Gaussian Process for Transfer Learning through Minimum Encoding*

Hao Shao[1], Rui Xu[2], and Feng Tao[3]

[1] School of WTO Research and Education,
Shanghai University of International Business and Economics
[2] School of Computer Science and Technology,
University of Science and Technology of China
[3] School of Business, East China University of Science and Technology

Abstract. In real applications, labeled instances are often deficient which makes the classification problem on the target task difficult. To solve this problem, transfer learning techniques are introduced to make use of existing knowledge from the source data sets to the target data set. However, due to the discrepancy of distributions between tasks, directly transferring knowledge will possibly lead to degenerated performance which is also called *negative trasnfer*. In this paper, we adopted the Gaussian process to alleviate this problem by directly evaluating the distribution differences, with the parameter-free Minimum Description Length Principle (MDLP) for encoding. The proposed method inherits the good property of solid theoretical foundation as well as noise-tolerance. Extensive experiments results show the effectiveness of our method.

Keywords: Transfer Learning, Gaussian Process, MDLP.

1 Introduction

Transfer learning [13] provides a solution for real applications to induce models on a target task given a large amount of auxiliary data. Take the four universities example into consideration, given a set of data sets of different universities and colleges, we can find useful information from them to help further classification on a data set of a new university (or a college) into several categories such as students, faculties, stuffs and so on. Assume that data sets of two universities, denoted by U_1, U_2, and two colleges, denoted by C_1, C_2, are in the source domain, we try to classify instance in a new college, say C_t, with a few labeled data.

Directly applying the latent functions or hypothesis drawn from the source domain generally may not lead to satisfactory performance on the target domain,

* This work was supported by Humanity and Social Science Youth foundation of Ministry of Education of China (No. 13YJC630126), the Fundamental Research Funds for the Central Universities (No.WK0110000032), the NSFC (No.71171184/71201059/71201151), the Funds for Creative Research Group of China (No. 70821001) and the NSFC major program (No.71090401/71090400).

H. Yin et al. (Eds.): IDEAL 2013, LNCS 8206, pp. 384–391, 2013.

which is regarded as negative transfer. Some of the information from the source domain may probably hinder the performance if the distributions between the source domain and the target domain are too dissimilar [10]. Although there exists extensive research works on transfer learning, as stated in the survey paper [13], how to avoid negative transfer is an important issue but yet not be studied intensively. In the example, if we directly use the hypothesi on U_1 to C_t, the error rate may be very high due to the different distributions underlying the two data sets. To avoid negative transfer, it is reasonable to consider the similarities between a source task and a target task, and several works have been done to find the similarities between tasks [1,12,2,3].

In multi-task learning, the similarities between tasks can be evaluated by distances between instances or the similarity between distributions. It is reasonable because all the tasks are treated as symmetric. There are enough instances in each task for us to find the underlying distributions. However, difficulties are encountered when we try to deal with transfer learning. In transfer learning setting, instances in the target task is often inadequate, generally it is difficult to find the distribution. Methods to measure the similarities for multi-task learning can not be directly adopted and the negative transfer tends to happen. Furthermore, labeling instances in the target task is expensive and time consuming.

In this paper, we tackle this problem from a two-level point of view, and try to directly solve the negative transfer problem by adopting the Gaussian process. Our main idea to calculate the task-level similarity is based on the purpose to use only auxiliary information in the source domain. The intuition is that it is difficult to find the ground-truth distribution on the target task, so we do not try to perform operations on the target data but only to use the labeled data for testing the "hypotheses". Therefore, an MDLP (Minimum Description Length Principle) [11] based method is adopted, because it can evaluate models on data sets and also parameter-free and robust to noise. For the instance-level similarity, we design a kernel functions by using the obtained information from the task-level similarity. We call our algorithm the GPTL (Gaussian Process for Transfer Learning).

2 Problem Setting and Preliminaries

2.1 Gaussian Process for Classification

A Gaussian process is a stochastic process and a collection of random variables. All these variables in a finite number have joint Gaussian distributions. We can interpret in that, the Gaussian distribution is over vectors, and the Gaussian process is over functions. We have the following definition that, a function f is distributed as a Gaussian process with mean function m and covariance function k, which is also denoted as a "kernel".

$$f \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right) \tag{1}$$

Due to the space limit, we will introduce GP in more detail in the extension of this manuscript.

