

Predictive Modeling with Heterogeneous Sources

Xiaoxiao Shi* Qi Liu† Wei Fan‡ Qiang Yang§ Philip S. Yu¶

Abstract

Lack of labeled training examples is a common problem for many applications. At the same time, there is often an abundance of labeled data from related tasks, although they have different distributions and outputs (e.g., different class labels, and different scales of regression values). In the medical domain, for example, we may have a limited number of vaccine efficacy examples against a new swine flu H1N1 epidemic, whereas there exists a large amount of labeled vaccine data from previous years' flu. However, it is difficult to directly apply the older flu vaccine data as training examples because of the difference in data distribution and efficacy output criteria between different viruses. To increase the sources of labeled data, we propose a method to utilize these examples whose marginal distribution and output criteria can be different. The idea is to first select a subset of source examples similar in distribution to the target data; all the selected instances are then “re-scaled” and assigned new output values from the labeled space of the target task. A new predictive model is built on the enlarged training set. We derive a generalization bound that specifically considers distribution difference and further evaluate the model on a number of applications. For an siRNA efficacy prediction problem, we extract examples from 4 heterogeneous regression tasks and 2 classification tasks to learn the target model, and achieve an average improvement of 30% in accuracy.

Keywords: Predictive Model, Heterogeneous Learning, Transfer Learning

1 Introduction

Most learning techniques require that the training data and the test data have homogeneous outputs; that is, the data share either the same set of class labels, or

the same scale of regression values. In many applications such as biological sequence annotation, however, training data satisfying the requirement are still too limited to build an accurate learning model. At the same time, there can be an abundance of related source data with heterogeneous outputs (i.e., different class labels, different scales of regression values, different meanings of regression values, and the like). To improve predictive accuracy, we aim at developing a new framework so that the training data with heterogeneous outputs can be utilized.

An example in biology is illustrated in Fig 1. The goal is to obtain the efficacy outputs of different siRNAs [8], which is related to RNA inference¹ from molecular biology. In reality, there are several bio-research groups working on the problem but in different applications (e.g., research group 1 may mainly detect the outputs of plant siRNAs, while research group 3 may study animal siRNAs). Note that in practice, to reduce the tremendous cost² of bio-experiments, each research group will first train a learning model to predict the outputs, and only select the siRNAs with certain predicted values to perform true bio-tests. However, it is sometimes very difficult for some research groups to obtain sufficient training data for the learning models. For instance, it may be very time-consuming to analyze the animal siRNAs, which leaves research group 3 with a limited number of training examples only. This in turn makes it difficult to train an accurate prediction model based on group 3 alone. In this case, guided by the predicted outputs, research group 3 may have to take lots of expensive but unfruitful bio-tests (false positive), whereas at the same time, miss lots of important samples (false negative). To improve the predictive accuracy, one may thus intend to take advantage of the training data publicly available from other research groups. For instance, can the examples

*Computer Science Department, University of Illinois at Chicago, USA. xiaoxiao@cs.uic.edu.

†College of Life Science and Biotechnology, Tongji University, China. Part of the work was done when the author was a post-doc at HKUST. qiliu@tongji.edu.cn

‡IBM T.J.Watson Research, USA. weifan@us.ibm.com

§Department of Computer Science, Hong Kong University of Science and Technology. qyang@cse.ust.hk

¶Computer Science Department, University of Illinois at Chicago, USA. psyu@uic.edu

¹To be specific, RNA inference problem aims at design of efficient exogenous siRNAs with highly efficacy when binding them to other specific RNAs to decrease their activities. It thus requires to detect the binding efficacy of different siRNAs in different conditions. We abbreviate it as “efficacy outputs”.

²For example, Ontario Institute has to spend approximately \$32 million each year on molecular tests, from <http://www.oicr.on.ca/Portalnews/Vol3.Issue2/molecular.htm>

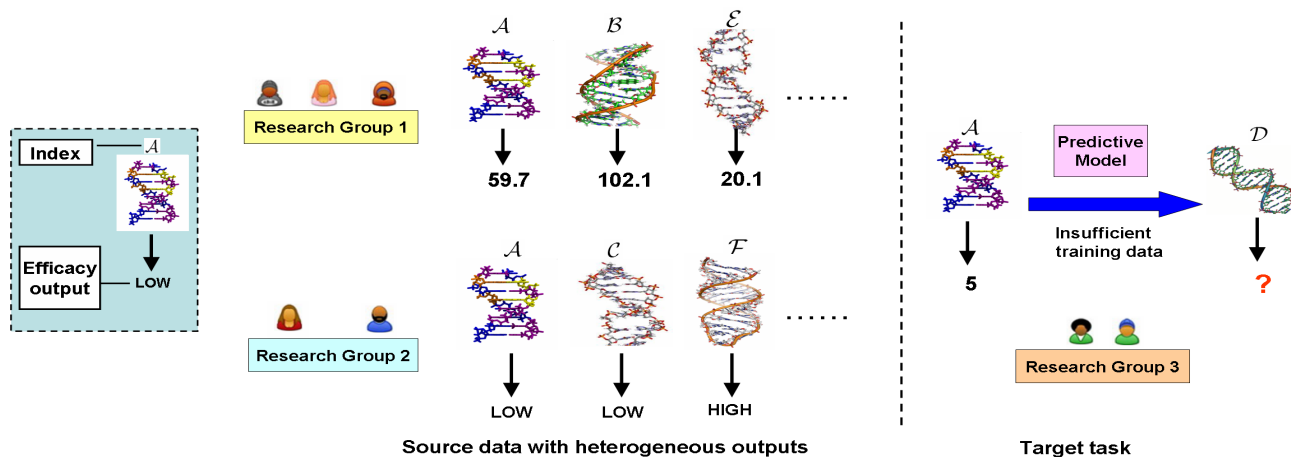


Figure 1: How to take advantage of the examples in research group 1 and research group 2 to help predict the efficacy output of siRNA molecule D in research group 3? Difficulties: (a) heterogeneous outputs because of different experimental conditions (e.g., siRNA molecule A has different outputs in 3 research groups); (b) different data distributions because of different preferences on the siRNA samples [14].

from research group 1 and group 2 help predict the output of molecule D in research group 3?

However, there are at least two obstacles. First, because of different experimental conditions (e.g., different equipments, different siRNA concentrations, and the like [14]), the outputs are heterogeneous among the data sets from different research groups. For example, siRNA molecule A has output value 59.7 as detected by research group 1, while only 5 as provided by research group 3, and even just a class category “LOW” in the data set from research group 2. Note that each of these efficacy outputs is valid and meaningful in the corresponding data set. However, the heterogeneity among the outputs makes the labeled examples apparently useless to other research groups. Second, the distributions are different among the data sets because each research group may have its own preferences over siRNA samples derived from different applications [14]. The source data can thus be too different from the target data to improve learning accuracy. This example will be further studied in the experiment section. Similar situations can be also found in the following applications:

- **Social Network.** For example, in social tagging applications (also called “social bookmarking”), users save links to web pages that they want to remember and share. These social tags are usually public, but they can vary over different bookmarking systems, such as ODP, Wikipedia, Backflip, Blink, and the like. One interesting problem is that can we integrate different bookmarking systems to help predict social tags for a new system given that their outputs (social tags) are heterogeneous?

- **Applied Sociology** (e.g., housing price census). Can the suburban housing price census data help predict the downtown housing prices? Note that they have different social backgrounds and thus different housing data distributions. Furthermore, because of different economic situations, they can have very different prices upon the same type of houses, which makes their outputs heterogeneous. Similar applications include health census data across different areas, different ages, and the like.
- **Transportation System Vehicle Heading Prediction.** In sustainable transportation systems, one important task is to estimate the vehicle’s destination using the average GPS heading and trajectory data. Since there are different types of vehicles (taxi, lorries, private sedans, buses) and each vehicle has unique set of possible destinations (e.g., buses have fixed stations while lorries may have another set of fixed routes), it forms large set of available heterogeneous sources to estimate the most likely destination given the current trip information of a vehicle.

