any single model:

$$(\mathbf{f} - \mathbf{H}\mathbf{w}^*)^T (\mathbf{f} - \mathbf{H}\mathbf{w}^*) \le (\mathbf{f} - \mathbf{h}^i)^T (\mathbf{f} - \mathbf{h}^i) \quad \forall i \tag{4}$$

Thus, for each test example, there is a smaller chance to make a mistake if we combine the predictions from different models using the optimal weight vector. It is important to note that the optimal weight vectors are different for different test examples, so weights should be decided locally.

This locally weighted ensemble framework differs from traditional model averaging methods in the following ways: 1) In transfer learning problems, the traditional methods of assigning model weights based on training set or assigning fixed prior weights are undesirable. Instead, we do not assume that training and test domains follow same distributions but rather focus on the test set when deriving the best model weights to transfer knowledge across domains. 2) Existing work usually weights each model globally, but the proposed method assigns per example weights to each model to identify variations in model performance for different test examples. As discussed, there may not exist one model globally optimal for all the test examples. Usually, different test examples favor different models and therefore the per example weighting scheme is better than the global weighting scheme in terms of classification accuracy.

One challenge is that the optimal per example weight vectors cannot be computed exactly in reality, since the true target vector $\mathbf{f}$ for each test example $\mathbf{x}$ is not known a priori. Importantly however, from its solution in Eq. (3), a model will have a higher weight if its prediction on $\mathbf{x}$ is closer to the true $P(y|\mathbf{x})$. In the rest of this paper, we propose a graph-based approach to approximate the optimal per example weight $w_{M_i,\mathbf{x}}$ under the "clustering-manifold" assumption that $P(\mathbf{x})$ is related to $P(y|\mathbf{x})$. Other approximation heuristics can be developed under this locally weighted framework as long as the weights reasonably approximate the model performance for given test examples.

3. GRAPH-BASED WEIGHT ESTIMATION

As discussed in Section 2.1, the optimal weights can be approximated by assigning a higher weight to a model that produces a more accurate label prediction for $\mathbf{x}$. So the main task is to formulate similarity between the model predictions and the unknown true target function. To achieve this goal, we can model the underlying $P(\mathbf{x})$ from the unlabeled test set in order to infer $P(y|\mathbf{x})$. Specifically, we make a "clustering-manifold" assumption, as commonly held in semi-supervised learning, that $P(y|\mathbf{x})$ is not expected to change much when the marginal density $P(\mathbf{x})$ is high. In other words, the decision boundary should lie in areas where $P(\mathbf{x})$ is low. Under such an assumption, we can compare the difference in $P(y|\mathbf{x})$ between the training and the test data locally with only unlabeled test data. However, probability density estimates are hard to obtain precisely, especially when $\mathbf{x}$ is high-dimensional. Instead, we propose to cluster the test data and assume that the boundaries between the clusters represent the low density areas. As a result, if the local cluster boundaries agree with the classification boundary of M around $\mathbf{x}$, then we assume that $P(y|\mathbf{x}, M)$ is similar to the true $P(y|\mathbf{x})$ around $\mathbf{x}$, and thus the weight for model M ought to be high at $\mathbf{x}$. In the following, we formally give a procedure of computing the weight and illustrate the procedure with an example.

For a test example $\mathbf{x}$ and a base model M to be combined, we first construct two graphs: $G_T = (V, E_T)$ and $G_M = (V, E_M)$. In both graphs, the vertex set V contains all the test examples. For G_M, there is an edge connecting two test examples if and only if the examples are classified into the same class by M. On the other hand, to construct G_T, we cluster the test examples into c' clusters and again, connect two test examples with an edge if and only if the two examples are in the same cluster. Then we can approximate the model weight as the similarity between the local structures around $\mathbf{x}$ in G_T and G_M. Specifically, under the clustering assumption, it is probable that two examples are in the same class if they belong to the same cluster in G_T. So we could use the percentage of common neighbors of $\mathbf{x}$ found in G_M and G_T to approximate the model accuracy on $\mathbf{x}$ and set the weight. Suppose the sets of neighbors for $\mathbf{x}$ in G_M and G_T are V_M and V_T respectively. The model weight at $\mathbf{x}$ is proportional to the similarity of its local structures between G_M and G_T:

$$w_{M,\mathbf{x}} \propto s(G_M, G_T; \mathbf{x}) = \frac{\sum_{v_1 \in V_M} \sum_{v_2 \in V_T} \mathbf{1}\{v_1 = v_2\}}{|V_M| + |V_T|} \tag{5}$$

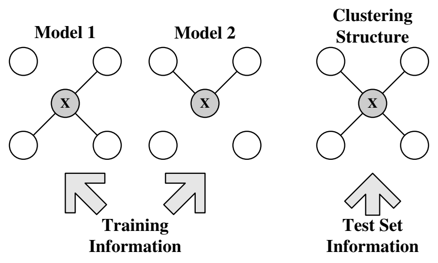

Figure 2: Local Neighborhood Graphs around x

According to its definition, $s(G_M, G_T; \mathbf{x})$ reflects the degree of consistency in labeling the test examples. If $\mathbf{x}$ has similar sets of neighbors in G_M and G_T, it is likely that the model M is consistent with the underlying structure around $\mathbf{x}$. As an example, Figure 2 shows the neighborhood graphs of a test example $\mathbf{x}$ constructed from two supervised models and the clustering algorithm on the test set. According to Eq. (5), the similarity between model 1 and the clustering structure is 0.75 at $\mathbf{x}$, but that between model 2 and the structure is 0.5. Therefore, for $\mathbf{x}$, model 1's weight will be set higher since it is more consistent with the local structure around $\mathbf{x}$. This is a simple and effective method to compute the similarity.