2.2 Preliminaries for Encoding

A data set D consists of n examples $d_1, d_2, \ldots, d_n$. Each example d_i is described with attributes $a_1, \ldots, a_m$ as an attribute value vector $(a_{i1}, a_{i2}, \ldots, a_{im})$ of length m and belongs to one of M classes, of which labels are represented by $c_1, c_2, \ldots, c_M$. A classifier h is a function which outputs a class label given an attribute value vector. We call the process of learning a classifier from a data set classification. A classifier with a simple interpretation and a high classification accuracy is most preferable.

The MDLP 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 [4,9]. As a principle for preferring a classifier in classification, it is stated in MDLP that the best classifier h_{best} among the ones h that can be learned from D is given as follows.

$$h_{\text{best}} = \arg\min_{h} \left(-\log P(h) - \log P(D|h)\right) \tag{2}$$

Due to the space limit, we neglect the detailed explanation of the theoretical background of GPTL, readers can refer to [5,7,6] or the extended version of this manuscript. We provide the coding methods here considering encoding a binary string of length a which consists of b binary 1s and $(a-b)$ binary 0s [4,9]. In the sender and the receiver problem, the receiver is assumed to know the length a. 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 $\log \binom{a}{b}$ [4,9]. We hereafter call this method a binary coefficient method and denote the required code length by $\Theta(a, b)$ as follows.

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

Now we consider a problem of sending an integer a under the assumption that $a = b$ is most likely and the occurrence probability $P(i)$ of $a = i$ is given by $P(b)\phi^{|b-i|}$, where ϕ is a constant given by the user.

We denote the length $-\log\left[P(b)\phi^{|b-a|}\right]$ required to send a given b by $\Lambda(a, b)$ and it can be easily obtained as

$$\Lambda(a, b) = \log\left[3 - \left(\frac{1}{2}\right)^{b}\right] + |b-a| \tag{3}$$

3 The GPTL Algorithm

3.1 Arrange Related Tasks

For the source tasks $S_1, S_2, ..., S_{\text{k}}$, we can easily obtain the latent function $\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_k$ underlying the source data sets based on the Gaussian process. We

denote the $\mathbf{f}_i$ as the "hypothesis" in the MDLP. As it is difficult to find the true distribution underlying the target data set given inadequate instances, our idea is to use existing latent functions and fit each $\mathbf{f}_i$ to the target data set to evaluate the similarities. MDLP has advantages to do such jobs.

In order to find the source data set which has the most similar distributions with the target data set, we need to find the one in $\mathbf{f}_i$ ($i = 1$ to k) which can maximize the probability $P(\mathbf{f}_i|D_t)$. It turns out to be the maximization of:

$$\arg\max_{\mathbf{f}_i} \frac{P(D_t|\mathbf{f}_i)P(\mathbf{f}_i)}{P(D_t)} \tag{4}$$

In the evaluation procedure, $P(D_t)$ is neglected as a constant. By negative encoding with MDLP, this could be calculated as

$$L_i \equiv -\log P(D_t|\mathbf{f}_i) - \log P(\mathbf{f}_i) = -\log \prod_i P(d_i^t|\mathbf{f}_i) - \log P(\mathbf{f}_i) \tag{5}$$

where d_i^t represents the i-th instance in D_t.

This could be regarded as the sender and receiver problem, and evaluated by MDLP. Then the key issue is to design a delicate coding method. Note that, a simple way to encode the first part of formula (2) is to specify the wrong predictions of the class labels in T based on $\mathbf{f}_i$. The second part can be regarded as the complexity of the latent function $\mathbf{f}_i$. This also coincides with the basic format of the refined MDL with one part as the stochastic complexity of the data related to the model and the other part as the parametric complexity.

