Using Unlabeled Data to Improve Inductive Models by Incorporating Transductive Models

ShengJun Cheng
School of Computer Science and Technology
Harbin Institute of Technology
Harbin 150001, China
Email: http://hitwer@gmail.com

Jiafeng Liu
Harbin Institute of Technology
Harbin150001, China

XiangLong Tang
Harbin Institute of Technology
Harbin150001, China

Abstract—**This paper shows how to use labeled and unlabeled data to improve inductive models with the help of transductive models. We proposed a solution for the self-training scenario. Self-training is an effective semi-supervised wrapper method which can generalize any type of supervised inductive model to the semi-supervised settings. it iteratively refines a inductive model by bootstrap from unlabeled data. Standard self-training uses the classifier model(trained on labeled examples) to label and select candidates from the unlabeled training set, which may be problematic since the initial classifier may not be able to provide highly confident predictions as labeled training data is always rare. As a result, it could always suffer from introducing too much wrongly labeled candidates to the labeled training set, which may severely degrades performance. To tackle this problem, we propose a novel self-training style algorithm which incorporate a graph-based transductive model in the self-labeling process. Unlike standard self-training, our algorithm utilizes labeled and unlabeled data as a whole to label and select unlabeled examples for training set augmentation. A robust transductive model based on graph markov random walk is proposed, which exploits manifold assumption to output reliable predictions on unlabeled data using noisy labeled examples. The proposed algorithm can greatly minimize the risk of performance degradation due to accumulated noise in the training set. Experiments show that the proposed algorithm can effectively utilize unlabeled data to improve classification performance.**

Keywords—Inductive model, Transductive model, Semi-supervised learning, Markov random walk.

I. INTRODUCTION

Traditional inductive models like Naive Bayes, CARTs[1], Support Vector Machines are always in supervised settings, which means these model can only be trained on labeled data. Training a good inductive model needs enough labeled examples. Unfortunately, preparing labeled data for such task is often expensive and time consuming, while unlabeled data are readily available. This was the major motivation that led to the arise of semi-supervised paradigm which utilizes few labeled examples and vast amounts of cheap unlabeled examples to learns a model. Semi-supervised learning has achieved considerable success in a wide variety of domains, existing semi-supervised learning methods can be roughly categorized into several paradigms[2], including generative models, semi-supervised support vector machines (S3VMs), graph based methods and bootstrapping wrapper method.

Self-training[3] is a simple and effective semi-supervised algorithm which has been successfully applied to various real-world tasks. It is an wrapper method, which means it can generalize any type of supervised inductive model to the semi-supervised settings[4]. Self-training initially trains a classifier on labeled data and then iteratively augments its labeled training set by adding several newly pseudo-labeled unlabeled examples with most confident predictions of its own. Standard self-training uses the classifier model(trained on labeled examples) to label and select candidates from the unlabeled training set, which may be problematic since the initial classifier may not be able to provide highly confident predictions as labeled training data is always rare. In addition, since self-training utilizes unlabeled data in an incremental manner, early noise introduced to training sets would be reinforced round by round, resulting in severe performance degradation. Although some techniques, e.g. data editing[5], have been employed to alleviate this noise-related problem[6], results are yet undesirable. As a result, it could always suffer from introducing too much wrongly labeled candidates to the labeled training set, which may severely degrades performance. Another drawback of self-training is that the newly added examples are not informative to the current classifier, since they can be classified confidently[7]. As a result, they may only help increase the classification margin, without actually providing any novel information to the current classifier.

In this paper, we show how to use unlabeled data to improve inductive models with the help of transductive models. We proposed a solution for the self-training scenario, a novel self-training style algorithm is proposed. Generally, unlike traditional self-training only using labeled data to label and select unlabeled example for training set augmentation, our algorithm utilizes both labeled and unlabeled data to facilitate the self-labeling process. In detail, all the labeled and unlabeled examples are presented as a graph, where a novel markov random walk with constrains is proposed to label all examples on graph in a transductive setting[8]. This graph-based method satisfy *manifold assumption*that examples with high similarities in the input space should share similar labels. Typically, Most graph based methods output label information to unlabeled data in a transductive setting such as Label propagation, markov random walks, Low density separation[9]. Those methods are designed to utilize unlabeled by representing both the data as a graph, with examples as vertices and similarities of examples as edges. Existing transductive graph-based methods assume all labels on labeled data correct, can not work under training sets subject to noise. While our transductive model can naturally deals with noisy labeled data, which utilize ”label smooth" to automatically adjust the potential wrong labels. By incorporating this transductive model to the self-training process, we expect any applied supervised inductive model can be greatly improved.