The weight approximation is based on the clustering assumption which requires that the manifold structure of the data is related to the conditional probability $P(y|\mathbf{x})$. Though this is a reasonable assumption for many problems, it may not always hold. Without knowing $P(y|\mathbf{x})$ a priori, it is impossible to verify the assumption. But this property is usually determined by the nature of the learning tasks. An example where the assumption does not hold is sentiment classification, where the clustering structure of a set of product reviews reveals the topics but may have nothing to do with whether the users like or dislike the product. Therefore, we propose to check the validity of the clustering assumption by evaluating the clustering quality on the training set using purity, entropy or F measure. If the task fails the test, we will ignore the weight approximation step, but simply combine the models using uniform weights. This strategy restricts the use of the graph-based weight estimation only to the cases where the clustering assumption is satisfied on both training and test sets. However, the strict checking criteria could guarantee the high accuracy of the proposed method. For the cases where the clustering assumption does not hold, other techniques need to be explored.

When the condition holds, we compute the per-example model weights based on Eq. (5) with a normalization term:

$$w_{M_i,\mathbf{x}} = \frac{s(G_{M_i}, G_T; \mathbf{x})}{\sum_{i=1}^{k} s(G_{M_i}, G_T; \mathbf{x})}, \tag{6}$$

where M_i is one of the k models. Then the final prediction of the weighted ensemble E for $\mathbf{x}$ is:

$$P(y|E, \mathbf{x}) = \sum_{i=1}^{k} w_{M_i,\mathbf{x}} P(y|M_i, \mathbf{x}), \tag{7}$$

where $P(y|M_i, \mathbf{x})$ is the prediction made by model M_i. Then the predicted label for $\mathbf{x}$ goes to y^* which minimizes the risk:

$$y^* = \arg\min_y \int_{y' \in Y} \lambda(y', y) P(y|E, \mathbf{x}) \mathrm{d}y' \tag{8}$$

where $\lambda(y', y)$ is the cost incurred when the true class label is y' but the prediction goes to y. With the most commonly used zero-one loss function, $y^* = \arg\max_y P(y|E, \mathbf{x})$.

3.1 Local Structure Based Adjustment

The weighting scheme shown in Eq. (7) works on the basis that at least some of the models do reasonably well on predicting the label for $\mathbf{x}$. However, if the concepts carried by all the models conflict with the actual concept at $\mathbf{x}$, the similarity measure $s(G_M, G_T; \mathbf{x})$ is expected to be low for each model M. But after the normalization in Eq. (6), the locally weighted ensemble framework would still make decisions based on these models for $\mathbf{x}$ and it is probable that the combined output is still in conflict with the true one. In such a scenario, it is reasonable to abandon the labeled information conveyed by the supervised models but rather rely on the local structure around $\mathbf{x}$ only.

Since the similarity measure $s(G_M, G_T; \mathbf{x})$ reflects the degree of consistency between model M's prediction and $\mathbf{x}$'s neighborhood structure, we can use the average $s(G_M, G_T; \mathbf{x})$ over all M to judge whether the labeled information is reliable or not. In fact, $s(G_M, G_T; \mathbf{x})$, representing the average percentage of common neighbors shared by supervised models and clustering results, is within $[0, 1]$. To be exact, when a test example shares the same neighbors in two graphs, their similarity is 1, whereas if no common neighbor is found, it is 0. So for example, if only two models are used and their $s(G_M, G_T; \mathbf{x})$ are both 0.01 at $\mathbf{x}$, then we should avoid normalizing the weights into 0.5 since both models should rather be discarded. Let $s_{avg}(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} s(G_{M_i}, G_T; v)$ be the average similarity between the base models' predictions on $\mathbf{x}$ and the clustering structure around $\mathbf{x}$. Then if $s_{avg}(\mathbf{x}) \geq \delta$, where δ is the threshold, we believe in the prediction obtained from Eq. (7); otherwise, we discard all the supervised classifiers and construct an "unsupervised" classifier based on the neighborhood of $\mathbf{x}$.

