Compact Coding for Hyperplane Classifiers in Heterogeneous Environment

Hao Shao¹, Bin Tong¹, and Einoshin Suzuki²

¹ Graduate School of Systems Life Sciences, Kyushu University

² Department of Informatics, ISEE, Kyushu University

Abstract. Transfer learning techniques have witnessed a significant development in real applications where the knowledge from previous tasks are required to reduce the high cost of inquiring the labeled information for the target task. However, how to avoid *negative transfer* which happens due to different distributions of tasks in heterogeneous environment is still a open problem. In order to handle this kind of issue, we propose a Compact Coding method for Hyperplane Classifiers (CCHC) under a *two-level* framework in inductive transfer learning setting. Unlike traditional methods, we measure the similarities among tasks from the *macro level* perspective through minimum encoding. Particularly speaking, the degree of the similarity is represented by the relevant code length of the class boundary of each source task with respect to the target task. In addition, informative parts of the source tasks are adaptively selected in the *micro level* viewpoint to make the choice of the specific source task more accurate. Extensive experiments show the effectiveness of our algorithm in terms of the classification accuracy in both UCI and text data sets.

1 Introduction

Transfer learning [20] provides a solution of how to learn a target task in real applications where a large amount of auxiliary data from source domains are given. However, the negative transfer problem [13] tends to happen if the distributions of the source and the target domains are too dissimilar. A typical example is about language study, where learning English is the target task, and learning Japanese and French are regarded as the source tasks. It would be much effective for a learner to study English if he (she) is familiar with French. However, the learning process would be impeded if the learner only masters Japanese, because the lexical and semantic structure of Japanese is different from that of English. Negative transfer is more likely to happen once we underestimate the side effect resulting from the distribution differences of multiple source tasks [20], which is common in real applications.

To solve this problem, it is reasonable to measure the similarity between tasks and instances in heterogeneous environment [2,9,15,19] by considering the distribution dis(similarity) of different domains. For example, [15] proposed a dictionary-based compression dissimilarity measure between two instances in

different domains for clustering. In this paper, we confine our focus on the classification problems using hyperplane lassifiers. Attempts to avoid negative transfer are made in transfer learning for classification by evaluating the distance between tasks [2,3] or distance between two instances [22,26,4] in different domains. For example, [2] tried to find a common representation among different groups of tasks which can be regarded as the transferred information. A novel kernel function is sophisticatedly designed in [4] by exploiting the Gaussian process model. Although this method is able to model the *negative* similarity between two instances, it can be only applied to one source task. We note that also, measuring the similarity between only tasks has a limited power, since the negative transfer would be induced when some instances are considered “too dissimilar” in one task compared to another, even if the degree of the dissimilarity between the two tasks is very small. In addition, in the typical setting of the transfer learning, the labeled information is insufficient in the target task. It is difficult to get a reliable classifier by only considering the similarities between tasks [3].

Unlike the traditional methods above, we establish a different paradigm by proposing a method that consists of *macro level* and *micro level* evaluations, in order to spot relevant portions of the source tasks with the target task. Our fundamental idea is to measure the similarity in the *macro level* (task level) by compact coding which is inspired by the Minimum Description Length Principle (MDLP) [16]. The MDL Principle is built with a solid theoretical foundation that is suitable for model selection to avoid *overfitting*. It is successfully applied in the inductive transfer learning problem [10], but only confined to one source task and one target task. Put it another way, it is radical to consider that how to select informative knowledge from multiple source tasks can be conceptually regarded the same as the model selection. In our method, informative parts of the source tasks are adaptively selected in the *micro level* viewpoint to make the choice of the specific source task more accurate.

2 Problem Setting and Preliminaries for Encoding

In this paper, we deal with the inductive transfer learning problem where several source tasks are available. There is a task set $\mathbb{S}$ from the source domain which contains K tasks S_i ($i = 1, \dots, K$), one target training data set T and one test data set E from the target domain. Instances in S_i and T are labeled. All instances in S_i , T and E share the same attribute vector of m dimensions. From each data set, we obtain a hyperplane classifier $\mathbf{w}\mathbf{x} = 0$, where $\mathbf{x} = (x_1, x_2, \dots, x_m, 1)$. The weight vectors of hyperplanes for the source tasks are denoted by $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K$, where $\mathbf{w}_i = (w_i^1, w_i^2, \dots, w_i^m)$ and our objective is to find the weight vector $\mathbf{w}_t$ for the target task. In the transfer learning setting, the number of labeled instances in the target domain is assumed to be much smaller than that in the source domain. Although the labeled data is not sufficient to obtain an accurate hyperplane classifier on T , we can still draw a rough estimation using the limited labeled data, which is denoted by $\mathbf{v}_t$.