To encode $-\log P(\mathbf{f}_i)$, since each $\mathbf{f}_i$ is assumed to follow a Gaussian distribution as $\mathbf{f}_i \sim \mathcal{N}(0, K)$, we only need to calculate the code length of the kernel matrix K (K is a matrix with each entry K_{ij} as a kernel function $k(x_i, x_j)$). Commonly, the squared-exponential covariance function is adopted as a kernel in Gaussian process:

$$k(x_i, x_j) = \sigma_f^2 \exp(-\frac{1}{2l^2}(x_i - x_j)^2) \tag{6}$$

Then we only need to code the two parameters σ_f and l. A direct way is to code the two real values using $\Lambda(\sigma_f, 0)$ and $\Lambda(l, 0)$. Notice that, even with limited information from labeled data in the target task, we could obtain roughly a function $\mathbf{f}_t$ as similar to the true distribution of T. Let the corresponding parameters in the kernel function of $\mathbf{f}_t$ be as σ'_f and l'. Therefore, we can use $\Lambda(\sigma_f, \sigma'_f)$ and $\Lambda(l, l')$ to send the two values under the assumption that $\sigma_f = \sigma'_f$ and $l = l'$ are most likely. Then the code length should be:

$$-\log P(\mathbf{f}_i) = \Lambda(\sigma_f, \sigma'_f) + \Lambda(l, l') \tag{7}$$

Then we obtain the code length L_i between each S_i and T. The source task with the shortest code length is regarded to "best fit" the target data, thus the

distribution is more similar to the one in the target task. Then the source data sets are arranged in ascending order on the code length with the target data set T.

3.2 The Instance Level Similarities

To represent the similarity between two instances from different tasks, we are motivated to construct a kernel function by using the code length which is regarded as the similarity measurements between tasks. Based on an existed kernel $k(x_i, x_j)$, we design an extended one $\tilde{k}(x_i, x_j)$ as follows:

$$\tilde{k}(x_i, x_j) = exp(\frac{1}{1 + exp(-L_i)})k(x_i, x_j) \tag{8}$$

It is easy to prove that $\tilde{k}(x_i, x_j)$ is also a kernel if $exp(\frac{1}{1+\exp(-L_i)})$ is regarded as a positive constant.

The intuition behind formula (8) is that, the kernel function should take both the task similarity and instance similarity together. Note that, when instances are from the same task, the coefficient $exp(\frac{1}{1+\exp(-L_i)})$ is set to be 1 and thus the kernel function takes the original format.

4 Experiments

We perform experiments on the data sets from the UCI repository[1] and the Text data sets. The three data sets in the UCI repository used in the experiments are *mushroom* and *splice*. We adopted a pre-processing method [14,16] on them to fit the transfer learning scenario. For the Text data sets, we choose 20NewsGroup data sets[2].

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. We adopt the same strategy in [14] and [16] to split each data set into two.

The number of examples in the source task is set to be 1000 which is the same as in [16]. 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.

We follow the splitting strategy on 20Newsgroups data sets as [14]. 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

[1] `http://archive.ics.uci.edu/ml/`
[2] `http://people.csail.mit.edu/jrennie/20Newsgroups`

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.

Our GPTL is compared with the COITL [16] and TrAdaBoost [14], Active Transfer (AT) [15] and k-NN with $k = 3$ as well as the basic SVM. The hyperplanes are obtained by C-SVC with polynomial kernel, which are considered effective to data with a large number of features and without class noise.

For the UCI data sets, each of them has one source task and one target task, GPTL is thus used only in the instant level to select useful parts. For the Text data sets, both the two levels 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%.

Table 1 and 2 provide the results on the *mushroom* data set and the *splice* data set, respectively. Generally speaking, the error rates go up with the noise level increases. If the target domain has more labeled instances, for example, $|T| = 100$, the accuracy is obviously better. From Table 1 we observed that GPTL is better than other methods in most circumstances. However, because the *mushroom* data set is well organized, for all the methods, it is easier to find the underlying hypotheses. That is why in some cases our algorithm is outperformed by the state-of-the-art methods. In Table 2, in this larger and more complex data set, our method is the best among all the methods, and the improvements are much larger than that in the mushroom data set.