The "unsupervised" classifier U is not trained on any labeled training set. Its prediction on $\mathbf{x}$ is mainly determined by the neighbors of $\mathbf{x}$ with labels predicted by the combined classifier. Specifically, $P(y|U, \mathbf{x})$ can be decomposed as:

$$P(y|U, \mathbf{x}) = \sum_C P(y|U, \mathbf{x} \in C) P(\mathbf{x} \in C|\mathbf{x}). \tag{9}$$

Here, C is one of the clusters in the test set. We assume that the cluster membership is deterministic, then $P(\mathbf{x} \in C|\mathbf{x})$ is approximated as follows:

$$P(\mathbf{x} \in C|\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in C \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Hence, $P(y|U, \mathbf{x})$ is approximately the same as $P(y|U, \mathbf{x} \in C)$ when $\mathbf{x}$ belongs to cluster C. We can further approximate $P(y|U, \mathbf{x} \in C)$ as the average $P(y|E, \mathbf{x})$ for $\mathbf{x} \in C'$ where C' contains test examples which satisfy both $\mathbf{x} \in C$ and $s_{\text{avg}}(\mathbf{x}) \geq \delta$. In other words, only examples that have reliable predictions from the weighted ensemble will count in this procedure. Therefore,

$$P(y|U, \mathbf{x} \in C) \approx \frac{1}{|C'|} \sum_{\mathbf{x} \in C'} P(y|E, \mathbf{x}) \tag{11}$$

where $|C'|$ is the size of C'. The above strategy can be simplified if we set $P(y|E, \mathbf{x}) = 1$ when y is the label for $\mathbf{x}$ predicted by E. So $P(y|U, \mathbf{x} \in C)$ can be estimated by a

```
Algorithm: Locally Weighted Ensemble (LWE)
Input: (1)A training set $D$ or $k$ training sets $D_1, \ldots, D_k$
       (2)$k$ classification models $M_1, \ldots, M_k$ ($k > 1$)
       (3)A test set $T$ which comes from a different domain
       but the classification task is the same.
       (4)A threshold $\delta$ and cluster number $c'$.
Output: The set of predicted labels $Y$ for examples in $T$.
Algorithm:
  1. Perform clustering on the training set(s), IF the average
     purity of clustering is less than 0.5, set $w_{M_i,\mathbf{x}} = \frac{1}{k}$ for
     all $M_i$ and $\mathbf{x}$, and compute the posterior using Eq.(7)
     for each $\mathbf{x} \in T$. RETURN.
  2. Group test examples into $c'$ clusters and construct
     neighborhood graphs based on the clustering results and
     all the $k$ models. Set $T' = \Phi$.
  3. FOR each $\mathbf{x} \in T$,
       • FOR each model $M_i$, compute the model weight
         $w_{M_i,\mathbf{x}}$ according to Eq.(5).
       • IF $s_{\text{avg}}(\mathbf{x}) \geq \delta$, decide $\mathbf{x}$'s label based on the
         weighted ensemble's output $P(y|E,\mathbf{x})$ obtained us-
         ing Eq.(7), ELSE put $\mathbf{x}$ into $T'$.
  4. FOR each $\mathbf{x} \in T'$, predict $\mathbf{x}$'s label from the "unsuper-
     vised" classifier $U$, i.e., estimate $P(y|U,\mathbf{x})$ using Eq.(11)
     or Eq.(12). RETURN.
```

Figure 3: Locally Weighted Ensemble Framework

majority vote among examples in C':

$$P(y|U, \mathbf{x} \in C) \approx \frac{P(y, \mathbf{x} \in C'|E)}{P(\mathbf{x} \in C')} \approx \frac{c(y, C'|E)}{|C'|} \qquad (12)$$

where $c(y, C'|E)$ is the number of examples with label y predicted by ensemble E in C'. So the probability of $\mathbf{x}$ having label y is the percentage of examples in the cluster C' that have y as their class labels, where C' is the cluster that $\mathbf{x}$ belongs to and contains test examples with predicted labels. The final predicted label for $\mathbf{x}$ is determined by Eq. (8) with $P(y|E,\mathbf{x})$ replaced by $P(y|U,\mathbf{x})$. If zero-one loss function is applied, the class label for $\mathbf{x}$ whose cluster is C should be the majority label prediction among the test examples which satisfy both $\mathbf{x} \in C$ and $s_{\text{avg}}(\mathbf{x}) \geq \delta$.

3.2 Algorithm Description

The framework is summarized in Figure 3. We first verify whether the clustering structure is relevant to the classification task by performing clustering on the training set. If the purity of clustering on the training set is below 0.5, we simply combine models using uniform weights. Otherwise, if the clustering quality is satisfactory, in step 2, we construct the neighborhood graphs for both the supervised models and the clustering results. Then in step 3, the weight of each model at each test example is computed, which reflects the consistency of model predictions among the test example's neighborhood. We then separate the test examples by checking if its average model weight is greater than a confidence threshold. For those test examples on which cross domain models can make sufficiently accurate predictions, the final label predictions are decided by the locally weighted ensemble. But, for the test examples that the models are not expected to classify correctly, the labels are determined by majority voting among those neighbors with highly confident predictions within the same cluster structure.