We provide here the preliminaries for MDLP, which may be viewed as a principle for avoiding overfitting, i.e., it is a means to balance the simplicity of a

classifier and its goodness-of-fit to the data [7,11]. It states that the best classifier h_{best} is given as follows.

$$h_{\text{best}} = \arg \min_h (-\log P(h) - \log P(D|h)) \quad (1)$$

where h is a hypothesis on a data set D , and $P(h)$ and $P(D|h)$ represent the probability that h occurs and the conditional probability that D occurs given h , respectively. Consider the problem of encoding h as a binary string. According to the coding theory [6], the length of the code string for h using an optimally efficient code is $-\log P(h)$. Similarly, $-\log P(D|h)$ may be regarded as the length of the code string for D encoded with the help of h . In the MDLP for classification, these code lengths are calculated in a problem setting where the receiver has D except for the class labels of examples in D . The sender first sends h , then the class labels of examples in D with the help of h . The code used in this setting is a prefix code, in which no extension of a code word can itself be a code word. Intuitively, the hypothesis chosen by the MDLP coincides with the maximum a posteriori hypothesis.

The MDLP can be interpreted as assigning priors to theories based on a compact coding, i.e., $P(h)$ is defined by the coding method for calculating $-\log P(h)$. It is important that the coding method is efficient otherwise the assigned priors deviate from the philosophy of the MDLP.

We then consider a problem of encoding a binary string of length a which consists of b binary 1s and $(a - b)$ binary 0s under the framework of the sender and the receiver problem. An obvious method is to send the number b of binary 1s with the code length $\log(a + 1)$ then specify the positions of binary 1s with the code length $\log \binom{a}{b}$ [7,11]. We hereafter call this method a binary coefficient method and denote the required code length with $\Theta(a, b)$ as follows.

$$\Theta(a, b) \equiv \log(a + 1) + \log \binom{a}{b}$$

3 Compact Coding for Hyperplane Classifiers in Heterogeneous Environment

We provide in this section a detailed explanation of the coding scheme for our CCHC. Given a small number of labeled instances in the target domain, it is difficult to measure the similarity between a source task and the target task. Suppose there are two source tasks S_1 , S_2 and one target training data set T , as shown in Fig. 1. The corresponding weight vectors of the hyperplanes are $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{v}_t$, $\mathbf{w}_1 = \{1, 1, -3\}$, $\mathbf{w}_2 = \{1, 0, -1\}$ and $\mathbf{v}_t = \{1, 0, -2\}$.

Intuitively, by applying $\mathbf{w}_1$ and $\mathbf{w}_2$ to T , we see from Fig. 1 that both $\mathbf{w}_1$ and $\mathbf{w}_2$ have 1 wrong prediction on T and it is difficult to tell which one is more similar to the target domain if no other information is provided.

To deal with such a problem, our main idea is to divide the classification task into a two-level evaluation: in the macro level, we evaluate the degrees of

Fig. 1. An example of the three hyperplanes

similarity of the source tasks with the target task and in the micro level, we try to select useful parts of a relevant source task. In such a way, we intend to avoid negative transfer by adaptively selecting more relevant data from the source domain for training the classifier, and exclude those data which bring negative effects on the results.

Our algorithm includes two sub procedures:

1. **Macro level:** Sort S_i in descending order on the degrees of similarity with the target data set T .
2. **Micro level:** Divide the data set of the related source task into several components and select related parts to help training the classifier in the target domain.

3.1 Macro Level: Arrange Related Tasks

The objective of the macro level evaluation is to sort the source tasks based on the degrees of similarity with T . In the general MDLP framework, the model space contains candidates of models induced from the data, and the best model is obtained by minimizing the two-part code length. The model that fits the data more is assigned with a shorter code length. Consider $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K$ are in the model space. Since each of the $\mathbf{w}_i$ is induced from the source task S_i , we call $P(\mathbf{w}_i|S_i)$ a *a posteriori* probability for the source task S_i and a $\mathbf{w}_i$ that compresses the data well has a higher $P(\mathbf{w}_i|S_i)$. By substituting S_i with T , we infer that if $P(\mathbf{w}_i|T)$ is high, $\mathbf{w}_i$ is assumed to be a good hypothesis which can be induced from T . In such a case, we may say that S_i and T are similar. Therefore, $P(\mathbf{w}_i|T)$ could be used as a degree of similarity between S_i and T . Taking the negative logarithm function, the best model $\mathbf{w}^*$ to explain the data T is the one that minimizes the sum of the length, in bits, of the description of the model and the length of the description of the data when encoded with the help of the model. We could sort $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K$ in descending order of $P(\mathbf{w}_i|T)$ as follows:

$$P(\mathbf{w}_i|T) \propto P(T|\mathbf{w}_i)P(\mathbf{w}_i)$$

Or in ascending order of the code length $(-\log P(\mathbf{w}_i) - \log P(T|\mathbf{w}_i))$.