Table 1. Results on *mushrom* 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
	GPTL	**0.082**	**0.11**	**0.132**	**0.147**	**0.163**	0.165
	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
	GPTL	**0.061**	0.074	**0.079**	**0.103**	**0.105**	0.136
	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 3, Table 4 and Table 5 provide the results on *rec* vs *talk*, *rec* vs *sci*, and *sci* vs *talk*, respectively, with $|T|$ equals to 50 and 100. From these table, it can be seen that our method is obviously better than other methods, even under noise conditions. It is a proof of the robustness of our method given only a few labeled instances in the target domain. We also 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.

Table 2. 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
	GPTL	**0.2**	**0.211**	**0.235**	**0.243**	**0.296**	**0.314**
	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
	GPTL	**0.19**	**0.207**	**0.232**	**0.23**	**0.249**	**0.255**
	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 3. 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
	GPTL	**0.135**	**0.138**	**0.165**	**0.185**	**0.203**	**0.219**
	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
	GPTL	0.105	**0.144**	**0.16**	**0.179**	**0.24**	**0.258**
	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 4. 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
	GPTL	**0.133**	**0.145**	**0.152**	**0.176**	**0.215**	**0.21**
	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
	GPTL	**0.136**	**0.142**	**0.15**	**0.169**	**0.19**	**0.202**
	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

Table 5. 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
	GPTL	**0.175**	**0.191**	**0.203**	**0.22**	**0.25**	**0.256**
	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
	GPTL	**0.165**	**0.173**	**0.18**	0.196	0.21	**0.215**
	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

5 Conclusion

This paper proposed a transfer learning algorithm based on Gaussian process. By directly evaluating the distribution differences between tasks, we alleviate the problem of negative transfer. MDLP was also adopted to balance both the simplicity of the hypothesis and the goodness-of-fit to the data. In the experiments, our method was proved to outperform other methods and also be robust to noise.

References

1. 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)
2. Bakker, B., Heskes, T.: Task clustering and gating for bayesian multitask learning. Journal of Machine Learning Research 4, 83–99 (2003)
3. Cao, B., Pan, S.J., Yang, Q.: Adaptive Transfer Learning. In: AAAI 2010 (2010)
4. Wallace, C., Patrick, J.: Coding Decision Trees. Machine Learning 11(1), 7–22 (1993)
5. Shao, H., Tong, B., Suzuki, E.: Compact Coding for Hyperplane Classifiers in Heterogeneous Environment. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS, vol. 6913, pp. 207–222. Springer, Heidelberg (2011)
6. Shao, H., Tong, B., Suzuki, E.: Extended MDL Principle for Feature-based Inductive Transfer Learning. Knowledge and Information Systems 35(2), 365–389 (2013)
7. Shao, H., Suzuki, E.: Feature-based Inductive Transfer Learning through Minimum Encoding. In: SDM 2011, pp. 259–270 (2011)
8. Shao, H., Tong, B., Suzuki, E.: Query by Committee in a Heterogeneous Environment. In: Zhou, S., Zhang, S., Karypis, G. (eds.) ADMA 2012. LNCS, vol. 7713, pp. 186–198. Springer, Heidelberg (2012)
9. Quinlan, J.R., Rivest, R.L.: Inferring Decision Trees Using the Minimum Description Length Principle. Information and Computation 80(3), 227–248 (1989)
10. Rosenstein, M.T., Marx, Z., Kaelbling, L.P.: To Transfer or Not To Transfer. In: NIPS 2005 Workshop on Transfer Learning (2005)
11. Grünwald, P.D.: The Minimum Description Length Principle. MIT Press, Cambridge (2007)
12. Ben-David, S., Schuller, R.: Exploiting task relatedness for multiple task learning. In: Schölkopf, B., Warmuth, M.K. (eds.) COLT/Kernel 2003. LNCS (LNAI), vol. 2777, pp. 567–580. Springer, Heidelberg (2003)
13. Pan, S.J., Yang, Q.: A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (2009)
14. Dai, W., Yang, Q., Xue, G., Yu, Y.: Boosting for Transfer Learning. In: ICML 2007, pp. 193–200 (2007)
15. 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)
16. Shi, Y., Lan, Z., Liu, W., Bi, W.: Extended Semi-supervised Learning Methods for Inductive Transfer Learning. In: ICDM 2009, pp. 483–492 (2009)