Table 1: Data Sets Description

Task	Data Sets	Training	Test
Email Spam Filtering	User1(U00) User2(U01) User3(U02)	Public messages	Each user's emails
20 News-group	Comp vs Sci (C vs S) Rec vs Talk (R vs T) Rec vs Sci (R vs S) Sci vs Talk (S vs T) Comp vs Rec (C vs R) Comp vs Talk (C vs T)	Documents from a set of sub categories	Documents from a different set of sub categories
Reuters	Orgs vs People (O vs Pe) Orgs vs Place (O vs Pl) People vs Place (Pe vs Pl)	Documents from a set of sub categories	Documents from a different set of sub categories
Intrusion Detection	DOS	Probing & R2L Intrusions	DOS Intrusions
	Probing	DOS & R2L Intrusions	Probing Intrusions
	R2L	DOS & Probing Intrusions	R2L Intrusions

4. EXPERIMENTS

In this part, we demonstrate the effectiveness of the locally weighted ensemble framework. The algorithms are evaluated on various data sets covering many application domains. Results show that the proposed framework could combine the predictive powers obtained from multiple sources and gain great improvements in classification accuracy. The software, datasets and more details about the experiments are available at http://ews.uiuc.edu/∼jinggao3/kdd08transfer.htm.

4.1 Data Sets and Experiment Setup

We conduct experiments on one synthetic and four real data sets, where training and test distributions are different.

Synthetic Data.

The two training sets and the test set as shown in Figure 1 are generated from several Gaussian distributions with the same variance. In each training set, there are 40 positive and 20 negative examples and in the test set, the number of positive and negative examples are 20 and 40 respectively.

Email spam filtering.

The email spam data set, released by ECML/PKDD 2006 discovery challenge, contains a training set of publicly available messages and three sets of email messages from individual users as test sets. The 4000 labeled examples in the training set and the 2500 test examples for each of the three different users differ in the word distribution. The aim is to design a server-based spam filter learned from public sources and transfer it to individual users.

Document classification.

The 20 newsgroups data set contains approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups nearly evenly. The Reuters-21758 corpus contains Reuters news articles from 1987. From the two text collections, we generate nine cross-domain learning tasks. Both text collections have a two-level hierarchy so that each learning task involves a top category classification problem but the training and test data are drawn from different sub categories. For example, the goal is to distinguish documents from two top newsgroup categories: rec and talk. So a training set involves documents from "rec.autos," "rec.motorcycles," "talk.politics" and "talk.politics.misc," whereas the test set includes sub-categories "rec.sport.baseball," "rec.sport.hockey," "talk.politics.mideast" and "talk.religions.misc". The strategy is to split the sub-categories among the training and the test sets so that the distributions of the two sets are similar but not exactly the same. The tasks are generated in the same way as in [9] and more details can be found there.

Intrusion detection.

The KDD cup'99 data set consists of a series of TCP connection records for a local area network. Each example in the data set corresponds to a connection, which is labeled as either normal or an attack, with exactly one specific attack type. Some high level features are used to distinguish normal connections from attacks, including host, service and traffic features. In the experiments, we use the 34 continuous features. Attacks fall into four main categories: DOS(denial-of-service), R2L(unauthorized access from a remote machine), U2R(unauthorized access to local superuser privileges), Probing(surveillance and other probing). Since in reality, we usually encounter the problem of detecting the variants of known attacks, it is realistic to have one type of intrusions in the training set but another type in the test set. We create three data sets, each contains a set of randomly selected normal examples and a set of attacks from one category. Since the number of U2R attacks is small, we only use examples from DOS, R2L and Probing categories. Then three cross-domain learning tasks are generated by training from two types of attacks to detect another type of attack. The details of the four real tasks are presented in Table 1.

Baseline methods.

We compare the weighted ensemble framework with different learning algorithms. In particular, since most data sets are high-dimensional, the following commonly used algorithms are appropriate choices: 1) Winnow (**WNN**) from learning package SNoW [6], 2) Logistic Regression (**LR**) implemented in BBR package [16]; and 3) Support Vector Machines (**SVM**) implemented in LibSVM [8]. When we only have a single source domain in the training, three single classifiers are trained using the above learning algorithms and combined according to the proposed weighted ensemble framework. But note that the proposed method is a general framework so that any kind of models could be plugged in and transferred to the test domain. Since semi-supervised learning (transductive learning) is closely related to the problem, we compare the proposed method with Transductive Support Vector Machines (TSVM) implemented in SVM light [20]. Furthermore, in the proposed framework, the two main steps are, predicting labels using weighted classifiers if the classifiers are sufficiently accurate in terms of alignment with clustering structures; and propagating the labels of predicted test examples to the unpredicted ones through the clustering structure. To demonstrate the effectiveness of both steps, we include the following three methods in the comparison: 1) A simple model averaging framework (**SMA**) where all model predictions are combined using uniform weights; 2) The locally weighted ensemble framework without the adjustment step, which simply adopts the weighted prediction for each test example. We call it partial locally weighted ensemble method (**pLWE**); 3) The locally weighted ensemble framework (**LWE**) involving both classifier combination and local structure based adjustment. Note that **SMA** is one of the global ensemble methods where the model weights are set the same for all the test examples. Suppose there are k models, then each model will have a weight $\frac{1}{k}$ at every test example. We use the clustering package CLUTO [21], which is designed for high-dimensional data clustering, to cluster the test set. Again, other clustering algorithms could be used as long as the "clustering" assumption is satisfied.