Although the labeled information in T is not sufficient to obtain the optimal hyperplane in the target domain, $\mathbf{v}_t$ could be adopted to improve the performance as it is a rough estimation of the underlying hyperplane. We believe that it can be used to help sorting the source tasks by evaluating the degree of similarity between a source task S_i and T . In the learning process, it can be iteratively updated using transferred information from the source domain. Therefore, we take $\mathbf{v}_t$ into consideration to help coding the hyperplanes of the source tasks. We need to sort $\mathbf{w}_i$ in descending order of the posterior probability as follows:

$$P(\mathbf{w}_i|T, \mathbf{v}_t) \propto P(T, \mathbf{v}_t|\mathbf{w}_i)P(\mathbf{w}_i)$$

It can be further decomposed as:

$$\begin{aligned} P(\mathbf{w}_i|T, \mathbf{v}_t) &\propto P(T|\mathbf{w}_i)P(\mathbf{v}_t|\mathbf{w}_i)P(\mathbf{w}_i) \\ &\propto P(T|\mathbf{w}_i)P(\mathbf{w}_i|\mathbf{v}_t)P(\mathbf{v}_t) \\ &\propto P(T|\mathbf{w}_i)P(\mathbf{w}_i|\mathbf{v}_t) \end{aligned}$$

where $P(\mathbf{v}_t)$ is neglected in the third line because we could not settle a proper prior for $\mathbf{v}_t$ due to the small size of T , and it is of the same value when we compare different $\mathbf{w}_i$ given the specific $\mathbf{v}_t$.

By taking the negative log function,

$$L_i = -\log P(\mathbf{w}_i|\mathbf{v}_t) - \log P(T|\mathbf{w}_i) \quad (2)$$

The best $\mathbf{w}^*$ is the one that minimizes the two-part code length among all the models. Note that by compact coding, an optimal code assigns shorter codes to the model that suits the data most while others are given longer codes. This principle also coincides with the principle of minimum encoding. As an extension of the coding method that searches for the best hypothesis among all the candidates, we could use the code length as a similarity measure. Therefore, the degree of similarity between one source task S_i using the class boundary $\mathbf{w}_i$ in terms of the target task T can be represented by the code length as (2).

The first part is the complexity of one model given another and the second part can be treated as the negative log-likelihood of the data given the model. It is considered to be effective because the hyperplanes are concise summaries of the data, and in the classification tasks, it makes sense to use class boundaries in terms of T to measure the similarities between a source task and the target task.

Now let's consider the calculation of the first part by first introducing how to encode a real number x under the assumption that $x = \mu$ is most likely, where μ is also a real number. Let f be a continuous probability measure on the real numbers. Here we legitimately assume that f is a Gaussian, $f = 1/(2\pi\sigma^2)^{1/2} \exp(-(x - \mu)^2/2\sigma^2)$. Then we need to determine the value of σ .

The probability of x with precision ε is thus [25]:

$$P(x) = \int_{x-\frac{\varepsilon}{2}}^{x+\frac{\varepsilon}{2}} f(x)dx \approx \varepsilon f(x) \quad (3)$$

To obtain the variance σ , we assume that the probability of $x = \mu$ is q ($0 < q < 1$), so we have

$$P(x = \mu) = \varepsilon \frac{1}{(2\pi\sigma^2)^{1/2}} = q \quad (4)$$

From (4), we can have that σ equals to $\varepsilon/(2\pi q^2)^{1/2}$, and in the calculation, we need to set two parameters ε and q . q ought to be high because in our coding scheme, x is more likely to take the value around μ . We set $\varepsilon = 0.01$ and $q = 0.8$ throughout the paper. Note that, a general precision is adequate for our similarity measure to distinguish different hyperplanes because the code lengths would be uniform for each hyperplane under the same precision.

Let $A(x, u)$ be the code length of sending x given μ ,

$$A(x, u) = -\log \left(\varepsilon \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \right) = -\log \left(q e^{-\frac{\pi(x-\mu)^2 q^2}{\varepsilon}} \right) \quad (5)$$

Now we consider how to calculate $-\log P(\mathbf{w}_i)$, with the help of $\mathbf{v}_t$. $\mathbf{v}_t$ is taken into consideration because we want to use as much information as possible that can be inferred from the small number of labeled instances in the target task. Note that both $\mathbf{w}_i$ and $\mathbf{v}_t$ contain m entries, $\mathbf{w}_i = (w_i^1, w_i^2, \dots, w_i^m)$ and $\mathbf{v}_t = (v_t^1, v_t^2, \dots, v_t^m)$. Consider the sender and the receiver's problem, both the sender and the receiver have $\mathbf{v}_t$ in advance. To send $\mathbf{w}_i$, we assume that each entry in $\mathbf{w}_i$ is more likely to be approximate to the corresponding entry in $\mathbf{v}_t$. Therefore, the code length of $-\log P(\mathbf{w}_i)$ with the help of $\mathbf{v}_t$ should be

$$-\log P(\mathbf{w}_i | \mathbf{v}_t) = \sum_{j=1}^m A(w_i^j, v_t^j) \quad (6)$$

For the second part of (2), the class labels of all instances in T form a binary string, and thus it is convenient for us to send only the wrong predictions of $\mathbf{w}_i$ on this string. Let $\omega(\mathbf{w}_i, T)$ denote the number of misclassified examples on T ,

$$-\log P(T | \mathbf{w}_i) := \Theta(|T|, \omega(\mathbf{w}_i, T)) \quad (7)$$

Combining equations (6) and (7), we have the code length L_i for $\mathbf{w}_i$ on T :

$$L_i = \sum_{j=1}^m A(w_i^j, v_t^j) + \Theta(|T|, \omega(\mathbf{w}_i, T)) \quad (8)$$

Then the source data sets could be arranged in ascending order on the code lengths with the target data set T . We have L_{min} as the shortest code length among the k code lengths.