Problem Formulation: In this paper, we aim at solving the above problems which extract examples from different sources with heterogeneous outputs, to better learn the target task. As a focus theme, we consider the target task to be a regression problem given that a regression model can be easily extended to solve classification problem by logistic regression. To be specific, the following problems are studied:

1. Can the source regression data help predict another regression task given that (a) they have differ-

ent data distributions, and (b) they have heterogeneous outputs, i.e., different output scales, different meanings of the regression values?

- Can classification data help learn regression task given that (a) they have different data distributions and (b) there is no direct transformation from class labels to the target regression values?

Denote the target training set as $\mathcal{L} = (\mathbf{X}, \mathbf{Y})$, where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m)^T$ denotes the data array (\mathbf{x}_i is a column vector) and $\mathbf{Y} = (y_1, y_2, \dots, y_m)^T$ is the corresponding regression outputs; further denote the target test set as $\mathcal{U} = (\mathbf{u}_1, \dots, \mathbf{u}_t)^T$ where \mathbf{u}_i is the column vector data array, and the size of \mathcal{U} is far larger than that of the training set (i.e., $|\mathcal{U}| \gg |\mathbf{X}|$). In the process of evaluating the prediction quality, one can also access the true outputs of $\forall \mathbf{u}_i \in \mathcal{U}$. The goal is thus to predict regression outputs for $\forall \mathbf{u}_i \in \mathcal{U}$ such that the predictive value is close to the true value. Also denote the n source tasks as $\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_n\}$, and $\mathbf{S}_i = (\mathbf{W}_i, \mathbf{V}_i)$ where \mathbf{W}_i is the data array as \mathbf{X} , and \mathbf{V}_i is the corresponding outputs. Note that the outputs of the source data and those of the target data are allowed to be heterogeneous, defined as the following.

DEFINITION 1.1. *Given the target training set \mathcal{L} and a source data set \mathbf{S}_t , if $\exists (\mathbf{x}, y) \in \mathcal{L}$ and $(\mathbf{w}, v) \in \mathbf{S}_t$ where \mathbf{x}, \mathbf{w} are the data array, y and v are the corresponding outputs, and $\mathbf{x} = \mathbf{w}$ but $y \neq v$, the output spaces of \mathcal{L} and \mathbf{S}_t are said to be **heterogeneous**; otherwise, they are **homogeneous**.*

In our problem setting, we do not assume the output spaces of source data and target data to be homogeneous. In other words, even the same data can have different outputs in different data sets. Note that one straightforward method to “unify” the heterogeneous outputs is to apply min-max normalization [17] to transform the source outputs into the same scale as the target outputs. However, this straightforward idea implicitly requires that the source output spaces must be linear to the target output space, which is usually not true in our problem setting. We theoretically study this strategy in Lemma 4.1 and empirically evaluate its performance in the discussion section.

Our Solution: To take advantage of the examples from various sources with heterogeneous outputs, there are two key problems to answer. First, how do we handle different data distributions? In the proposed model, we develop an algorithm on the basis of the clustering based Kullback-Leibler divergence [13], to sample source training data that are similar to the target data in distribution. The general idea is to perform

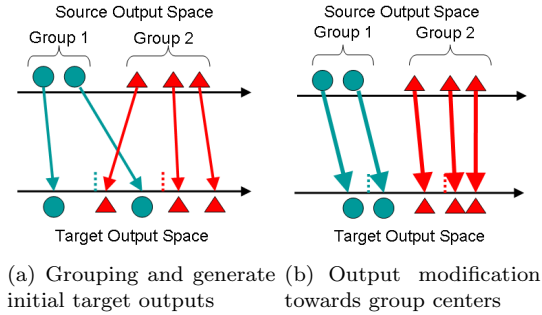


Figure 2: Generate Regression Output (Grouping can be directly generated from class labels, or from clustering on the source regression values; Group centers are marked with dash lines in the target output space.)

clustering on the combined data (i.e., source and target data), and then select the clusters of examples that can induce a small KL divergence according to Lemma 2.1 and Lemma 2.2 in Section 2.1. Another important problem is how to judiciously assign the selected instances with new outputs from the label space of the target task. Simply put, our question is how to transform the source outputs into the target outputs? To solve similar heterogeneous-output transformation problems, we first present a principle in the following and further develop a method based on the principle.

PROPOSITION 1.1. *Similarity Preserving Principle for Heterogeneous-output Transformation. Denote the source output space as \mathbf{V} and the target output space as \mathbf{Y} , the transformation function $f : \mathbf{V} \rightarrow \mathbf{Y}$ is similarity preserving, if for $\forall \mathbf{v}_1, \mathbf{v}_2 \in \mathbf{V}$, $\pi(\mathbf{v}_1, \mathbf{v}_2) = \pi(f(\mathbf{v}_1), f(\mathbf{v}_2))$, where $\pi(y_1, y_2)$ is a similarity indicator that returns true when y_1, y_2 are similar.*

This principle requires the examples similar in the source output space to be also similar in the new target output space. Note that there are various interpretations of similarity. We apply a straightforward definition on the basis of clustering: the data in the same cluster are said to be similar. Based on this strategy, we first generate initial outputs of the source instances by the regression model learnt from the initial training set \mathcal{L} ; we then group the source data in the source output space (Fig 2(a)) and modify their assigned outputs towards their group centers to preserve their original similarity (Fig 2(b)). At last, the selected source examples, coupled with their transformed regression outputs, are included into the target training set \mathcal{L} to build a new regression model. We derive a generation bound for the proposed model in Theorem 2.1 and further evaluate it in 7 biology and 10 housing census data sets. For example, in one of the biology data sets, the proposed

heterogeneous regression model employs only 1% of target training data to achieve the same accuracy as the comparison method with 60% of training examples.

2 Heterogeneous Regression Model

The general framework of the proposed method HEGS is described in Algorithm 1. For each of the source tasks, HEGS first selects a subset of examples that is similar to the target data in distribution evaluated by the KL divergence (Step 3). The algorithm then generates new outputs for each of the selected instances in Step 4. With the new training data (Step 5), the algorithm then returns a regression model in Step 7. It is important to note that we can apply various regression models in Step 1 and Step 7. HEGS can thus be regarded as a wrapper algorithm for regression models to train with heterogeneous examples. In this paper, we employ linear ridge regression as an example.

Linear Ridge Regression Given a training set $\mathcal{L} = (\mathbf{X}, \mathbf{Y})$, where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m)^T$ and $\mathbf{Y} = (y_1, y_2, \dots, y_m)^T$, a linear ridge regression model minimizes the following function:

$$(2.1) \quad \mathcal{J}(\mathbf{w}) = \sum_i (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|^2$$

where λ is a positive regularization parameter that controls the trade-off between the bias and variance of the estimate. [2] further shows that the predicted value, i.e., $\mathbf{w}^T \mathbf{x}$, for a new unlabeled data \mathbf{x} can be written as:

$$(2.2) \quad \mathbf{Y}^T (\mathbf{K} + \lambda \mathbf{I})^{-1} \kappa$$

where $\mathbf{K}_{i,j} = \mathbf{x}_i^T \mathbf{x}_j$ and $\kappa_i = \mathbf{x}_i^T \mathbf{x}$, in which kernel trick can be easily applied. In this paper, we focus on how to take advantage of different sources with heterogeneous outputs to help build the target regression model. We thus simply employ basic linear ridge regression model as an example, and compare it as a baseline in the experiment. With the above linear model which can be applied in Step 1 and Step 7, we next describe how to perform sample selection in Step 3; and how to judiciously generate new outputs for the selected examples in Step 4.