The main contribution of this paper can be summarized as follows:

- We show that incorporating transductive models to inductive models in semi-supervised settings can improve classification performance.
- We propose a novel self-training algorithm which utilizes a graph-based transductive model for using both labeled and unlabeled data to label and select unlabeled example for training set augmentation.
- We propose a novel transductive model based on graph random walk with constrains. This transductive model can deal with labeled training set with noise and provide more reliable predictions for all unlabeled examples, has strong tolerance to noise in the training set.
- We conduct extensive experiments on several UCI benchmark data sets to evaluate its performance with 3 different inductive model and empirically demonstrate that our algorithm can effectively exploit unlabeled data to achieve better generalization performance.

The rest of this paper is organized as follows. Section 2 describes the problem and gives the algorithm in detail. Section 3 presents the experimental results on UCI data sets when various inductive models are utilized. A short conclusion and future work are presented in Section 4.

mds

January 11, 2007

II. The Proposed Algorithm

A. Problem Description and Notation

Let L denote the labeled training set with size $|L|$ and U denote the unlabeled training set with size $|U|$. The goal of our algorithm is to learn a classifier from $L \cup U$ to classify unseen examples. Generally, initial labeled examples are quite few, i.e. $|L| \ll |U|$.

The proposed algorithm learns a inductive model f from labeled and unlabeled data as follows: 1)initialize the model f using labeled set L; 2)use f to predict labels on unlabeled set U; 3)select a subset S from U for which f has the most confident predictions; 4)construct a neighborhood graph G with $L \cup U$ under certain similarity measure; 5)incorporate a transductive model into the self-labeling process: knowing the prior information about labels on $L \cup U$, start a constrained random walk on Graph G to label all the unlabeled examples in U; 6)choose k most confident examples from U for labeled training set augmentation according to the output of random walk; 7)refine f with augmented labeled data. The procedure goes on until there are no unlabeled examples left.

The key distinction between the proposed algorithm and standard self-training is the incorporated transductive model that utilizes both labeled and unlabeled examples to give prediction on unlabeled data. Most graph based semi-supervised methods are transductive, which are nonparametric and can deal with multi-classification problems. We proposed a novel constrained markov random walk for the transduction purpose. The most desirable property of the proposed transdutive model is that it can work well even if training set contains label-noise. Therefore, it is perfectly suitable for the self-training process, as the pseudo-labeled set S may contain some wrongly labeled examples. At this step, it is expected to yield more reliable predictions on unlabeled data than the classifier does with training set subject to label-noise. Next, we will present the details of the proposed transductive graph-based model.

B. Markov Random Walk with Constrains

Markov random walk is regarded as a transductive graph based approach which exploits manifold assumption to label all the unlabeled examples. Typically, it is given an undirected graph $G = (V, E, W)$, where a node $v \in V$ corresponds to an example in $L \cup U$, an edge $e = (a, b) \in V \times V$ indicates that the label of the two vertices a,b should be similar and the weight W_{ab} reflects the strength of this similarity. In this paper, graph is constructed by using the k nearest neighbor criterion. For each example $v \in L \cup U$, Let $\mathcal{C} = \{1, , m\}$ be the set of possible labels. Two row-vectors $\mathbf{Y}_v, \hat{\mathbf{Y}}_v$ are presented. The first vector $\mathbf{Y}_v$ is the input. The lth element of the vector $\mathbf{Y}_v$ encodes the prior knowledge about label l for example v. For instance, a labeled example v with label c has $\mathbf{Y}_{vc}$ set to 1, and the remaining $m-1$ elements of $\mathbf{Y}_v$ set to 0. Unlabeled examples have all their elements set to 0, that is $\mathbf{Y}_{vl} = 0$ for $l = 1...m$. The second vector $\hat{\mathbf{Y}}_v$ is the output of the algorithm, using similar semantics as $\mathbf{Y}_v$. For instance, a high value of $\hat{\mathbf{Y}}_{vl}$ indicates that algorithms believe that the vertex(example) v should have label l.