For the example in Table 1, by calculating the code lengths for each of the hyperplane, we could obtain the following results:

$$L_1 = \Theta(|T|, \omega(\mathbf{w}_1, T)) + \sum_{j=1}^3 \Lambda(w_1^j, v_t^j) = 587.31bits$$

$$L_2 = \Theta(|T|, \omega(\mathbf{w}_2, T)) + \sum_{j=1}^3 \Lambda(w_2^j, v_t^j) = 297.22bits$$

Based on our similarity measure, $\mathbf{w}_2$ is more similar to $\mathbf{v}_t$ than $\mathbf{w}_1$. That means, the source data set S_2 is more similar to the target task. Therefore, we incline to select more information from S_2 in the micro level evaluation.

3.2 Micro Level Evaluation

In the macro level evaluation, the source data sets are sorted in descending order on the similarities with the target data set T , with indexes re-assigned from 1 to K . Suppose we now intend to transfer S_1 in the source domain to T , as some of the data in S_1 may not be useful, or sometimes harmful to the results, we need to adaptively select some parts of the data in the micro level evaluation.

A source task S_i is divided into n_s parts ($1 \leq n_s \leq |S_i|$). Our objective is to transfer only useful parts to T based on the coding method. Note that, if n_s equals to 1, all the instances in S_i are transferred and if n_s equals to $|S_i|$, it coincides with the instance-based transfer methods which evaluate all instances in the data sets.

To divide the source data sets, a simple clustering method such as k -means can be adopted in our framework. When we select one related source task S_i , we first perform clustering on this data set, which generates n_s clusters, namely $S_i^1, S_i^2, \dots, S_i^{n_s}$. Each cluster is then regarded as a data set and is evaluated in the same way as the macro level by compact coding, with a code length assigned to evaluate the degree of similarity to the target task T .

3.3 The Transfer Learning Algorithm

We provide our CCHC algorithm as follows:

Algorithm CCHC

for $i = 1$ to K

calculate L_i for each S_i by (8), obtain L_{min}

sort S_i based on the ascending order of L_i

$TR = \emptyset$

for $j = 1$ to K

perform clustering on S_j , obtain S_j^t ($t = 1, \dots, n_s$)

calculate l_t for each S_j^t by (8)

sort S_j^t based on the ascending order of l_t

for $t = 1$ to n_s

```

 $TR = TR \cup S_j^t$  with the shortest  $l_t$ 
perform classification by SVM on  $TR$  and obtain  $\mathbf{w}'$ 
calculate  $L' = -\log P(\mathbf{w}'|\mathbf{v}_t) - \log P(T|\mathbf{w}')$ 
if  $L' < L_{min}$ 
     $L_{min} = L'$ 
     $\mathbf{v}_t = \mathbf{w}'$ 
     $S_j = S_j - S_j^t$ 
else break
 $\mathbf{w}_t = \mathbf{v}_t$ 
output  $\mathbf{w}_t$ 

```

where TR denotes the training set for the classification task. l_t represents the code length for each part of the source task in the micro level evaluation.

4 Experiments

4.1 Experimental Setting

We perform experiments on the data sets from the UCI repository¹ and the Text data sets. The three data sets in the UCI repository used in the experiments are *mushroom*, *splice* and *krvsnp*. We adopted a pre-processing method [22,26] on them to fit the transfer learning scenario. For the Text data sets, we choose 20NewsGroup data sets².

The *mushroom* data set has 8124 examples with 22 attributes in each example and one binary class label. The *splice* data set has 3190 examples with 60 attributes in each example and one binary class label. The *mushroom* data set is split into the source task and the target task based on the attribute *stalk-shape*, the source task contains examples whose *stalk-shape* are *tapering* and the examples in the target task have *stalk-shape* of *enlarging*. The *splice* data set is divided into two based on the first attribute. If the first attribute value of an example is “A” or “G”, it is added into the target task, otherwise it is added into the source task. [22] and [26] show that the splitting method could ensure different distributions between the source and the target tasks. The *krvsnp* data set has 3196 examples with 36 attributes in each example and one binary class label. Since it is not included in [26], we adopt the same strategy to split the data into two. The data set is divided based on the eleventh attribute as it shows a result similar to the *splice* data set. The source and the target tasks contain examples with the attribute value f and t , respectively.

The number of examples in the source task for the *mushroom* data, the *splice* data and the *krvsnp* data is set to be 1000 which is the same as in [26]. We investigated the influence of the number of instances in the target data set, and the noise level in the target data set. The noise is added by reversing the correct class labels of the examples in the training data sets.