2.1 KL Divergence based Sample Selection HEGS (Algorithm 1) extracts examples from various sources to help learn the regression model. It is thus very important to handle different distributions among the data. To do so, a novel KL divergence based sample selection algorithm is proposed to only draw the source examples that are similar to the target data in distribution. The first step is to perform clustering on the combined data (source data and the whole target data)

Input: Target training data $\mathcal{L} = (\mathbf{X}, \mathbf{Y})$; Target test data \mathcal{U} ; # Source tasks n ; Source tasks $\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_n\}$

Output: Regression model parameter \mathbf{w}

```

1  $\mathbf{w}_0 \leftarrow \text{regression}(\mathcal{L})$  // Initial parameter
2 for each  $\mathbf{S}_i \in \mathbf{S}$  do
3    $\tilde{\mathbf{D}}_i \leftarrow \min_{\mathbf{D} \subseteq \mathbf{S}_i} \text{KL}(\mathcal{U} \cup \mathbf{X} \|\mathbf{D})$ . // Algorithm 2
4    $\tilde{\mathbf{Y}}_i \leftarrow \text{getOutput}(\mathbf{w}_0, \tilde{\mathbf{D}}_i)$ . // Algorithm 3
5    $\mathcal{L} \leftarrow \mathcal{L} \cup (\tilde{\mathbf{D}}_i, \tilde{\mathbf{Y}}_i)$ 
6 end
7 Return  $\mathbf{w} \leftarrow \text{regression}(\mathcal{L})$ 

```

Algorithm 1: Main framework of **HE**terogeneous **Re**Gre**S**sion (**HEGS**)

as shown in Fig 3. Intuitively, if a subset of source data and target data are similar, they will be assigned into the same cluster. Based on this intuition, we should select the clusters of examples where source data and target data are about the same proportion (evenly mixed), while the clusters that strongly bias on either source data or target data should not be selected, as cluster C_3 in Fig 3. We next derive Lemma 2.1 and Lemma 2.2 to model the intuitive idea with the clustering-based KL divergence [13] originally used to evaluate the differences of distributions. We first introduce a new formula for the clustering-based KL divergence in Lemma 2.1, in order to extend the new formula into an incremental version in Lemma 2.2, which can be run efficiently to perform sample selection.

LEMMA 2.1. Clustering-based KL divergence: denote the current sample set as \mathbf{D} , the target data set as \mathbf{T} , the traditional KL divergence

$$(2.3) \quad \text{KL}(\mathbf{T} \|\mathbf{D}) = \sum_x \mathbb{P}_{\mathbf{T}}(x) \log \frac{\mathbb{P}_{\mathbf{T}}(x)}{\mathbb{P}_{\mathbf{D}}(x)}$$

where $\mathbb{P}_{\mathbf{T}}(x), \mathbb{P}_{\mathbf{D}}(x)$ denote the distribution of x in \mathbf{T} and \mathbf{D} accordingly, can be rewritten with a new formula:

$$(2.4) \quad \text{KL}_c(\mathbf{T} \|\mathbf{D}) = \frac{2}{|\mathbf{T}|} \mathbb{U} + \log \frac{|\mathbf{D}|}{|\mathbf{T}|}$$

s.t. $\forall c, \mathbb{E}_{x \in \mathbf{T}, x \in c}[x] = \mathbb{E}_{x \in \mathbf{D}, x \in c}[x]$

where $|\mathbf{T}|, |\mathbf{D}|$ denote the data size of \mathbf{T} and \mathbf{D} accordingly, $\mathbb{E}_{x \in \mathbf{T}, x \in c}[x]$ denotes the centroid of data from \mathbf{T} in cluster c , and \mathbb{U} is defined as the following:

$$(2.5) \quad \mathbb{U} = \sum_C \left(\frac{|\mathbf{T} \cap \mathbf{C}|^2}{|\mathbf{C}|} \log \frac{|\mathbf{T} \cap \mathbf{C}|}{|\mathbf{D} \cap \mathbf{C}|} \right)$$

where \mathbf{C} denotes the cluster of the combined data $\mathbf{T} \cup \mathbf{D}$.

Proof. From [13], the clustering-based KL divergence can be written as:

$$(2.6) \quad \begin{aligned} & \mathbf{KL}_c(\mathbf{T}||\mathbf{D}) \\ &= \frac{1}{\mathbb{E}(\mathbf{T})} \left(\sum_{\mathbf{C}} (P'_{\mathbf{T}}(\mathbf{C})S(\mathbf{T}, \mathbf{C}) \log \frac{S(\mathbf{T}, \mathbf{C})}{S(\mathbf{D}, \mathbf{C})}) \right. \\ & \quad \left. + \sum_{\mathbf{C}} (P'_{\mathbf{T}}(\mathbf{C})S(\mathbf{T}, \mathbf{C}) \log \frac{P'_{\mathbf{T}}(\mathbf{C})}{P'_{\mathbf{D}}(\mathbf{C})}) \right) + \log \frac{\mathbb{E}(\mathbf{D})}{\mathbb{E}(\mathbf{T})} \end{aligned}$$

$$s.t. \quad \forall c, \quad \mathbb{E}_{x \in \mathbf{T}, x \in c}[x] = \mathbb{E}_{x \in \mathbf{D}, x \in c}[x]$$

where \mathbf{C} is the cluster generated from the combined data set $\mathbf{T} \cup \mathbf{D}$, and

$$S(\mathbf{T}, \mathbf{C}) = \frac{|\mathbf{T} \cap \mathbf{C}|}{|\mathbf{C}|}, P'_{\mathbf{T}}(\mathbf{C}) = \frac{|\mathbf{T} \cap \mathbf{C}|}{|\mathbf{T}| + |\mathbf{D}|},$$

$$\mathbb{E}(\mathbf{T}) = \frac{|\mathbf{T}|}{|\mathbf{T}| + |\mathbf{D}|}$$

Likewise the definitions of $S(\mathbf{D}, \mathbf{C})$, $P'_{\mathbf{D}}(\mathbf{C})$ and $\mathbb{E}(\mathbf{D})$ (by replacing \mathbf{T} with \mathbf{D} in the nominator).

In Eq (2.6), note that $\frac{S(\mathbf{T}, \mathbf{C})}{S(\mathbf{D}, \mathbf{C})} = \frac{P'_{\mathbf{T}}(\mathbf{C})}{P'_{\mathbf{D}}(\mathbf{C})}$. It can thus be written as

$$\begin{aligned} & \mathbf{KL}_c(\mathbf{T}||\mathbf{D}) \\ &= \frac{2}{\mathbb{E}(\mathbf{T})} \left(\sum_{\mathbf{C}} (P'_{\mathbf{T}}(\mathbf{C})S(\mathbf{T}, \mathbf{C}) \log \frac{P'_{\mathbf{T}}(\mathbf{C})}{P'_{\mathbf{D}}(\mathbf{C})}) \right) + \log \frac{\mathbb{E}(\mathbf{D})}{\mathbb{E}(\mathbf{T})} \\ &= \frac{2(|\mathbf{T}| + |\mathbf{D}|)}{|\mathbf{T}|} \left(\sum_{\mathbf{C}} \left(\frac{|\mathbf{T} \cap \mathbf{C}|}{|\mathbf{T}| + |\mathbf{D}|} \frac{|\mathbf{T} \cap \mathbf{C}|}{|\mathbf{C}|} \log \frac{|\mathbf{T} \cap \mathbf{C}|}{|\mathbf{D} \cap \mathbf{C}|} \right) \right) \\ & \quad + \log \frac{\mathbb{E}(\mathbf{D})}{\mathbb{E}(\mathbf{T})} \\ &= \frac{2}{|\mathbf{T}|} \mathbb{U} + \log \frac{|\mathbf{D}|}{|\mathbf{T}|} \end{aligned}$$

■

One of the important advantages of the new formula of KL divergence in Eq (2.4) is that we avoid to calculate the marginal distribution $\mathbb{P}_{\mathbf{T}}(x)$ and $\mathbb{P}_{\mathbf{D}}(x)$ explicitly as in Eq (2.3), which is normally difficult to obtain without prior knowledge. Instead, only some basic statistics of the clusters are applied to calculate the KL divergence. For example, $|\mathbf{T} \cap \mathbf{C}|$ in Eq (2.5) represents the number of examples in \mathbf{T} that are contained in cluster \mathbf{C} , which is illustrated in Fig 3. Note that the basic intuition of the clustering-based KL divergence is that similar data will be assigned into the same cluster. In this way, if the source and target data are similar, they will be evenly mixed in most clusters. A similar intuition is employed

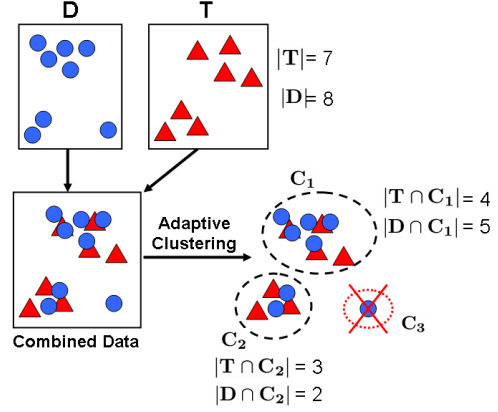


Figure 3: Notation illustration for Lemma 2.1, and the intuition illustration for Algorithm 2: cluster \mathbf{C}_3 will not be selected because it biases on \mathbf{D} , which will make the KL value large according to Lemma 2.1.

in [11] to tackle query classification problem except that clustering is replaced by an intermediate taxonomy.