The constrains of random walks is formalized via three possible actions: inject, continue and abandon(denoted by $inj, cont, abnd$ with pre-defined probabilities $p_v^{inj}, p_v^{cont}, p_v^{abnd}$. Clearly, their sum is unit: $p_v^{inj} + p_v^{cont} + p_v^{abnd} = 1$. To label any example v(either labeled or unlabeled), we initiate a random-walk starting at v facing three options: with probability p_v^{inj} the random-walk stops and return(i.e. $inject$) the pre-defined vector information $\mathbf{Y}_v$. We constrain p_v^{inj} for unlabeled examples. Second, with probability p_v^{abnd} the random-walk abandons the labeling process and returns the all-zeros vector $\mathbf{0}_m$. Third, with probability p_v^{cont} the random-walk continues to one of vs neighbors v' with probability proportional to $W_{v'v}$. Note that by definition $W_{v'v} = 0$ if $(v', v) \notin E$. We summarize the above process with the following set of equations. The transition probabilities are,

$$\Pr[v'|v] = \begin{cases} \dfrac{W_{v'v}}{\sum\limits_{u:(u,v)\in E} W_{uv}}, (v', v) \in E \\ 0 \text{ , otherwise} \end{cases} \tag{1}$$

The expected score $\hat{\mathbf{Y}}_v$ for node $v \in V$ is given by,

$$\hat{\mathbf{Y}}_v = p_v^{inj} \times \mathbf{Y}_v + p_v^{cont} \times \sum_{v':(v',v)\in E} \Pr[v'|v] \hat{\mathbf{Y}}_{v'} + p_v^{abnd} \times \mathbf{0}_m \tag{2}$$

In this paper, the three probabilities $p_v^{inj}, p_v^{cont}, p_v^{abnd}$ are set using the same heuristics adapted from [10], which are defined by,

$$p_v^{cont} = \frac{c_v}{z_v} \quad ; \quad p_v^{inj} = \frac{d_v}{z_v} \quad ; \quad p_v^{abnd} = 1 - p_v^{cont} - p_v^{inj} \tag{3}$$

$c_v$is monotonically decreasing with the number of neighbors for node v in graph G. Intuitively, the higher the value of c_v, the lower the number of neighbors of vertex v and higher the information they contain about the labeling of v. The other quantity $d_v$is monotonically increasing with the entropy (for labeled vertices). It is noteworthy that abandonment occurs only when the continuation and injection probabilities are low enough. This is most likely to happen at unlabeled nodes with high degree. In effect, high p_v^{abnd} prevents the algorithm from propagating information through high degree nodes.

The final labeling information for all $v \in L \cup U$ can be computed through random walk based on Eq.(2). The algorithm converges when label distribution on each node ceases to change. Note that initial labeled data set L assumes to be noise-free, while the pseudo-labeled dataset S may contain classification noise, hence, certain modification about the transition probabilities needs to be made:

- Since labels on L, which are considered noise-free, should not change during the random walk. For example $v \in L$, the transition probabilities should be fixed as follows: $p_v^{inj} = 1$, $p_v^{cont} = 0$, $p_v^{abnd} = 0$;
- Since examples in S may be wrongly labeled by the classifier, labels on S are allowed to change. For $\forall v \in S$, the transition probabilities should be computed according to Eq.(3);
- For unlabeled example $u \in U - S$, we only constrain $p_u^{inj} = 0$.

Note that the predicted label y_u and labeling confidence CF(u, y_u) of each example $u \in U - S$ can be easily obtained from $\mathbf{Y}_u$:

$$y_u = \arg\max_l \hat{\mathbf{Y}}_{ul} \ , \ l = 1, ...m \tag{4}$$

$$\text{CF}(u, y_u) = \hat{\mathbf{Y}}_{uc} \ , c = y_u \tag{5}$$

In this paper, our strategy is to incorporates such transductive model into the standard self-training's labeling process, concrete procedures of the proposed algorithm is outlined in **Algorithm 1.** It is noteworthy that size of S only has mediate and minor impact on the final performance. For convenience, $|S|$ is empirically set equal to the number of initial labeled examples,i.e. $|L|$, and we also set k equal to $|L|$, The maximum iteration number M is set to 50.