¹ <http://archive.ics.uci.edu/ml/>

² <http://people.csail.mit.edu/jrennie/20Newsgroups>

We follow the splitting strategy on 20Newsgroups data sets as [22]. Three data sets are chosen which are *rec* vs *talk*, *rec* vs *sci* and *sci* vs *talk*. For example, in *rec* vs *talk* data set, all the positive instances are from the category *rec*, while negative ones are from the category *talk*. The instances in the source domain and the target domain are selected based on the subcategories. In the experiments, each of the target tasks in the three data sets are chosen as the single target task, and the training data in the three data sets are all chosen as the source tasks. In such a way, $\mathbb{S}$ contains 3 different tasks as S_1 , S_2 and S_3 and we test our algorithm in this transfer learning setting. Note that, 20Newsgroups data is regarded to be class-balanced. The challenging issues such as handling class-imbalanced data will be considered as one of our future works.

Our CCHC is compared with the COITL [26] and TrAdaBoost [22], which are two state-of-the-art instance-based methods in transfer learning, Active Transfer (AT) [23] which is a feature-based method, and k -NN with $k = 3$ as well as the basic SVM. We follow the values of the parameters in the original papers. The source data sets are merged into one data set for these algorithms. k -means clustering method was chosen as the clustering method in our micro level evaluation. After carefully testing the parameters, for *mushroom*, *splice* and *krvsnp* data sets, k is set to be 4 and for the remaining data sets, $k=2$. All the experiments are repeated ten times and we report the average results. The hyperplanes are obtained by C-SVC with polynomial kernel [5], which are considered effective to data with a large number of features and without class noise.

For the UCI data sets, each of *mushroom*, *splice* and *krvsnp* has one source task and one target task, CCHC is thus used only in the micro level to select useful parts. For the Text data sets, both the macro level and the micro level evaluations are examined. We mainly test two factors in the experiments. One is the different number of instances in the target task. We set $|T|$ equals to 50 and 100, respectively. The other is the noise level in the target task from 0% to 15%. For example, when $|T|=50$ and the noise level is 15%, there are only a small number of instances correctly labeled and this number can be regarded as “few”. We also check the number of source tasks and the number of clusters in each task that are transferred to the target task.

4.2 Experimental Results

Table 1, 2 and 3 provide the results on the *mushroom* data set, the *krvsnp* data set and the *splice* data set, respectively. As the general tendency, the error rates become larger with the noise level increases from 0% to 15%. Also, given more labeled instances in the target domain, such as $|T| = 100$, the results are obviously better. It can be observed from the tables that our method outperforms the other methods in most conditions. For the *mushroom* data set in Table 1, however, our method is outperformed by others in a small number of conditions such as in 15% noise level while $|T| = 50$. The possible reason is that, *mushroom* data sets are well organized and it is simple to find the underlying hypotheses even with a few labeled instances. The improvements by transferring information from the source domain to the target domain are not as promising as expected.

Table 1. Results on *mushroom* data set

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$ T =50$	SVM	0.087	0.12	0.146	0.173	0.19	0.207
	CCHC	0.079	0.106	0.129	0.127	0.156	0.171
	TrAdaBoost	0.158	0.159	0.173	0.191	0.168	0.195
	KNN	0.117	0.125	0.153	0.147	0.167	0.163
	COITL	0.132	0.144	0.146	0.161	0.168	0.159
	AT	0.156	0.196	0.177	0.16	0.185	0.142
$ T =100$	SVM	0.067	0.052	0.111	0.129	0.167	0.198
	CCHC	0.058	0.05	0.081	0.116	0.162	0.179
	TrAdaBoost	0.145	0.143	0.158	0.178	0.167	0.166
	KNN	0.081	0.084	0.104	0.12	0.147	0.159
	COITL	0.103	0.08	0.087	0.121	0.11	0.112
	AT	0.2	0.189	0.177	0.199	0.184	0.178

Table 2. Results on *krvsnp* data set

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$ T =50$	SVM	0.15	0.22	0.192	0.167	0.256	0.282
	CCHC	0.095	0.128	0.126	0.143	0.153	0.18
	TrAdaBoost	0.514	0.504	0.504	0.489	0.491	0.493
	KNN	0.25	0.277	0.328	0.303	0.297	0.311
	COITL	0.246	0.283	0.33	0.288	0.285	0.313
	AT	0.18	0.192	0.144	0.13	0.18	0.19
$ T =100$	SVM	0.11	0.099	0.171	0.166	0.243	0.207
	CCHC	0.066	0.085	0.115	0.132	0.167	0.169
	TrAdaBoost	0.509	0.505	0.496	0.484	0.497	0.478
	KNN	0.221	0.215	0.224	0.242	0.24	0.276
	COITL	0.24	0.219	0.229	0.24	0.233	0.276
	AT	0.11	0.133	0.122	0.135	0.127	0.138

In Table 2 of *krvsnp*, our method outperforms other methods in most conditions. In Table 3 of *splice*, our method is the best one among all the methods even with 15% noise. And the improvements are nearly 10% compared to those of the state-of-the-arts methods. From the experiments, we observed that the quality of $\mathbf{v}_t$ tends to fluctuate when the noise is high. In *splice* data sets with 60 attributes, this impact becomes less serious and CCHC could gain much help from the abundant information in the source tasks to improve the hyperplane in the target task.