With the new formula in Lemma 2.1, we derive the sample selection method in Algorithm 2. The basic idea is to select the clusters of examples that can induce a small KL divergence. It is interesting to note that this strategy biases toward the clusters where source data and target data are about the same proportion. We illustrate the idea in Fig 3 in which cluster \mathbf{C}_3 will not be selected because the sub-term $\log \frac{|\mathbf{D}|}{|\mathbf{T}|}$ tends to infinity, which makes KL very large. In the KL based sample selection algorithm, the first procedure is to perform an adaptive clustering on the combined data. It applies k-means to divide a cluster into two sub-clusters recursively until one of the following conditions meet: (1) the cluster contains data only from the source task, or only from the target task; or (2) $\|\mathbb{E}_{\mathbf{x} \in \mathbf{T}, \mathbf{x} \in c}[\mathbf{x}] - \mathbb{E}_{\mathbf{x} \in \mathbf{D}, \mathbf{x} \in c}[\mathbf{x}]\| \rightarrow 0$, where $\mathbb{E}_{\mathbf{I}}[\mathbf{x}]$ denotes the centroid/mean value of $\mathbf{x} \in \mathbf{I}$. These two criteria are the sufficient conditions of Lemma 2.1 [13], which can also help adaptively determine the number of clusters. Algorithm 2 then selects those clusters that can help reduce the KL divergence. It first selects the cluster that can induce the least KL divergence in Step 11; if the new KL is larger than the original KL (Step 12), the algorithm terminates because there is no other cluster that can decrease the KL; otherwise, the cluster will be included into the sample set (Step 15). To calculate the KL divergence more efficiently in Step 11, we derive Lemma 2.2 to incrementally obtain the KL when adding a new cluster of examples into the sample set.

LEMMA 2.2. *Denote the current sample set with t clusters of examples as \mathbf{D}_t , the target data set as \mathbf{T}_t . As-*

sume that a cluster of examples $\tilde{\mathbf{C}} = \tilde{\mathbf{C}}_{\mathbf{T}} \cup \tilde{\mathbf{C}}_{\mathbf{D}}$ are being considered to be included into the sample set, where $\tilde{\mathbf{C}}_{\mathbf{T}}$ denotes the target data contained in the cluster $\tilde{\mathbf{C}}$, $\tilde{\mathbf{C}}_{\mathbf{D}}$ denotes the source data contained in the cluster. The new KL divergence is

$$(2.7) \quad \begin{aligned} & \text{KL}((\mathbf{T}_t \cup \tilde{\mathbf{C}}_{\mathbf{T}}) \| (\mathbf{D}_t \cup \tilde{\mathbf{C}}_{\mathbf{D}})) \\ &= \frac{2}{|\mathbf{T}_t| + |\tilde{\mathbf{C}}_{\mathbf{T}}|} (\mathcal{U}_t + \frac{|\tilde{\mathbf{C}}_{\mathbf{T}}|^2}{|\tilde{\mathbf{C}}|} \log \frac{|\tilde{\mathbf{C}}_{\mathbf{T}}|}{|\tilde{\mathbf{C}}_{\mathbf{D}}|}) + \log \frac{|\mathbf{D}_t| + |\tilde{\mathbf{C}}_{\mathbf{D}}|}{|\mathbf{T}_t| + |\tilde{\mathbf{C}}_{\mathbf{T}}|} \end{aligned}$$

Proof. According to Lemma 2.1 and Eq (2.6), $\text{KL}((\mathbf{T}_t \cup \tilde{\mathbf{C}}_{\mathbf{T}}) \| (\mathbf{D}_t \cup \tilde{\mathbf{C}}_{\mathbf{D}}))$ can be extended as:

$$(2.8) \quad \begin{aligned} & \text{KL}((\mathbf{T}_t \cup \tilde{\mathbf{C}}_{\mathbf{T}}) \| (\mathbf{D}_t \cup \tilde{\mathbf{C}}_{\mathbf{D}})) \\ &= \frac{2}{|\mathbf{T}_t| + |\tilde{\mathbf{C}}_{\mathbf{T}}|} \left(\sum_{\mathbf{C}} \left(\frac{|\mathbf{T}_t \cap \mathbf{C}|^2}{|\mathbf{C}|} \log \frac{|\mathbf{T}_t \cap \mathbf{C}|}{|\mathbf{D}_t \cap \mathbf{C}|} \right) \right. \\ & \quad \left. + |\tilde{\mathbf{C}}_{\mathbf{T}}| \frac{|\tilde{\mathbf{C}}_{\mathbf{T}}|}{|\tilde{\mathbf{C}}|} \log \frac{|\tilde{\mathbf{C}}_{\mathbf{T}}|}{|\tilde{\mathbf{C}}_{\mathbf{D}}|} \right) + \log \frac{|\mathbf{D}_t| + |\tilde{\mathbf{C}}_{\mathbf{D}}|}{|\mathbf{T}_t| + |\tilde{\mathbf{C}}_{\mathbf{T}}|} \\ &= \frac{2}{|\mathbf{T}_t| + |\tilde{\mathbf{C}}_{\mathbf{T}}|} (\mathcal{U}_t + \frac{|\tilde{\mathbf{C}}_{\mathbf{T}}|^2}{|\tilde{\mathbf{C}}|} \log \frac{|\tilde{\mathbf{C}}_{\mathbf{T}}|}{|\tilde{\mathbf{C}}_{\mathbf{D}}|}) + \log \frac{|\mathbf{D}_t| + |\tilde{\mathbf{C}}_{\mathbf{D}}|}{|\mathbf{T}_t| + |\tilde{\mathbf{C}}_{\mathbf{T}}|} \end{aligned}$$

If the cluster $\tilde{\mathbf{C}}$ is selected into the sample set, we can also incrementally update the following terms:

$$(2.9) \quad \begin{aligned} \mathbf{D}_{t+1} &= \mathbf{D}_t \cup \tilde{\mathbf{C}}_{\mathbf{D}} \\ \mathbf{T}_{t+1} &= \mathbf{T}_t \cup \tilde{\mathbf{C}}_{\mathbf{T}} \\ \mathcal{U}_{t+1} &= \mathcal{U}_t + \frac{|\tilde{\mathbf{C}}_{\mathbf{T}}|^2}{|\tilde{\mathbf{C}}|} \log \frac{|\tilde{\mathbf{C}}_{\mathbf{T}}|}{|\tilde{\mathbf{C}}_{\mathbf{D}}|} \end{aligned}$$

■

With Lemma 2.2, the new KL divergence can be incrementally calculated in the loop between Step 10 and Step 16 without re-calculating the clustering statistics of the previous sample set. This improvement helps reduce the running time in the loop from $O(n^2)$ to $O(n)$ where n is the number of clusters adaptively decided in Step 1 to Step 7. It is thus an important step to enable the clustering based KL divergence to perform sample selection efficiently in Algorithm 2, especially for large data sets and for streaming data. With the selected sample set, we next introduce how to judiciously generate their new outputs in the target output space.