Algorithm 1 Proposed Algorithm

```
Input:
    -$L \cup U$: training sets
    -$Learner$: learning algorithm for inducing a classifier
    -$M$: number of iteration
Output:
    -$f$: the returned classifier
    Construct neighborhood graph $G = (V, E, W)$;
 2: initialize $p_v^{inj}, p_v^{cont}, p_v^{abnd}$;
    $L' \leftarrow NULL$;
 4: while $Iter \leq M$ do
        Use $f$ to make predictions on $U$;
 6:     Select S from U { $|S|$ most confident predictions of $f$};
        Recompute $p_v^{inj}, p_v^{cont}, p_v^{abnd}$ for $v \in S$ using Eq.(3);
 8:     Reset prior labeling knowledge $\mathbf{Y}_v$;
        Output $\hat{\mathbf{Y}}_u$ for all $u \in U - S$ by constrained random
        walk on Graph G;
10:     Compute $y_u$, CF$(u, y_u)$ for all $u \in U$ using Eq.(4)and
        Eq.(5)
        Choose the k most confident examples from $U$ based
        on CF$(u, y_u)$;
12:     Add the chosen pseudo-labeled examples to $L'$;
        $f \leftarrow Learn(L \cup L')$
14: end while
```

TABLE I: Data set summary

Data set	Attribute	Size	Class	Class distribution(%)
australian	14	690	2	55.5/44.5
bupa	6	345	2	42.0/58.0
colic	22	368	2	63.0/37.0
diabetes	8	768	2	65.1/34.9
german	20	1000	2	70.0/30.0
hypothyroid	25	3163	2	4.8/95.2
ionosphere	34	351	2	35.9/64.1
kr-vs-kp	36	3196	2	52.2/47.8
sick	29	3772	2	6.1/93.9
tic-tac-toe	9	958	2	65.3/34.7
vehicle	18	846	4	25.1/25.7/25.7/23.5
wdbc	13	178	3	33.1/39.9/27.0

III. Experiments and Discussion

In this section, we design experiments to verify the efficacy of our algorithm. We mainly focus on the self-training framework, trying to find out how transductive model can improve the semi-supervised inductive model. 12 UCI data sets are used in the experiments[11]. Information on these data sets is shown in Table 1. For each data set, about 25% data are kept as test examples. 10% of the remaining data set is used as the labeled training set L; the rest examples are treated as the unlabeled set U.

The proposed algorithm is compared with standard self-training and SETRED[6]. SETRED is an improved self-training algorithm by incorporating data editing techniques to help identify and remove wrong labels from the training sets during the self-training process. For fair comparison, the termination criteria used by self-training and SETRED are similar to that used by our algorithm.

we used three supervised inductive model as base learners to perform classifier induction, aiming to investigate how each comparing algorithm behaves along with base learners bearing diverse characteristics. Specifically, Naive Bayes,

TABLE II: Classification error rates of 3 compared algorithms on 12 datasets using naive bayes

Dataset	Classification error rates: Naive Bayes						
	initial	proposed algorithm		SETRED		Self-training	
		final	improve/%	final	improve/%	final	improve/%
australian	0.243	0.224	**7.9**	0.234	3.7	0.236	2.9
bupa	0.481	0.442	8.1	0.459	4.6	0.438	**8.9**
colic	0.217	0.207	**4.6**	0.212	2.3	0.221	-1.8
diabetes	0.267	0.257	3.7	0.245	**8.2**	0.264	1.1
german	0.285	0.281	1.4	0.276	**3.2**	0.298	-4.6
hypothyroid	0.024	0.021	**12.5**	0.023	4.2	0.021	**12.5**
ionosphere	0.155	0.129	**16.8**	0.151	2.6	0.166	-7.1
kr-vs-kp	0.142	0.137	3.5	0.143	-0.7	0.128	**9.9**
sick	0.089	0.084	5.6	0.084	5.6	0.079	**11.2**
tic-tac-toe	0.343	0.324	**5.5**	0.328	4.4	0.346	-0.9
vehicle	0.398	0.322	**19.1**	0.367	7.8	0.429	-7.8
wine	0.103	0.096	6.8	0.089	**13.6**	0.116	-12.6
average	–	–	**8.0**	–	4.9	–	1.0