We also evaluate the number of source tasks and the number of clusters in the source tasks that are transferred in Table 4, where S denotes the source task for the *mushroom*, *krvsnp* and *splice* data sets. The values in the table represent

Table 3. Results on *splice* data set

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$ T =50$	SVM	0.302	0.321	0.354	0.368	0.391	0.346
	CCHC	0.18	0.207	0.212	0.214	0.216	0.222
	TrAdaBoost	0.343	0.296	0.318	0.323	0.314	0.39
	KNN	0.375	0.354	0.347	0.379	0.398	0.415
	COITL	0.385	0.401	0.388	0.422	0.394	0.436
	AT	0.468	0.47	0.441	0.469	0.48	0.478
$ T =100$	SVM	0.232	0.23	0.276	0.293	0.309	0.322
	CCHC	0.18	0.182	0.184	0.199	0.209	0.203
	TrAdaBoost	0.234	0.3	0.302	0.267	0.294	0.299
	KNN	0.331	0.349	0.352	0.377	0.367	0.396
	COITL	0.319	0.339	0.327	0.362	0.367	0.374
	AT	0.472	0.458	0.474	0.484	0.465	0.477

Table 4. Transferred components of the source tasks in UCI data sets

			Percentage of noise on T					
			0%	3%	6%	9%	12%	15%
<i>mushroom</i>	$ T = 50$	S	2	2	1	1	2	1
	$ T = 100$	S	2	1	2	2	2	2
<i>krvsnp</i>	$ T = 50$	S	1	2	3	3	2	2
	$ T = 100$	S	1	2	3	2	2	3
<i>splice</i>	$ T = 50$	S	2	3	1	2	3	3
	$ T = 100$	S	2	2	2	3	3	3

the number of clusters in each source task that are used in the learning process. As shown in Table 4, in the two data sets *krvsnp* and *splice*, the components of the source tasks are transferred as much as possible. The reason is that in the two data sets, the number of attributes is more than that in the *mushroom* data set, and the labeled instances in the target training task are not adequate to obtain good hyperplanes. Therefore, our method tries to find relevant information from the source tasks as much as possible.

Table 5, Table 6 and Table 7 provide the results on *rec* vs *talk*, *rec* vs *sci*, and *sci* vs *talk*, respectively, with $|T|$ equals to 50 and 100. It can be seen from the three tables that our method is obviously better than other methods, even under noise conditions. However, CCHC is sometimes outperformed by COITL or TrAdaBoost in a small number of situations while $|T| = 100$. We ascribe the results to the reason that when the number of labeled instances increases, other methods also could find good hyperplanes. But under the circumstances when $|T| = 50$, our method is always the best one. It is a proof of the robustness of our method given only a few labeled instances in the target domain. We also

Table 5. Results on *rec* vs *talk* as the target task

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$ T =50$	SVM	0.154	0.174	0.189	0.228	0.227	0.23
	CCHC	0.122	0.144	0.167	0.196	0.21	0.223
	TrAdaBoost	0.236	0.234	0.275	0.309	0.32	0.321
	KNN	0.207	0.255	0.237	0.256	0.244	0.305
	COITL	0.206	0.255	0.229	0.25	0.241	0.294
	AT	0.472	0.364	0.38	0.388	0.488	0.49
$ T =100$	SVM	0.088	0.166	0.177	0.269	0.285	0.295
	CCHC	0.084	0.154	0.151	0.186	0.191	0.248
	TrAdaBoost	0.265	0.283	0.338	0.348	0.355	0.338
	KNN	0.23	0.248	0.267	0.277	0.281	0.283
	COITL	0.224	0.245	0.257	0.267	0.278	0.283
	AT	0.406	0.363	0.492	0.477	0.315	0.517

Table 6. Results on *rec* vs *sci* as the target task

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$ T =50$	SVM	0.163	0.177	0.187	0.181	0.223	0.24
	CCHC	0.159	0.167	0.179	0.18	0.221	0.227
	TrAdaBoost	0.266	0.289	0.292	0.353	0.351	0.375
	KNN	0.245	0.224	0.255	0.246	0.261	0.3
	COITL	0.243	0.236	0.259	0.255	0.268	0.311
	AT	0.41	0.37	0.44	0.325	0.354	0.419
$ T =100$	SVM	0.139	0.152	0.153	0.174	0.21	0.213
	CCHC	0.126	0.128	0.15	0.168	0.208	0.21
	TrAdaBoost	0.24	0.23	0.24	0.228	0.267	0.284
	KNN	0.216	0.185	0.192	0.182	0.212	0.231
	COITL	0.206	0.2	0.194	0.183	0.22	0.231
	AT	0.433	0.316	0.399	0.359	0.315	0.438

notice that, when the *sci* vs *talk* data set is used as the target task, the error rates for all the methods are slightly higher than that of the other two data sets. The possible reason is that the discrepancy of distributions between the source domain and the target domain is large.