2.2 Similarity Preserving Output Generation

The KL based sample selection algorithm selects the source data that are similar to the target data in distribution. However, these examples cannot be used as

```

Input: Target data set (labeled and
          unlabeled)  $\mathbf{T}$ ; source data set  $\mathbf{S}$ 
Output:  $\tilde{\mathbf{D}} = \min_{\mathbf{D} \subseteq \mathbf{S}} \text{KL}(\mathbf{T} \| \mathbf{D})$ 

// Adaptive clustering for Lemma 1
1  $\mathbf{C} \leftarrow \{\mathbf{T} \cup \mathbf{S}\}$ ;
2 for each  $\mathbf{C}_i \in \mathbf{C}$  do
3   if  $\mathbf{S} \cap \mathbf{C}_i \neq \emptyset$  and  $\mathbf{T} \cap \mathbf{C}_i \neq \emptyset$  and
    $\|\mathbb{E}_{\mathbf{x} \in \mathbf{T}, \mathbf{x} \in \mathbf{C}_i}[\mathbf{x}] - \mathbb{E}_{\mathbf{x} \in \mathbf{S}, \mathbf{x} \in \mathbf{C}_i}[\mathbf{x}]\| > \theta$  then
4      $\{\mathbf{C}_{+1}, \mathbf{C}_{+2}\} \leftarrow \text{Kmeans}(\mathbf{C}_i, 2)$ ;
     /* Replace the cluster  $\mathbf{C}_i$  with its
     two sub clusters. */
5      $\mathbf{C} \leftarrow \mathbf{C} \setminus \{\mathbf{C}_i\}$ ;  $\mathbf{C} \leftarrow \mathbf{C} \cup \{\mathbf{C}_{+1}, \mathbf{C}_{+2}\}$ 
6   end
7 end
// Select the clusters that reduce KL
8  $\hat{\mathbf{C}}^* \leftarrow \min_{\hat{\mathbf{C}} \in \mathbf{C}} \text{KL}(\hat{\mathbf{C}}_{\mathbf{T}} \| \hat{\mathbf{C}}_{\mathbf{D}})$  where  $\hat{\mathbf{C}}_{\mathbf{T}}$  denotes
the target data contained in cluster  $\hat{\mathbf{C}}$ ,  $\hat{\mathbf{C}}_{\mathbf{D}}$ 
denotes the source data in  $\hat{\mathbf{C}}$ .
9  $\mathbf{C} \leftarrow \mathbf{C} \setminus \{\hat{\mathbf{C}}^*\}$ ;  $\tilde{\mathbf{D}} \leftarrow \hat{\mathbf{C}}_{\mathbf{D}}^*$ ;  $\tilde{\mathbf{T}} \leftarrow \hat{\mathbf{C}}_{\mathbf{T}}^*$ 
10 while  $\mathbf{C}$  is not empty do
11    $\tilde{\mathbf{C}}^* \leftarrow \min_{\tilde{\mathbf{C}} \in \mathbf{C}} \text{KL}((\tilde{\mathbf{T}} \cup \tilde{\mathbf{C}}_{\mathbf{T}}) \| (\tilde{\mathbf{D}} \cup \tilde{\mathbf{C}}_{\mathbf{D}}))$ 
   (Lemma 2.2)
12   if  $\text{KL}((\tilde{\mathbf{T}} \cup \tilde{\mathbf{C}}_{\mathbf{T}}^*) \| (\tilde{\mathbf{D}} \cup \tilde{\mathbf{C}}_{\mathbf{D}}^*)) > \text{KL}(\tilde{\mathbf{T}} \| \tilde{\mathbf{D}})$ 
   then
13     break;
14   end
   /* Include the whole cluster of
   examples into the training set */
15    $\tilde{\mathbf{D}} \leftarrow \tilde{\mathbf{D}} \cup \tilde{\mathbf{C}}_{\mathbf{D}}^*$ ;  $\mathbf{C} \leftarrow \mathbf{C} \setminus \{\tilde{\mathbf{C}}^*\}$ ;  $\tilde{\mathbf{T}} \leftarrow \tilde{\mathbf{T}} \cup \tilde{\mathbf{C}}_{\mathbf{T}}^*$ ;
   Update the other terms according to
   Eq (2.9).
16 end
17 Return  $\tilde{\mathbf{D}}$ 

```

Algorithm 2: KL based Sample Selection

training data directly because they have heterogeneous outputs with the target data. Algorithm 3 is thus derived to generate new outputs for the selected instances from the label space of the target task. The idea is to follow the Similarity Preserving Principle presented in Proposition 1.1. In other words, we aim at assigning new outputs to the source data where their similarity in the original output space can be preserved. Algorithm 3 first explores the similarity of the source examples, by grouping them on the source output space, as shown in Fig 2(a). Note that if the source task is a classification data set, the examples with the same class label are assigned into the same cluster. For regression tasks, we directly perform clustering on the regression values, and we further study the algorithm on the sensitivity to the

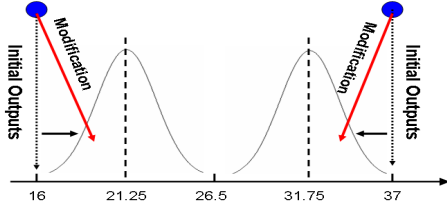


Figure 4: Example of generating new outputs

number of clusters in Fig 9(d) in the discussion section. After grouping, for each of the clusters, we calculate the initial output for each of its members, by directly using the regression model $t_1 = \mathbf{w}_0^T \mathbf{d}_1$, where \mathbf{d}_1 is the data array, and \mathbf{w}_0 is the regression parameter learnt from the initial target training data. According to the Similarity Preserving Principle, if \mathbf{d}_1 and \mathbf{d}_2 are similar in their initial output space, the new output t_2 of \mathbf{d}_2 should be also similar to t_1 of \mathbf{d}_1 . The rest of the algorithm (Step 6 to Step 9) thus modifies the values of t_1 and t_2 to make them closer to their center. We give an example in the following.

Example Let $\mathbf{d}_1 = (3, 2)^T$, $\mathbf{d}_2 = (7, 5)^T$, $\mathbf{w}_0 = (6, -1)^T$. Assume that $\mathbf{d}_1, \mathbf{d}_2$ are in the same cluster in their original output space, and the cluster contains only the two data points. We then have $t_1 = \mathbf{w}_0^T \mathbf{d}_1 = 16$, $t_2 = \mathbf{w}_0^T \mathbf{d}_2 = 37$ in Step 4. Step 6 calculates the center/mean value of t_1 and t_2 : $m = (16 + 37)/2 = 26.5$. Step 8 then deviates their regression values towards the center under a normal distribution as shown in Fig 4. The new regression values for \mathbf{d}_1 is $\tilde{y}_1 = \text{Norm}((16 + 26.5)/2, |26.5 - 16|/2) = \text{Norm}(21.25, 5.25)$; for \mathbf{d}_2 is $\tilde{y}_2 = \text{Norm}(31.75, 5.25)$. The reason to include a normal distribution in Step 8 is that we do not know the exact value of the new regression output. We thus allow some deviations to avoid over-deterministic, and further average the results in multiple runs in practice. Similar techniques can also be found in the research of uncertain data mining [1].

2.3 Generalization Bound To analyze HEGS (Algorithm 1) with generalization bound, we first define the loss function as the following:

DEFINITION 2.1. Given a linear system $\mathbf{w}^T \mathbf{X}$ where \mathbf{X} denotes the data array, for $\forall \eta \geq 0$, we define the loss function as:

$$(2.10) \quad \epsilon_{\eta, \mathbf{X}} = |\mathbf{X}|^{-1} \sum_{\mathbf{x}_i \in \mathbf{X}, (y_i - \mathbf{w}^T \mathbf{x}_i)^2 > \eta} \mathbf{1}$$

where y_i is the ground truth of the data \mathbf{x}_i , and $|\mathbf{X}|$ denotes the data size.

```

Input: Initial regression parameter  $\mathbf{w}_0$ ; selected
          source data  $\mathbf{D}$ 
Output: Transformed outputs  $\tilde{\mathbf{Y}}$  for  $\mathbf{D}$ 
/* 'grouping' in Fig 2(a). */
1  $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_n\} \leftarrow$  for all selected data
   $\mathbf{d} \in \mathbf{D}$ , grouping according to their outputs.
2 for each  $\mathbf{C}_i \in \mathbf{C}$  do
  /* 'initial outputs' in Fig 4. */
3   for each  $\mathbf{d}_p \in \mathbf{C}_i$  do
4      $t_p = \mathbf{w}_0^T \mathbf{d}_p$ 
5   end
6    $m_i \leftarrow |\mathbf{C}_i|^{-1} \sum_{\mathbf{d}_p \in \mathbf{C}_i} t_p$ 
  /* Fig 2(b) and Fig 4 */
7   for each  $\mathbf{d}_p \in \mathbf{C}_i$  do
8      $\tilde{y}_p \leftarrow \text{Norm}((m_i + t_p)/2, |m_i - t_p|/2)$ 
9   end
10 end
11 Return  $\tilde{\mathbf{Y}} = \{\tilde{y}_1, \tilde{y}_2, \dots, \tilde{y}_m\}^T$ 

```

Algorithm 3: Similarity Preserving Output Generation

Definition 2.1 defines “error” as the prediction with square error larger than η . The loss function is then the proportion of errors in the whole data set. It is interesting to note that with a large η , the regression model is more free to deviate from the true regression value; but when η is too small, the term ϵ_η will grow large because more examples are classified as mistakes. As a result, a typical regression model is to minimize $\epsilon(\eta) + \theta\eta$ where θ is a regularization parameter related to model complexity. For example, for simple linear regression, the optimal $\eta^* = \max_{\eta} (\min_{\mathbf{w}} (\eta = \|\mathbf{Y} - \mathbf{w}^T \mathbf{X}\|^2))$ for a large θ . With the definition of loss function, we employ and modify the error bound in [5] as the following theorem.