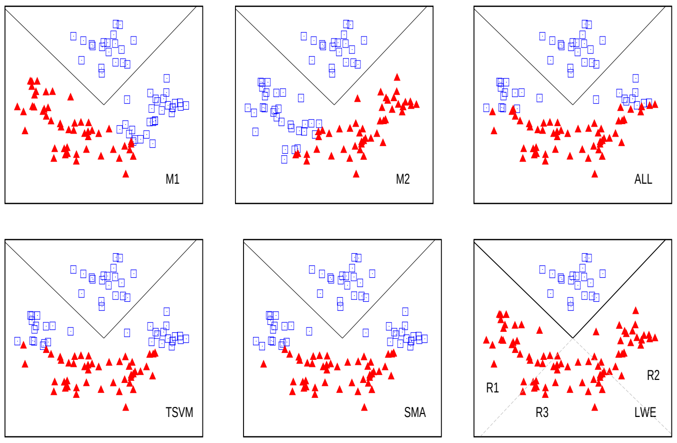

Figure 4: Performance on Synthetic Data

We compare with a set of different baseline methods on the synthetic and intrusion detection data sets. In each task, we have two source domains for training and the remaining one for the testing. The proposed weighted ensemble methods (pLWE and LWE) are built upon two single models trained from the two source domains using SVM. First, we compare pLWE and LWE with the simple averaging method (SMA) based on the two SVM models. Second, we can choose the training set as 1) one of the two source data sets, or 2) the union of the two source data sets. On the three possible training sets, we study the performance of supervised learning models (SVM) and semi-supervised models (TSVM) and compare them with the proposed methods.

Performance measures.

To compare the performance of the classification methods, we look at a set of standard evaluation metrics. First, we use classification accuracy, which is simply defined as the percentage of correct predictions among all test examples. Second, under squared loss function, the algorithms can be evaluated using Mean Squared Errors defined as follows: $L = \frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - \mathcal{P}(+|\mathbf{x}_i))^2$ where $f(\mathbf{x}_i)$ is the output of the classifier, which is the estimated posterior probability of $\mathbf{x}_i$ belonging to positive class, $P(+|\mathbf{x}_i)$ is the true posterior probability and $\{\mathbf{x}_i\}_{i=1}^{n}$ represents the test set. Another measure is used in evaluating the intrusion detection task: the area under ROC curve (AUC), the best of which is 1 corresponding to 100% detection and 0% false alarm. In the experiments, we focus on binary classification, but the framework can be easily applied on multi-class tasks.

4.2 Performance Evaluation

In this part, we report the experimental results regarding the effectiveness of the locally weighted ensemble. The results clearly demonstrate that on the transfer learning prob-

Table 2: Performance Comparison on a Series of Data Sets

Accuracy

Methods	Spam Filtering			20 Newsgroup						Reuters		
	U00	U01	U02	C vs S	R vs T	R vs S	S vs T	C vs R	C vs T	O vs Pe	O vs Pl	Pe vs Pl
WNN	0.7680	0.7888	0.8696	0.6554	0.5938	0.7942	0.7557	0.8926	0.9341	0.7058	0.6520	0.5685
LR	0.7060	0.7528	0.8500	0.7349	0.7217	0.7885	0.7904	0.8334	0.9176	0.7355	0.7122	0.5565
SVM	0.6604	0.7288	0.7844	0.7118	0.6824	0.7816	0.7577	0.8156	0.9389	0.6934	0.6998	0.5694
SMA	0.7416	0.8012	0.8768	0.7272	0.6845	0.7980	0.7806	0.8563	0.9348	0.7339	0.7008	0.5685
TSVM	0.8352	0.8512	0.9528	0.7697	0.8995	0.8996	0.8559	0.8964	0.8826	0.7380	0.6989	0.5843
pLWE	0.8584	0.8820	0.9520	0.7872	0.7217	0.8845	0.8330	0.9193	0.9664	0.7694	0.7008	0.5972
LWE	**0.8908**	**0.8844**	**0.9820**	**0.9744**	**0.9923**	**0.9823**	**0.9692**	**0.9816**	**0.9890**	**0.7967**	**0.7304**	**0.6852**

Mean Squared Error

Methods	Spam Filtering			20 Newsgroup						Reuters		
	U00	U01	U02	C vs S	R vs T	R vs S	S vs T	C vs R	C vs T	O vs Pe	O vs Pl	Pe vs Pl
WNN	0.1836	0.1713	0.1003	0.2775	0.2968	0.1575	0.1978	0.0851	0.0525	0.2462	0.3055	0.3774
LR	0.1944	0.1672	0.1013	0.2057	0.2036	0.1567	0.1624	0.1340	0.0613	0.2190	0.2444	0.3900
SVM	0.2374	0.1890	0.1489	0.2140	0.2353	0.1644	0.1826	0.1360	0.0453	0.2217	0.2230	0.2827
SMA	0.1556	0.1337	0.0870	0.2030	0.2183	0.1349	0.1614	0.0979	0.0430	0.1987	0.2318	0.3049
TSVM	0.1428	0.1394	0.0814	0.1749	**0.1080**	0.1128	0.1281	0.1198	0.1061	0.2250	0.2128	0.2688
pLWE	0.1218	0.1012	0.0550	0.1795	0.2027	0.1029	0.1399	0.0699	0.0302	0.1845	0.2333	0.3000
LWE	**0.0988**	**0.1022**	**0.0333**	**0.0965**	0.1409	**0.0384**	**0.0534**	**0.0308**	**0.0140**	**0.1678**	**0.2120**	**0.2091**

