Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

Supervised Self-taught Learning: Actively Transferring Knowledge from Unlabeled Data

Kaizhu Huang, Zenglin Xu, Irwin King, Michael R. Lyu, and Colin Campbell

Abstract— **We consider the task of Self-taught Learning (STL) from unlabeled data. In contrast to semi-supervised learning, which requires unlabeled data to have the same set of class labels as labeled data, STL can transfer knowledge from different types of unlabeled data. STL uses a three-step strategy: (1) learning high-level representations from unlabeled data only, (2) re-constructing the labeled data via such representations and (3) building a classifier over the re-constructed labeled data. However, the high-level representations which are exclusively determined by the unlabeled data, may be inappropriate or even misleading for the latter classifier training process. In this paper, we propose a novel Supervised Self-taught Learning (SSTL) framework that successfully integrates the three isolated steps of STL into a single optimization problem. Benefiting from the interaction between the classifier optimization and the process of choosing high-level representations, the proposed model is able to select those discriminative representations which are more appropriate for classification. One important feature of our novel framework is that the final optimization can be iteratively solved with convergence guaranteed. We evaluate our novel framework on various data sets. The experimental results show that the proposed SSTL can outperform STL and traditional supervised learning methods in certain instances.**

I. Introduction

SELF-TAUGHT LEARNING (STL) utilizes information from unlabeled data and it has been a subject of active interest recently [18], [12]. The use of unlabeled data has been actively considered within the context of semi-supervised learning (SSL) [3], [24], [26]. However, SSL has the usual requirement that the unlabeled data shares the same distribution as the labeled data. In particular, the unlabeled data should contain the same implicit set of labels as those of the labeled data. Unfortunately, unlabeled data which shares the same label distribution as the limited labeled data may be difficult to obtain. Instead, a huge amount of seemingly irrelevant unlabeled data could be available and might be relevant to the learning task of interest. Self-taught learning has been proposed to deal with this learning scenario.

As an example we consider the automated classification of images of rare insects in the natural world. In this task, the number of labeled training samples may be limited and it may be difficult obtaining unlabeled images of the same insects. Both supervised learning and SSL potentially fail to solve this problem due to the limited amount of labeled and unlabeled images. In contrast, STL might improve the classification performance by utilizing a huge corpus of unlabeled images of different insects. These samples look seemingly irrelevant but can be easily obtained [18]. The motivation for STL is that this huge corpus of randomly chosen images may contain basic visual patterns, such as edges and colors which might be similar to those in images of interest. Another example would be that handwritten digits from other languages may help in the recognition of English characters, since these digits contain strokes that are similar to those in the English characters, although they have different labels. Studies by Raina [18] have demonstrated that STL could be promising for the types of tasks mentioned above and can indeed improve classification accuracy in some cases.

The learning procedure in STL can be divided into three separate stages. In the first stage, the high level representations (or *basis*), such as edges in the images or strokes in handwritten characters, are learned from available unlabeled data which are not from the same distribution class as labeled objects. In the second stage, STL represents the labeled data as a linear combination of these high level features or basis obtained from the first stage. Coefficients from the basis are then treated as input features for the next stage. In the third stage, we exploit traditional supervised learning algorithms, such as Support Vector Machines (SVM) [23], [21], to learn a decision function based on these coefficients.

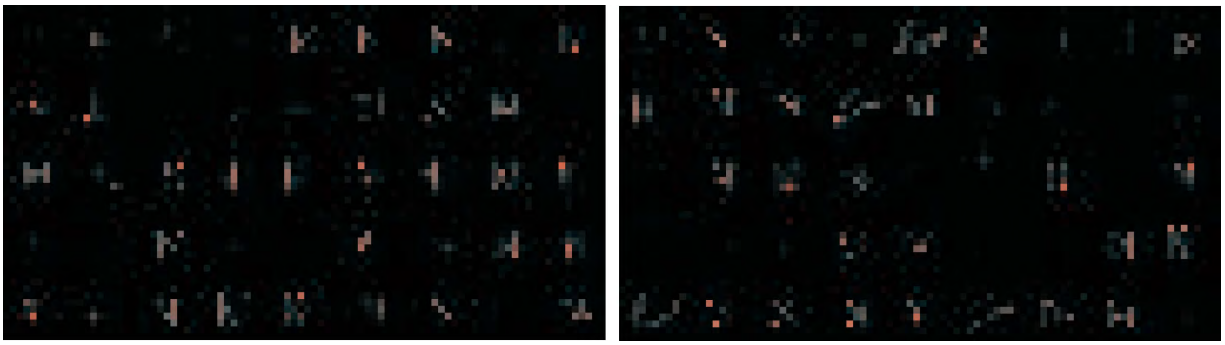
One problem with the framework described above is that the first stage is pursued in an ad hoc manner. Specifically, the learned high-level features in this step is determined by the unlabeled data only and this data could be very different from the labeled data. The learned representation might be unsuitable or even misleading for classifying the labeled data in the following two stages. To illustrate this shortcoming, we consider another example involving the classification of the two digits "1" and "7". Suppose that we have a huge number of other unlabeled uppercase English characters "I", "M", and "N". Obviously, the vertical strokes dominates the other strokes and no explicit horizontal stokes occur in these three characters. Hence, the feature of the horizontal stroke may not even appear in the final high-level features learned from the unlabeled data. However, to separate "1" and "7", the most discriminate feature is the horizontal stroke. Figure 1(a) visually shows the 50 high-level features extracted by STL from 200 "I", "M", "N" characters. As observed, almost no horizontal stroke patterns are extracted.