THEOREM 2.1. Let \mathcal{H} be a hypothesis space. Let \mathcal{U} be unlabeled samples of size m' . Let \mathbf{D} be a labeled sample of size m generated by drawing βm points from target data and $(1-\beta)m$ points from source data. If $\hat{h} \in \mathcal{H}$ is the empirical minimizer of $\epsilon_{\eta, \mathbf{D}}(h)$ on \mathbf{D} and $h^* = \min_{h \in \mathcal{H}} \epsilon_\eta(h)$ is the target risk minimizer, then given any $\eta > 0$, with probability at least $1 - \delta$ (over the choice of the samples),

$$\begin{aligned} \epsilon_{\eta, \mathbf{D}}(\hat{h}) &\leq \epsilon_\eta(h^*) \\ &+ 2\sqrt{\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}} \sqrt{\frac{g(\theta) \log(2m) - \log \delta}{2m}} \\ &+ 2(1-\alpha) \left(\frac{1}{2} \hat{d}_{\mathcal{H} \Delta \mathcal{H}}(\mathcal{U}, \mathbf{D}) + 4\sqrt{\frac{2g(\theta) \log m' + \log \frac{4}{\delta}}{m'}} + \lambda \right) \end{aligned}$$

Table 1: Description of the data sets (#Feature =161)

Order	Type	Size	Scale	References
1	Regression	2431	0~99.99	[8]
2	Regression	561	1~127.8	[8]
3	Regression	601	0~100	[8]
4	Regression	290	2.1~98	[15]
5	Regression	344	0.2~98.5	[15]
6	Classification	7443	4 classes	[10]
7	Classification	196	2 classes	[16]

Note: Some references, such as [8], refer to several data sets from different research groups

where α is the importance ratio between source and target task as in [5], $g(\theta) \propto \theta$ is a function to indicate model complexity similar to VC dimension in classification problem, and $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}, \mathbf{D})$ is the distribution differences between the two data sets also as defined in [5], and $\lambda = \min_{h \in \mathcal{H}} \epsilon_{\eta, \mathcal{U}}(h) + \epsilon_{\eta, \mathbf{D}}(h)$ denotes the combined risk of the optimal hypothesis.

The error bound includes three components: the first one is the error of the target task when it only uses its own training data; the second term reflects the complexity of the model; the last term is the differences between source and target data, and how the differences affect the result. In the proposed approach, the KL based sample selection algorithm is employed to minimize the difference between source and target data, which reduces the value of $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}, \mathbf{D})$; The similarity preserving output generation method tries to minimize the last term λ by generating outputs that balance the target task (by generating outputs from the label space of the target task as in Fig 2(a)), and the source task (by following the Similarity Preserving Principle as in Fig 2(b)) to minimize the combined risk.

3 Experiments

We empirically study the proposed heterogeneous regression model HEGS on two real world applications with 17 data sets in total. We mainly evaluate the regression accuracy by RMSE (root mean square error):

$$(3.11) \quad \text{RMSE} = \sqrt{n^{-1} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

where n is the number of test data, y_i is the real regression value and \hat{y}_i is the predicted value given by the model. Since linear ridge regression is embedded into the proposed model as an example, we thus set the traditional linear ridge method as a baseline.

3.1 Real World Data Sets The first application is the siRNA efficacy prediction problem discussed in

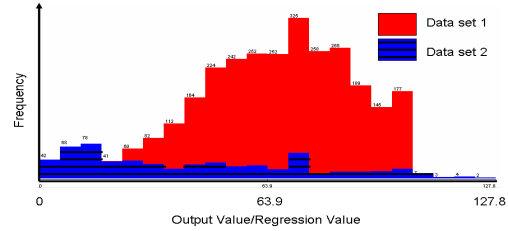


Figure 5: Distribution of the output values (biology)

Fig 1 where we have collected data sets from 7 different research groups (Table 1). The 7 data sets share the same feature space which describes the results of the same bio-tests of the instances. We set each of the 5 regression data sets as target task individually, while the remaining data sets (including 2 classification data sets and 4 regression data sets) as source tasks. Note that because of different bio-experimental conditions aforementioned, the outputs of different data sets are heterogeneous. Fig 5 illustrates how the regression values of the first two regression data sets differ from each other. Furthermore, each research group has its own preferences over siRNA sequence; the distributions among different data sets are thus very different [14]. The results summarized on 10 runs are reported in Fig 6, where in each run we randomly sample certain proportion (x-axes) of target data as training data while the rest are test data. It is clear that HEGS outperforms the baseline (traditional linear ridge regression without using the source data), especially when there is limited training data in the target task. For example, when there is only 5% of examples in data set 1, the RMSE of the comparison method is about 90, while that of HEGS is just about 25, about 70% more accurate than the comparison method. It is interesting to note that the performances of both methods do not drop with the increasing size of samples. It gets worse at the beginning until there is sufficient training data. This is because with limited labeled examples, the regression model can obtain little knowledge, which can easily reach a local optima in the model space. For HEGS, the local optima affects the process of generating new outputs for the source examples, which makes the accuracy drop. When the size of the training set grows, the model has more examples to learn from with the reducing error. It is also important to note that for some of the data sets, they require a lot of examples for the traditional method to train a relatively accurate model. For example, in data set 4, about 60% of training examples are needed for the baseline method to achieve the same accuracy as HEGS with only 1% of target training data.

The second data set is from the field of applied

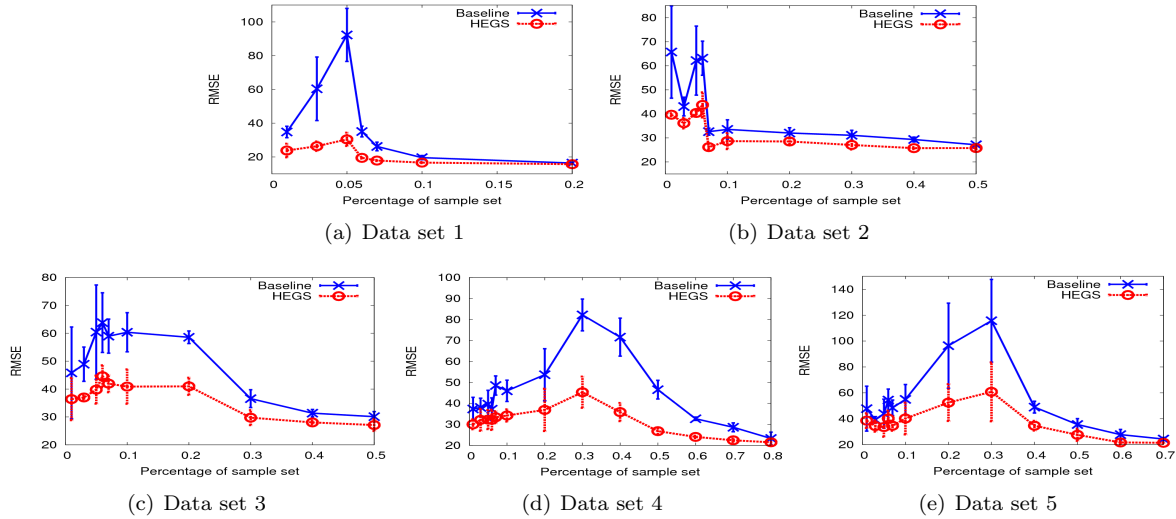


Figure 6: Experiments on siRNA efficacy prediction. The traditional linear ridge regression model is set to be the baseline because it is embedded in HEGS. We can thus focus on evaluating the ability of HEGS to integrate heterogeneous data, isolating the effect of different embedded regression models.