Table 3: Performance Comparison on Intrusion Detection Data Set

Accuracy

Intrusions	DOS		Probing		R2L		ALL		SMA	pLWE	LWE
	SVM	TSVM	SVM	TSVM	SVM	TSVM	SVM	TSVM			
DOS	NA	NA	0.9334	0.9352	0.9547	0.9303	0.9294	0.9281	0.9512	0.9609	**0.9623**
Probing	0.8171	0.7820	NA	NA	0.6599	0.8384	0.5808	0.8433	0.5444	0.9627	**0.9636**
R2L	0.5551	0.7602	0.7873	0.8215	NA	NA	0.7615	**0.9036**	0.5360	0.8020	0.8024

AUC

Intrusions	DOS		Probing		R2L		ALL		SMA	pLWE	LWE
	SVM	TSVM	SVM	TSVM	SVM	TSVM	SVM	TSVM			
DOS	NA	NA	0.9774	0.9797	0.9287	0.9188	0.9755	0.9543	0.9854	0.9858	**0.9862**
Probing	0.8877	0.8572	NA	NA	0.5001	0.8982	0.8160	0.8866	0.9745	0.9772	**0.9793**
R2L	0.7114	0.8077	0.9206	0.8727	NA	NA	0.8717	**0.9435**	0.9221	0.9399	0.9418

lems where training and testing data have different distributions, the proposed locally weighted ensemble approach greatly outperforms supervised, semi-supervised single-model algorithms, and a simple averaging ensemble.

Performance Study.

The results of the toy problem introduced in Figure 1 are summarized in Figure 4. The results of linear SVM on the training sets from two domains are the top two on the left, denoted as M_1 and M_2. Due to the difference between training and test distributions, both make incorrect predictions at "mirrored" areas. After merging the training sets, the SVM model ("ALL" on top right) still does not work and the constructed hyperplane is obviously a horizontal line. This is due to the fact that there exist conflicting concepts in the merged training set. On the other hand, transductive SVM (TSVM bottom left) trained on merged training sets fails as well since the label propagation is confused by the conflicting training examples. Simple averaging of M_1 and M_2, shown as "SMA" (bottom middle) also makes mistakes in the uncertain areas. However, examples incorrectly classified by these methods are now correctly predicted by the locally weighted ensemble approach (LWE) and the decision boundary matches the V-shape well. To see how this works, first, the clustering algorithm discovers the two clusters above and below the V-shape. For any example $\mathbf{x} \in R_1$, its neighbors in the cluster contain the examples in all three regions R_1, R_2 and R_3. At the same time, its neighbors predicted by M_1 are those examples $\in R_1$ and R_3. Importantly, its neighbors predicted by M_2 are only examples $\in R_1$. Since there are more common neighbors between the clustering structure and M_1, M_1 will be given higher weight at $\mathbf{x}$. Thus, according to M_1, the examples in R_1 are classified to be negative. Similarly, M_2 will be chosen to predict examples $\in R_2$ as negative. In summary, by weighting the two models locally according to the degree of consistency between models and clusters, the examples at the uncertain areas are predicted correctly.

Results of all the methods on the Email Spam Filtering, 20 Newsgroup and Reuters sets are summarized in Table 2 with best results shown in bold font. Refer to Table 1 for the details of each task. It is clearly seen that, for all tasks and using any performance measure, the locally weighted ensemble method (LWE) significantly improves the transfer learning performance compared with other baseline methods. We can observe that most of the transfer learning problems are tough due to the unknown discrepancy between the training and the test distributions. The single-model methods (WNN, LR, SVM) usually have poor performance with accuracy around 0.7 and mean squared error greater than 0.1 on most of the tasks. The simple model averaging algorithm using uniform weights can help reduce the expected error compared with single models. However, its performance is not quite satisfactory since they only rely on the labeled information from the source domain and make no efforts in selecting useful information and transferring the knowledge into the test domain. By incorporating the structure information of the test set into learning, the transductive learning approach can beat the supervised learning methods most of the times. But we can see more improvement achieved by using the proposed locally weighted ensemble framework. After the first step of combining classifiers by weighting them judiciously, both accuracy and mean squared error are improved over all the baselines. Then propagating confident predictions along the clustering structure in the test set can significantly boost