TABLE III: Classification error rates of 3 compared algorithms on 12 datasets using CART

Dataset	Classification error rates: CART						
	initial	proposed algorithm		SETRED		Self-training	
		final	improve/%	final	improve/%	final	improve/%
australian	0.222	0.193	**13.0**	0.199	10.4	0.205	7.7
bupa	0.399	0.368	**7.8**	0.391	2.0	0.388	2.8
colic	0.181	0.163	**10.0**	0.174	3.9	0.168	7.2
diabetes	0.316	0.288	8.8	0.272	**13.9**	0.289	8.5
german	0.351	0.324	**7.6**	0.336	4.3	0.336	4.3
hypothyroid	0.012	0.011	**8.3**	0.016	-33.3	0.021	-75
ionosphere	0.155	0.121	**22.0**	0.144	7.1	0.149	3.9
kr-vs-kp	0.035	0.025	28.6	0.018	**48.6**	0.022	37.1
sick	0.024	0.021	**12.5**	0.026	-8.3	0.023	4.2
tic-tac-toe	0.292	0.258	**11.6**	0.268	8.2	0.277	5.1
vehicle	0.254	0.208	**18.1**	0.223	12.2	0.234	7.9
wine	0.085	0.061	**28.2**	0.072	15.3	0.064	24.7
average	–	–	**14.7**	–	7.0	–	3.2

CART, SVM models are used in the experiments. We use LibSVM[12] implementation for SVM. Note that only Naive Bayes is generative model that can yield probabilistic outputs, CART uses the proportion of dominating class in leaf node as probabilistic output and LibSVM is configured to give probabilistic estimates by using the training option ”-b 1”. For our algorithm and SETRED, we choose a medium number of nearest neighbors, i.e 8 for graph construction, we utilize EUCLIDEAN distance as the similarity measure mainly based on its simplicity and empirical evidences.

Experiments are carried out on each data set for 100 runs under randomly partitioned labeled/unlabeled/test split-s. TableII to TableIV present classification errors of these compared algorithms under different inductive models. The ”*initial*” column denotes the average error rates of classification with labeled data only. Columns denoted by ”*final*” and ”*improve*” represent the average error rates and performance improvements of each algorithm respectively.

TableII to TableIV show that proposed algorithm can effectively improve the performance with all the underlying in-

TABLE IV: Classification error rates of 3 compared algorithms on 12 datasets using LibSVM

Dataset	Classification error rates: LibSVM						
	initial	proposed algorithm		SETRED		Self-training	
		final	improve/%	final	improve/%	final	improve/%
australian	0.268	0.229	**14.6**	0.234	12.7	0.241	10.1
bupa	0.382	0.353	**7.6**	0.385	-0.8	0.390	-2.1
colic	0.267	0.176	**34.1**	0.198	25.6	0.235	12.1
diabetes	0.421	0.345	**18.0**	0.428	-1.6	0.448	-6.4
german	0.214	0.188	12.1	0.202	5.6	0.179	**16.4**
hypothyroid	0.029	0.024	17.2	0.021	**27.6**	0.026	10.3
ionosphere	0.183	0.164	10.4	0.142	**22.4**	0.181	1.1
kr-vs-kp	0.034	0.027	20.6	0.031	8.8	0.023	**32.4**
sick	0.036	0.034	4.0	0.031	**13.9**	0.033	8.3
tic-tac-toe	0.071	0.036	49.0	0.056	21.1	0.030	**57.7**
vehicle	0.398	0.316	**20.6**	0.367	7.8	0.429	-7.8
wine	0.103	0.048	**53.4**	0.066	36	0.116	-12.6
average	–	–	**21.8**	–	14.9	–	10.0