In Table 8, we report the number of source tasks and the number of clusters in each source task used in the micro level stage, where S_1 , S_2 and S_3 denote the source task from *rec* vs *talk*, *rec* vs *sci* and *sci* vs *talk*, respectively. The integer values in the table represent the numbers of clusters in each source task that are used in the micro level. It can be observed that, our method is able to choose relevant parts from relevant tasks. For example, when *rec* vs *talk* is used as the target task, by examining the learning procedure of the algorithm,

Table 7. Results on *sci* vs *talk* as the target task

		Percentage of noise on T					
		0%	3%	6%	9%	12%	15%
$ T =50$	SVM	0.183	0.195	0.223	0.24	0.307	0.364
	CCHC	0.174	0.189	0.205	0.232	0.244	0.272
	TrAdaBoost	0.301	0.282	0.317	0.372	0.372	0.409
	KNN	0.285	0.305	0.301	0.327	0.307	0.385
	COITL	0.286	0.303	0.318	0.327	0.31	0.385
	AT	0.368	0.425	0.446	0.382	0.366	0.371
$ T =100$	SVM	0.221	0.235	0.239	0.264	0.329	0.342
	CCHC	0.167	0.174	0.172	0.21	0.232	0.254
	TrAdaBoost	0.195	0.219	0.255	0.244	0.213	0.23
	KNN	0.184	0.185	0.196	0.194	0.194	0.233
	COITL	0.172	0.184	0.188	0.188	0.192	0.234
	AT	0.352	0.399	0.323	0.327	0.418	0.362

Table 8. Transferred components of the source tasks in 20Newsgroup data sets

			Percentage of noise on T					
			0%	3%	6%	9%	12%	15%
<i>rec</i> vs <i>talk</i> as T	$ T = 50$	S_1	0	0	0	1	0	0
		S_2	1	1	1	1	1	0
		S_3	1	1	1	1	1	1
	$ T = 100$	S_1	1	0	0	0	0	0
		S_2	1	1	1	1	1	1
		S_3	1	1	1	1	1	1
<i>rec</i> vs <i>sci</i> as T	$ T = 50$	S_1	1	1	0	1	1	1
		S_2	0	0	0	0	0	0
		S_3	1	1	1	1	1	0
	$ T = 100$	S_1	1	1	1	1	1	1
		S_2	0	0	0	0	1	0
		S_3	0	0	1	1	0	1
<i>sci</i> vs <i>talk</i> as T	$ T = 50$	S_1	1	1	1	1	1	1
		S_2	0	0	0	0	0	0
		S_3	1	1	1	0	0	1
	$ T = 100$	S_1	1	1	0	1	1	1
		S_2	0	0	0	0	0	0
		S_3	1	1	1	1	1	1

we found that in the macro level, our method arrange the source tasks as S_3 , S_2 and S_1 , thus CCHC is more likely to choose the clusters in S_3 and S_2 , while the third source task is regarded as to bring negative effects on the results. It can be a proof of the effectiveness of our method that can adaptively choose relevant information from the source domain in order to avoid negative transfer.

5 Related Work

In this section, we review research works similar to ours including different methods for inductive transfer learning, and related works on negative transfer. Our CCHC belongs to the supervised inductive transfer learning where both the source and the target tasks contain labeled data [18,3,19]. Existing methods for inductive transfer learning may be classified into instance-based approaches [22,26] and feature-based approaches [1,21,23,24]. For instance-based methods, each instance in the source task is evaluated before being added to the target task to help the classification process in the target task. Both [22] and [26] adopted a re-weighting method to judge the impact of examples in the source task in the learning of the target task. The feature-based methods try to find a subspace spanned by relevant features as common knowledge to help to improve the classification results on the target task. However, most of them are not parameter-free and sensitive to noise. [1] proposed a method to learn a low-dimensional representation for multiple tasks. [24] provided a spectral-based solution to transfer the eigenspace where the process is adjusted by the KL divergence. [21] tried to find a kernel given a set of predefined kernels. Our method also belongs to the feature-based transfer. For feature selection, the MDLP based method was proposed in [17] to learn prior knowledge over the features.

To tackle with the problem of negative transfer, current works mainly focus on finding the similarities among tasks or instances such as [2,4,23,22,26,9]. [22] extended the AdaBoost algorithm which aims at improving the accuracy of a weak learner by adjusting the weights of the instances in the training sets. Their TrAdaBoost algorithm could evaluate the instances from a large amount of data in the source domain and assign weights based on the similarities to the target task in order to boost the accuracy of the classifier. [26] also proposed a semi-supervised learning method by extending the co-training method which deals with the same problem as [22]. Instances that can be put into the target task are obtained by re-weighting those in the source task. [2] tried to find a common representation among different groups of tasks which can be regarded as the transferred information. However, the method was restricted to linear classification functions and the instances in the target task are not considered to be much less than that in the source tasks. The basic Probabilistic Latent Semantic Analysis (PLSA) is extended in [9], in order to simultaneously capture both the domain distinctions and commonality among multiple domains. In [4], a new kernel function was designed by exploiting the Gaussian process to evaluate the negative similarity between two instances, but it is designed for only one single source task and without taking multiple source tasks into the framework. An active learning approach ERS [14] (Error Reduction Sampling) is integrated into the transfer learning [23] with a heuristic similarity function. It is pointed out by the author that experts are heavily relied on, as the possibility to query an expert is set to be higher than 50%. Moreover, the degree of reliability of the oracles is not fully taken into consideration. Our work relates to the feature-based transfer learning, and a two-level evaluation method was proposed, in which the similarities among both the tasks and the components of tasks are considered.