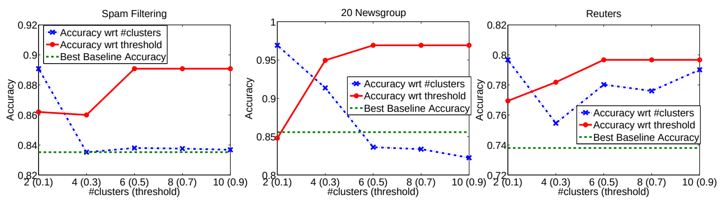

Figure 5: Parameter Sensitivity

the performance further. As an example, on the "C vs S" data set in the 20 newsgroup collection, the worst single model only achieves around 66% accuracy whereas the best single model makes correct predictions for 73% of the test examples. The tranductive SVM improves the accuracy to around 77% and LWE outperforms all the other methods by an impressive 97% accuracy. In most of the experiments, the improvement in accuracy after utilizing weighted ensemble is over 10% and up to 30% for some problems. The experimental results on these transfer learning tasks demonstrate the benefits of the empirical approximation of the optimal locally weighted ensemble framework. Both per-example weighting scheme and the adjustment step in the framework can successfully filter out the "harmful" labeled information, and thus help make the most reliable predictions.

Table 3 presents the performance of all methods on the three tasks of intrusion detection. Each row corresponds to a learning problem characterized by the test domain and the other two domains act as training, as discussed in Section 4.1. Besides the two training domains, a simple combination of examples from the two domains (represented as "ALL") could be another source of training. Based on each training source, we test the performance of SVM and TSVM on the test domain. We also build two single models from each training domain and combine them using uniform weights, which corresponds to SMA. The proposed pLWE and LWE are shown in the last two columns. For the first two learning tasks, it is obvious that the proposed LWE shows dominance for both accuracy and AUC. Especially on the test set of "Probing", the two training domains seem to be conflicting with each other, thus both the models trained from a union of the two domains and the simple averaging of the two models result in an accuracy around 50% to 60%. LWE achieves 96.36% accuracy by choosing the useful information from the two models. On the last learning task, the algorithm TSVM trained on the combination of training domains wins over the proposed method, which may be due to the fact that one of the single models we are combining has insufficient amount of examples to be relied on. We note that the worst single model's accuracy is around 56% and the simple averaging method even degrades to having 54% accuracy. Based on such weak classifiers, we could still improve the accuracy to 80%. In the future, we will explore more strategies to detect the cases when we should combine the source domains rather than building individual models.

Parameter Sensitivity.

There are two important parameters in the proposed algorithm, the number of clusters c' in the test set and the selection threshold δ to filter the predictions with low confidence. The traditional way of setting parameters through cross-validation cannot work when the training and test distributions are different. Again, since the true target function of the test domain is not known, there may not have effective methods to find the optimal values of the parameters. So here, we just give some sensitivity experimental results and state some basic principles in setting the parameters. We choose one cross-domain learning problem from each of the three data sets: email spam filtering, and 20 newsgroup and Reuters set, and the results are shown in Figure 5. We vary c' from 2 to 10 and δ from 0.1 to 0.9, and put both of them on the x-axis. We compare the accuracy of LWE approach when the parameters vary, with that of the best accuracy achieved by the baseline methods. We fix $\delta = 0.7$ when changing c', and let $c' = 2$ when tuning δ. It is clearly seen that when the threshold rises from 0.1 to 0.5, the learning performances on all three sets are gradually improving. After the point of 0.5, the performances maintain stable. This suggests that a low threshold is not desirable since many inaccurate predictions from the supervised models would be used in the adjustment step. Therefore 0.5 up to 1 could be a reasonable range to select the threshold δ. However, the users could choose to lower down or raise the threshold to match their beliefs in the abilities of the supervised models. As for the number of clusters c', the best performances in the experiments are achieved when $c' = 2$. When c' goes up, the over-fitting could occur when the number of examples in each cluster is not sufficient enough to give an accurate estimate of the model weights, and thus we could observe a drop in accuracy. We could also note that in spite of the changes caused by parameter variation, the proposed LWE improves over the best baseline method most of the time.

5. RELATED WORK

The problem with different training and test distributions started gaining much attention very recently. When it is assumed that the two distributions differ only in $P(\mathbf{x})$ but not in $P(y|\mathbf{x})$, the problem is referred to as *covariate shift* [25, 18] or *sample selection bias* [14]. The instance weighting approaches [25, 18, 5] try to re-weight each training example with $\frac{P_{\text{test}}(\mathbf{x})}{P_{\text{train}}(\mathbf{x})}$ and maximize the re-weighted log likelihood. Another line of work tries to change the representation of the observation $\mathbf{x}$ hoping that the distributions of the training and the test examples will become very similar after the transformation [3, 24]. [22] transforms the model learned from the training examples into a Bayesian prior to be applied to the learning process on the test domain. The major difference between our work and these studies is that they depend on a single source of information and try to learn a global single model that adapts well to the test set.

Constructing a good ensemble of classifiers has been an active research area in supervised learning [12]. By combining decisions from individual classifiers, ensembles can usually reduce variance and achieve higher accuracy than individual classifiers. Such methods include Bayesian averaging [17], bagging, boosting and many variants of ensemble approaches [2, 27, 13, 15]. Some ensemble methods assign weights locally [1, 19], but such weights are determined based on training data only. There has not been much work on ensemble methods to address the transfer learning problem. In [11, 26], it is assumed that the training and the test examples are generated from a mixture of different models, and the test distribution has different mixture coefficients than the training distribution. In [23], a Dirichlet Process prior is used to couple the parameters of several models from the same parameterized family of distributions. [10] extends the boosting method to perform transfer learning. Bennett *et al.* [4] proposed a methodology for building a meta-classifier which combines multiple distinct classifiers through the use of reliability indicators. The proposed weighted ensemble provides a more general framework for transfer learning because 1) the base models can be heterogeneous and can be any generative or discriminative models, and 2) the method does not depend on specific applications and makes no assumption about the form of distributions generating the training or the test data.