ductive model. he two-tailed paired t-test under the significant level of 95% shows that all the improvements of performance are significant. The biggest improvements achieved by these three self-training style algorithms have been boldfaced. In fact, if the improvements are averaged across all the data sets, base learners, it can be found that the average improvement of our algorithm is about 14.8%. It is impressive that with all the employed inductive models our algorithm has achieved the biggest improvement among the other 2compared algorithms. Moreover, if the algorithms are compared through counting the number of winning data sets, i.e. the number of data sets on which an algorithm has achieved the biggest improvement among the compared algorithms, our algorithm is almost always the winner. In detail, when Naive Bayes are used, it has 6 winning data sets while self-training has 4 and SETRED has 3; when CARTs are used, our algorithm and SETRED has 10 and 3 winning data sets respectively, while self-training do not have winning data sets; when SVMs are used, our algorithm, SETRED and self-training has 6, 3 and 3 winning data sets respectively.

In particular, comparing to SETRED which utilizes a specific data editing technique to actively identify wrongly-labeled examples in the enlarged training set, the proposed algorithm has achieved better results with no effort for cleaning the training set. This evidence supports our arguments that the incorporated transductive model is robust to noises introduced by the self-labeled process, thus it can achieve stable performance. Moreover, empirical results of on the 2 multi-class datasets(*vehicle,wine*) suggest that our algorithm is superior to self-training and SETRED when dealing with multi-class classification problems. This is mainly due to the fact that it can naturally handle multi-class classification by exploiting manifold assumption to yield confident predictions for training set augmentation.

IV. CONCLUSIONS

This paper shows the benefits of incorporating transdcutive models into semi-supervised bootstrap inductive models, such as self-training. This strategy utilizes both labeled and unlabeled data to yield more reliable predictions for unlabeled examples. We propose a robust self-training style algorithm which exploits manifold assumption to facilitate the self-training process. We adopt a transductive model based on graph random walks to prevent performance degradation due to classification noise accumulation. Empirical results on 12 UCI datasets show that proposed algorithm can effectively exploit unlabeled data to enhance performance.

Graph construction is vital to our algorithm. In this paper, we only use the common EUCLIDEAN distance as the distance measure, there is no guarantee that this is the optimal choice. Generally, the problem of choosing the best distance measure for a specific learning task is very difficult, and some efforts have been made towards tackling this problem under the name of distance metric learning. How to identify or learn the optimal distance measure and how does it affect the performance are worth further investigation.

ACKNOWLEDGMENT

The research is partially supported by National Natural Science Foundation of China under grant number 61173087,61073128. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] D. Steinberg and P. Colla, "Cart: classification and regression trees," *The Top Ten Algorithms in Data Mining*, pp. 179–201, 2009.

[2] X. Zhu, "Semi-supervised learning literature survey," *Computer Sciences TR 1530, Univ. of Wisconsin*, 2008.

[3] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1995, pp. 189–196.

[4] S. Abney, "Understanding the yarowsky algorithm," *Computational Linguistics*, vol. 30, no. 3, pp. 365–395, 2004.

[5] Y. Wang, X. Xu, H. Zhao, and Z. Hua, "Semi-supervised learning based on nearest neighbor rule and cut edges," *Knowledge-Based Systems*, vol. 23, no. 6, pp. 547–554, 2010.

[6] M. Li and Z.-H. Zhou, "Setred: Self-training with editing," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2005, pp. 611–621.

[7] M.-F. Balcan, S. Hanneke, and J. W. Vaughan, "The true sample complexity of active learning," *Machine learning*, vol. 80, no. 2-3, pp. 111–139, 2010.

[8] T. Joachims *et al.*, "Transductive learning via spectral graph partitioning," in *ICML*, vol. 3, 2003, pp. 290–297.

[9] A. Subramanya and J. Bilmes, "Semi-supervised learning with measure propagation," *The Journal of Machine Learning Research*, vol. 12, pp. 3311–3370, 2011.

[10] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly, "Video suggestion and discovery for youtube: taking random walks through the view graph," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 895–904.

[11] A. Frank and A. Asuncion, "Uci machine learning repository," 2010.

[12] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.