6 Conclusion

This paper proposed a compact coding algorithm CCHC for inductive transfer learning to solve the negative transfer problem. The coding scheme is inspired by MDLP and a two-level evaluation is adopted to adaptively select useful parts in the source domain, and meantime to neglect those parts which will result in negative effects from further consideration. By incorporating a simple classifier such as SVM, given a few labeled examples, our method performs well on the real data sets, and outperforms other state-of-the-art methods in most conditions.

Acknowledgment. This work was partially supported by the grant-in-aid for scientific research on fundamental research (B) 21300053 from the Japanese Ministry of Education, Culture, Sports, Science and Technology, and the Strategic International Cooperative Program funded by the Japan Science and Technology Agency (JST).

References

1. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-Task Feature Learning. In: Proc. 19th NIPS, pp. 41–48 (2007)
2. Argyriou, A., Maurer, A., Pontil, M.: An Algorithm for Transfer Learning in a Heterogeneous Environment. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 71–85. Springer, Heidelberg (2008)
3. Bakker, B., Heskes, T.: Task Clustering and Gating for Bayesian Multitask Learning. *Journal of Machine Learning Research* 4, 83–99 (2003)
4. Cao, B., Pan, S.J., Yang, Q.: Adaptive Transfer Learning. In: AAAI 2010 (2010)
5. Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines(2001), <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
6. Shannon, C.: A Mathematical Theory of Communication. *Bell System Technical Journal* 27, 379–423, 623–656 (1948)
7. Wallace, C., Patrick, J.: Coding Decision Trees. *Machine Learning* 11(1), 7–22 (1993)
8. Keogh, E., Lonardi, S., Ratanamahatana, C.A.: Towards Parameter-Free Data Mining. In: KDD 2004, pp. 206–215 (2004)
9. Zhuang, F.Z., Luo, P., Shen, Z.Y., He, Q., Xiong, Y.H., Shi, Z.Z., Xiong, H.: Collaborative Dual-PLSA: Mining Distinction and Commonality across Multiple Domains for Text Classification. In: CIKM 2010, Toronto, Canada (October 2010)
10. Shao, H., Suzuki, E.: Feature-based Inductive Transfer Learning through Minimum Encoding. In: SDM 2011, Phoenix/Mesa, Arizona (April 2011)
11. Quinlan, J.R., Rivest, R.L.: Inferring Decision Trees Using the Minimum Description Length Principle. *Information and Computation* 80(3), 227–248 (1989)
12. Borgwardt, K.M., Gretton, A.: Integrating Structured Biological Data by Kernel Maximum Mean Discrepancy. *Bioninformatics*, 1–9 (2006)
13. Rosenstein, M.T., Marx, Z., Kaelbling, L.P.: To Transfer or Not To Transfer. In: NIPS 2005 Workshop on Transfer Learning (2005)
14. Roy, N., McCallum, A.: Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In: ICML 2001, pp. 441–448 (2001)

15. Thach, N.H., Shao, H., Tong, B., Suzuki, E.: A Compression-based Dissimilarity Measure for Multi-task Clustering. In: Proc. Nineteenth International Symposium on Methodologies for Intelligent Systems, Warsaw (June 2011)
16. Grünwald, P.D.: The Minimum Description Length Principle. MIT Press, Cambridge (2007)
17. Dhillon, P.S., Ungar, L.: Transfer Learning, Feature Selection and Word Sense Disambiguation. In: NLP/CL ACL-IJCNLP, Singapore (August 2009)
18. Caruana, R.: Multitask Learning. *Machine Learning* 28(1), 41–75 (1997)
19. David, S.B., Schuller, R.: Exploiting Task Relatedness for Multiple Task Learning. In: COLT 2003, pp. 825–830 (2003)
20. Pan, S.J., Yang, Q.: A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10), 1345–1359 (2008)
21. Rückert, U., Kramer, S.: Kernel-based Inductive Transfer. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 220–233. Springer, Heidelberg (2008)
22. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for Transfer Learning. In: ICML 2007, pp. 193–200 (2007)
23. Shi, X., Fan, W., Ren, J.: Actively Transfer Domain Knowledge. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 342–357. Springer, Heidelberg (2008)
24. Shi, X., Fan, W., Yang, Q., Ren, J.: Relaxed Transfer of Different Classes via Spectral Partition. In: ECML/PKDD, vol. (2), pp. 366–381 (2009)
25. Ke, Y.: Inferring Informed Clustering Problems with Minimum Description Length principle. Doctoral dissertation, State University of New York at Albany (2007)
26. Shi, Y., Lan, Z.Z., Liu, W., Bi, W.: Extended Semi-supervised Learning Methods for Inductive Transfer Learning. In: ICDM 2009, pp. 483–492 (2009)