Multi-task learning(MTL) [7], which learns several related tasks at the same time with a shared representation, considers single $P(\mathbf{x})$ and multiple output variables, so the basic setting is different from our problem. The "clustering" assumption in our work is exploited in some transfer learning and semi-supervised learning works [9, 28], where clustering structure is utilized in smoothing predictions among neighbors. Our paper differs from these papers by utilizing the assumption in weighting different models locally to combine all sources of labeled information for knowledge transfer.

6. CONCLUSION

Knowledge transfer across domains with different distributions is an important problem in data mining that has not been fully investigated. In this work, we take advantage of the different predictive powers of several models trained on different domains or using different learning algorithms. We propose a locally weighted ensemble framework to transfer the combined knowledge to a new domain that is different from all the training domains. Importantly, the base models can be constructed by traditional learning algorithms not specifically designed for transfer learning. We analyze the optimality on expected error reduction by utilizing the locally weighted ensemble framework as compared to both single models and globally weighted ensembles. Based on the "clustering" assumption that the local structure of the test set is related to $P(y|\mathbf{x})$, we design an effective weighting scheme to approximate the optimal model weights. This is formulated by comparing the neighborhood graphs of each model with those from clustering. The experimental results on four real transfer learning data sets show that the proposed method improves over each base model 10% to 30% in accuracy and is more accurate than both semi-supervised learning and simple model averaging models. These results indicate that: 1) the locally weighted ensemble could successfully identify the knowledge from each model that is useful to predict in the test domain and transfer such information from all available base models; and 2) the proposed graph-based weight estimation method makes the framework practical by effectively approximating the optimal model weights. In the future, we plan to compare LWE with existing single-model based transfer learning algorithms, as well as to explore effective methods to set parameter values.

7. REFERENCES

[1] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 2004.

[3] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Proc. of NIPS' 07*, pages 137–144. 2007.

[4] P. N. Bennett, S. T. Dumais, and E. Horvitz. The combination of text classifiers using reliability indicators. *Information Retrieval*, 8(1):67–100, 2005.

[5] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proc. of ICML' 07*, pages 81–88, 2007.

[6] A. J. Carlson, C. M. Cumby, J. L. R. Nicholas D. Rizzolo, and D. Roth. Snow learning architecture. http://l2r.cs.uiuc.edu/~cogcomp/asoftware.php?skey =SNOW#projects.

[7] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[8] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[9] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu. Co-clustering based classification for out-of-domain documents. In *Proc. of KDD' 07*, pages 210–219, 2007.

[10] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proc. of ICML' 07*, pages 193–200.

[11] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.

[12] T. Dietterich. Ensemble methods in machine learning. In *Proc. of MCS '00*, pages 1–15, 2000.

[13] W. Fan. Systematic data selection to mine concept-drifting data streams. In *Proc. KDD' 04*, pages 128–137, 2004.

[14] W. Fan and I. Davidson. On sample selection bias and its efficient correction via model averaging and unlabeled examples. In *Proc. of SDM'07*.

[15] J. Gao, W. Fan, and J. Han. On appropriate assumptions to mine data streams: Analysis and practice. In *Proc. ICDM' 07*, pages 143–152, 2007.

[16] A. Genkin, D. D. Lewis, and D. Madigan. Bbr: Bayesian logistic regression software. http://stat.rutgers.edu/~madigan/BBR/.

[17] J. Hoeting, D. Madigan, A. Raftery, and C. Volinsky. Bayesian model averaging: a tutorial. *Statist. Sci.*, 14:382–417, 1999.

[18] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *Proc. of NIPS' 06*, pages 601–608. 2007.

[19] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[20] T. Joachims. Making large-scale svm learning practical. advances in kernel methods - support vector learning. *MIT-Press*, 1999.

[21] G. Karypis. Cluto - family of data clustering software tools. http://glaros.dtc.umn.edu/gkhome/views/cluto.

[22] X. Li and J. Bilmes. A Bayesian divergence prior for classifier adaptation. In *Proc. of AISTATS' 07*, 2007.

[23] D. M. Roy and L. P. Kaelbling. Efficient bayesian task-level transfer learning. In *Proc. of IJCAI '07*.

[24] S. Satpal and S. Sarawagi. Domain adaptation of conditional probability models via feature subsetting. In *Proc. of ECML/PKDD' 07*, pages 224–235, 2007.

[25] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[26] A. Storkey and M. Sugiyama. Mixture regression for covariate shift. In *Proc. of NIPS' 06*, pages 1337–1344.

[27] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proc. of KDD'03*, pages 226–235, 2003.

[28] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.