Table 2: Description of the data sets (#Feature =18)

Name	Size	Scale
Newton	18	2.47~21.46
Boston Roxbury	19	12.03~36.98
Lynn	22	6.58~27.71
Boston Savin Hill	23	15.17~34.02
Cambridge	30	1.73~29.53
Somerville	15	11.12~34.41
South Boston	10	3.53~18.46
Brookline	11	7.67~18.66
East Boston	11	10.29~19.01
Quincy	11	9.38~29.55



Figure 7: Distribution of the output values (housing)

sociology, which is generated from the housing census statistics³. The data was collected over 92 different regions in Boston area. We let the last attribute, which indicates the percentage of population of the corresponding types of house, be the regression label, and remove some discrete and uninformative attributes

³<http://stat.cmu.edu/datasets/boston.corrected.txt>

such as the region name, the instance index, and the like. Note that although the data cover 92 different regions, there are only very few data in some regions. For these regions, both the training data set and test data set are very small in size, which is insufficient to test the regression model. We thus only select the first 10 regions with the largest sample sets; and consider the first 5 regions, each of which the sample set is larger than 15, as the target task individually. We thus have 5 separate experiments on 5 regions (Newton, Boston Roxbury, Lynn, Boston Savin Hill, and Cambridge). For each of the experiments, we set the other 9 tasks as the source tasks. It is also important to mention that the output spaces of the 10 data sets are heterogeneous because of some complicated reasons such as neighborhood, religion, economy differences. Fig 7 illustrates the distributions of the output values of the first two data sets. The average experiment results on 10 runs are summarized in Fig 8. In most cases, HEGS outperforms the baseline especially when there is limited training data. For instance, in experiment 1, traditional linear ridge model requires 25% training examples to achieve the same accuracy as HEGS that has only 15% of the training data.

4 Discussion

We further discuss the proposed model HEGS in more detail, aiming at answering the following questions:

1. Can the source data be directly used as training samples without domain adaption such as done in HEGS?

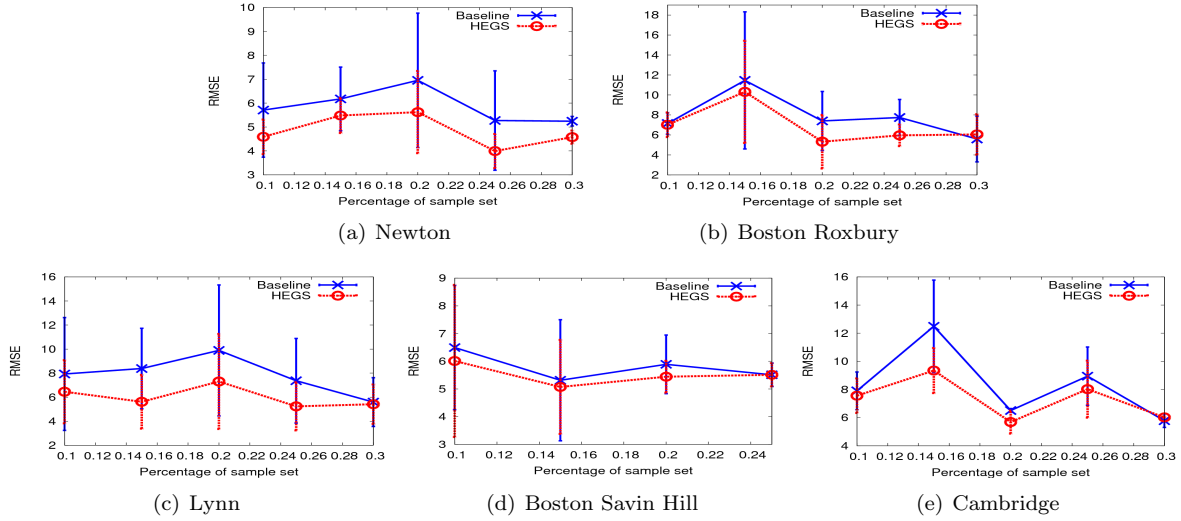


Figure 8: Experiments on Boston housing data set

2. Why are the KL based sample selection (Algorithm 2) and the similarity preserving output generation (Algorithm 3) necessary? For the output value transformation to target domain, can we just use the semi-supervised self-training strategy [20] or min-max normalization?
3. Is extracting examples from all the tasks better than only from the most related task?

To answer the first question, we apply the biology data sets in Table 1, in which the 5 regression tasks are set to be the target task individually, whereas the rest are source tasks. The results are plotted in Fig 9(a). We compare the strategy of directly using the source data (rightmost bar) with traditional regression model (leftmost bar), as well as the proposed model HEGS (middle bar). It is clear that the using the source examples can be very harmful to the learning. For example, in the first experiment, the error of the “direct” strategy is about 40, while that of the traditional method is about 20, and that of HEGS is only 16.6, over 50% more accurate than the “direct use” strategy. This is because the source data not only have different data distributions, but also have heterogeneous output spaces. Direct use of the source examples cannot improve predictive accuracy.

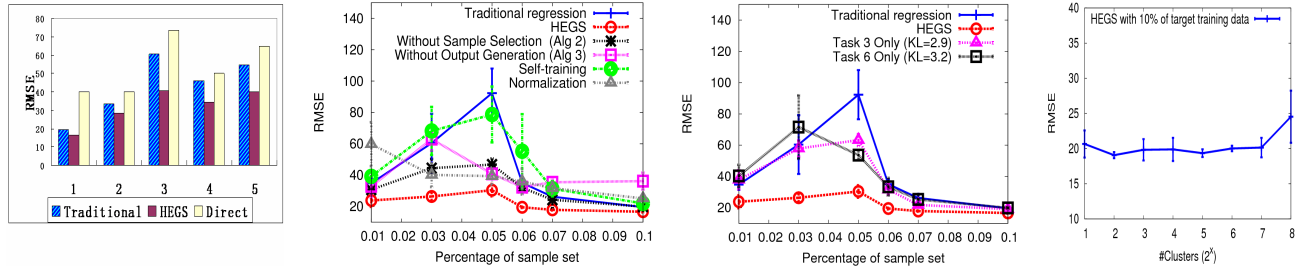
Fig 9(b) is used to study the necessity of the KL based sample selection process (Algorithm 2) and the similarity preserved output generation (Algorithm 3), where the first data set in the biology application is used as an example. In Fig 9(b), we observe that HEGS outperforms the other two strategies: one is to remove the sample selection process in HEGS; the other is to

remove the output generation algorithm. The reason of the result is that without the sample selection method, we may include some examples that are very different from the target data in distribution, which may hurt the learning accuracy; without the output generation process, it can be very difficult to find a consistent concept to model both the target and source data with heterogeneous outputs. Both the KL based sample selection method and the similarity preserving output generation are necessary to handle heterogeneous data. Furthermore, in Fig 9(b), it is also clear that HEGS outperforms the semi-supervised self-training strategy [20] which directly generates the outputs via the initial regression model as in Fig 2(a). This explains the necessity of the Similarity Preserving Principle (as in Fig 2(b)) to improve the accuracy. Interestingly, the similarity preserving output generation is also compared with the min-max normalization [17], another supervised output transformation algorithm. Note that although normalization is straightforward, we next prove that it implicitly requires the source output spaces must be linear to the target output space.

LEMMA 4.1. Denote the outputs of the source task as \mathbf{V} whose values are within the interval $[v_{min}, v_{max}]$, and let the outputs of the target task as \mathbf{Y} whose values are within the interval $[y_{min}, y_{max}]$. A normalization from \mathbf{V} to the same scale as \mathbf{Y} is defined to satisfy the following equation for $\forall v_0 \in \mathbf{V}$ [17]:

$$(4.12) \quad \frac{f(v_0) - y_{min}}{y_{max} - y_{min}} = \frac{v_0 - v_{min}}{v_{max} - v_{min}}$$

where $f : \mathbf{V} \rightarrow \mathbf{Y}$ is the transformation function. We next prove that it implicitly requires the linear



(a) Can the source data be directly used? (b) Why are the KL based sample selection and the similarity preserving output generation necessary? (c) Are examples from all tasks better than from single task? (d) Parameter sensitivity

Figure 9: Study of technical details

relationship between v_0 and its transformation value $y_0 = f(v_0)$; that is, we want to prove:

$$f(v_0) = wv_0 + c$$

where $v_0 \in \mathbf{V}$, w and c are constants.

Proof. The proof is straightforward. From Eq (4.12), we can get

$$(4.13) \quad f(v_0) = \frac{y_{max} - y_{min}}{v_{max} - v_{min}}v_0 + \left(\frac{y_{min}}{y_{max} - y_{min}} - \frac{v_{min}}{v_{max} - v_{min}} \right)$$

We then let $w = \frac{y_{max} - y_{min}}{v_{max} - v_{min}}$, $c = \left(\frac{y_{min}}{y_{max} - y_{min}} - \frac{v_{min}}{v_{max} - v_{min}} \right)$ which are all constants. ■

Lemma 4.1 reveals that an implicit assumption of the min-max normalization is that the source output spaces must be linear to the target output space. However, it is unrealistic to make this assumption in our problem setting since the target data and source data may be from very different applications. We replace the similarity preserving output generation algorithm with the normalization strategy in Algorithm 1, and compare it with HEGS in Fig 9(b). It is clear that the similarity preserving output generation algorithm outperforms the normalization strategy.

Another interesting issue is to study whether it is better to extract examples from all the tasks than only from the most related task. We also apply the first data set in the biological application as an example, the result of which is plotted in Fig 9(c). Note that we first calculate the KL divergence between the target data set and the source data set by Lemma 2.1, which reflects the correlation between the two data sets. The two source tasks with the smallest KL divergence are then used to report the result. From Fig 9(c), it is clear that the examples extracted from single task can more or less help improve the learning accuracy, compared

with the traditional regression model. However, examples integrated from all the tasks help achieve the most significant improvement. This is because we can extract different aspects of knowledge from different tasks, and the combination of them may achieve a better result.

Parameter Sensitivity: The only parameter in the proposed model is the number of clusters in the similarity preserving output generation algorithm; that is, the number of groups in the source output spaces. We use the first data set in the biology application to study the effect of #clusters on the algorithm. The results summarized on 10 runs is plotted in Fig 9(d), from which we can observe that the performance is not very sensitive to the number of clusters. Only when #clusters grows too large (e.g., $2^8 = 256$), the performance drops because there may be only 2 or 3 data in some of the clusters and clustering may become ineffective to find groups.

5 Related Works

Traditional supervised learning and semi-supervised learning work well under the assumption that training data and test data are independent and identically distributed (i.i.d.) and share feature and output spaces. Given that such kind of training examples are usually not sufficient enough to train accurate predictive models, transfer learning is then proposed to help build the target model by extracting knowledge from related data sets (e.g., [6, 3, 4, 12]). There are mainly two lines of research work. One mainly tackles the problem of different data distributions. For example, a general approach is based on re-sampling (e.g., [4]), where the motivation of it is to emphasize the examples that are discriminating and similar to the target data. There are also some other solutions such as transfer across feature subspaces (e.g., [3]), transfer across similar learning parameters (e.g., [9]), and the like. Another line of work further solves the case where the target and source data

have heterogeneous feature spaces. For example, [19] applies text data in social web to improve the performance of clustering on image data; [18] employs latent space to bridge the similarity among data with heterogeneous feature spaces, to improve ranking models; in [7], a model ensemble based approach is proposed to explore the consensus predictions among data with heterogeneous feature spaces.

Contrary to these works, we mainly study in a new scenario where the source and target data have heterogeneous outputs. Note that recently, [13] proposed a method to improve the partition of classification data by another classification task with different class labels. Different from this work, we are not restricted to that the training and test data are both from classification task. In other words, we use classification tasks to help build regression models, which may be more applicable and flexible to increase a wide range of training data for applications such as the example in Fig 1.

6 Conclusions

We study the problem where the training and test data have both heterogeneous outputs and different data distributions. We focus on regression problems, as the solution can be easily extended to classifications via logistic regression. The proposed model HEGS has two steps. First, a clustering based sample selection algorithm is proposed to select the clusters of examples where source and target data are similar, measured by an improved KL divergence criteria. Second, the selected source examples are assigned regression values in the target output space with the constraints to preserve their similarity. This is achieved by first grouping the source examples in their initial output spaces, and then modifying their target outputs towards their group centers. Finally, the selected instances, coupled with their transformed outputs, are included into the target training set to build a new regression model. The most important advantage of HEGS is that it can greatly increase the potential range of training data. As can be seen from the experimental section, we have achieved our objectives set out in the beginning of the paper. In the future, we will study other possibilities to perform output value transformation, and feature value transformation from source tasks to target task.

Acknowledgement Qiang Yang and Qi Liu thank NEC labs China and HKUST 4/CRF-SF/08 for their support.

References

[1] C. C. Aggarwal, P. S. Yu, *A Survey of Uncertain Data*

Algorithms and Applications. TKDE, **21(5)** 609-623 (2009)

[2] S. An, W. Liu, and S. Venkatesh, *Face recognition using kernel ridge regression*. In: CVPR 07 (2007)

[3] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, *Analysis of representations for domain adaptation*. In: NIPS 07 (2007)

[4] S. Bickel, J. Bogojeska, T. Lengauer, and T. Scheffer, *Multi-task learning for hiv therapy screening*. In: ICML 08 (2008)

[5] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman, *Learning Bounds for Domain Adaptation*. In: NIPS 07 (2007)

[6] R. Caruana, *Multitask learning*. Machine Learning **28(1)** 41-75 (1997)

[7] J. Gao, W. Fan, Y. Sun, and J. Han, *Heterogeneous source consensus learning via decision propagation and negotiation*. In: KDD 09 (2009)

[8] J. W. Klingelhoefer, L. Moutsianas, and C. Holmes, *Approximate bayesian feature selection on a large meta-dataset offers novel insights on factors that effect sirna potency*. Bioinformatics, **25**, 1594-1601 (2009)

[9] N. D. Lawrence, and J. C. Platt: *Learning to learn with the informative vector machine*. In: ICML 04 (2004)

[10] Y. Ren, W. Gong, Q. Xu, X. Zheng, D. Lin, Y. Wang, and T. Li, *siRecords: an extensive database of mammalian siRNAs with efficacy ratings*. Bioinformatics, **22(8)**, 1027-1028 (2006)

[11] D. Shen, J. Sun, Q. Yang, and Z. Chen, *Building bridges for web query classification*. In: SIGIR 06 (2006)

[12] X. Shi, W. Fan, and J. Ren, *Actively Transfer Domain Knowledge*. In: ECML/PKDD 08 (2008)

[13] X. Shi, W. Fan, Q. Yang, and J. Ren, *Relaxed Transfer of Different Classes via Spectral Partition*. In: ECML/PKDD 09 (2009)

[14] P. Sætrom, and O. Snøve, *A comparison of siRNA efficacy predictors*. Biochemical and Biophysical Research Communications, **321(1)**, 247-253 (2004)

[15] H. Tafer, S. Ameres, G. Obernosterer, C. Gebeshuber, R. Schroeder, J. Martinez, and I. Hofacker, *The impact of target site accessibility on the design of effective siRNAs*. Nature Biotechnology, **26(5)**, 578C583 (2008)

[16] S. Takasaki, Y. Kawamura, and A. Konagaya, *Selecting effective siRNA sequences by using radial basis function network and decision tree learning*. BMC bioinformatics, **7(Suppl 5)**, S22 (2006)

[17] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson International Edition (2005)

[18] B. Wang, J. Tang, W. Fan, S. Chen, Z. Yang, and Y. Liu *Heterogeneous Cross Domain Ranking in Latent Space*. In: CIKM 09 (2009)

[19] Q. Yang, Y. Chen, G. Xue, W. Dai, and Y. Yu, *Heterogeneous Transfer Learning for Image Clustering via the Social Web*. In: ACL and AFNLP 09 (2009)

[20] X. Zhu, *Semi-Supervised Learning Literature Survey*. Computer Sciences TR 1530, University of Wisconsin Madison (2008)