To solve this problem, we propose a novel Supervised Self-taught Learning (SSTL) model which manages to find the most appropriate high-level features or representations

Kaizhu Huang and Colin Campbell are with the Department of Engineering Mathematics, University of Bristol, Bristol BS8 1TR, United Kingdom (email: {K.Huang, C.Campbell}@bris.ac.uk). Zenglin Xu, Irwin King, and Michael R. Lyu are with Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong (email: {zlxu,king,lyu}@cse.cuhk.edu.hk).

The work described in this paper was partially supported by two grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4158/08E and No. CUHK4128/08E).

978-1-4244-3553-1/09/$25.00 ©2009 IEEE



(a) Patterns learned by STL (b) Patterns learned by our proposed framework

Fig. 1. High-level visual features extracted by STL and SSTL when classifying digits "1" and "7" with capital letters "I", "M", "N" as unlabeled samples. (a) and (b) presents the patterns learned by STL and SSTL respectively. In (a), STL fail to extract the discriminative features, i.e., horizontal stroke patterns. In (b), SSTL manages to learn many horizontal stroke patterns.

from the unlabeled data under *supervision* offered by the labeled training data. We attempt to learn from unlabeled data with the target input-output mapping in mind rather than to achieve this in an ad hoc manner. Specifically, the optimization is not performed in separate stages as in the original formulation of self-taught learning. Instead, the three stages of basis learning, coefficient optimization, and the classifier construction are integrated into a single optimization task. Thus the representations, the coefficients, and the classifier are optimized simultaneously. By interacting classifier optimization with choosing the high-level representations, the proposed model is able to select those discriminant features or representations, which are most appropriate for classification. Figure 1(b) demonstrates the high-level basis obtained by our SSTL framework in the "1" and "7" classification problem. As observed, the most discriminative features, the horizontal strokes, are indeed extracted.

To our knowledge, this is the first study that performs the Self-taught Learning in a supervised way. The underlying knowledge embedded in the unlabeled data can be transferred to the classification task actively and efficiently. In addition, one important feature of our framework is that the final optimization can be solved iteratively with a convergence guarantee. Moreover, we show that the proposed discriminative framework can even be formulated into a single optimization problem for the multi-class tasks. With these two advantages, the proposed framework can be easily applied in practice on many applications.

In the following, we first present the task and notation used throughout the paper. We then briefly review Self-taught Learning algorithm. We then present our novel Supervised Self-taught Learning framework. In Section V, we provide a series of experiments to verify the proposed framework. In Section VI, we discuss some issues raised by our study.

II. Problem Formalism

We consider a labeled training data set $D = \{(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots, (\mathbf{x}^l, y^l)\}$, consisting of l labeled samples drawn i.i.d. from a particular distribution S. Here $\mathbf{x}^i \in \mathbb{R}^n$ $(i = 1, 2, \ldots, l)$ describes an input feature vector, and $y^i \in \{1, 2, \ldots, q\}$ is the category label for $\mathbf{x}^i$. In addition, assume that m $(m \gg l)$ unlabeled data samples $\{\mathbf{x}^{l+1}, \mathbf{x}^{l+2}, \ldots, \mathbf{x}^{l+m}\}$ are also available. The basic task of STL and SSTL can be described as seeking a hypothesis $h : \mathbb{R}^n \rightarrow \{1, 2, \ldots q\}$ that can predict the label $y \in \{1, 2, \ldots q\}$ for the future input data sample $\mathbf{z} \in \mathbb{R}^n$ by appropriately exploiting both the labeled data and unlabeled data not drawn from the same distribution.

Remarks. The above problem is very different from the SSL. SSL requires that the unlabeled data should be sampled from the same distribution of the labeled data. However, here we relax this requirement and the unlabeled samples do not satisfy this constraint. Thus these unlabeled samples might share labels different from those of the labeled data. The problem is also very different from Transfer Learning (TL)[22], [4], [19] in that the latter framework requires the auxiliary data to be already labeled.

III. Self-taught Learning by Sparse Coding

STL solves the above task in three separate stages. We describe these three stages below:

A. Stage I: Learning Representations

In the first stage, high-level representations are learned from the unlabeled data. For instance, edges could be learned from images or strokes could be learned from alternative handwritten characters even if our purpose is to classify English handwritten digits. These high-level representations can be learned by using sparse coding (SC). SC is a powerful technique that has received much interest recently. It can learn an over-complete basis from data. We refer interested readers to [20], [15], [12], [16]. The formulation is as follows:

$$\begin{aligned} \min_{\mathbf{a},\mathbf{b}} \quad & \sum_{i=l+1}^{l+m} \|\mathbf{x}^i - \sum_{j=1}^{p} a_j^{(i)} \mathbf{b}_j\|_2^2 + \beta \|\mathbf{a}^{(i)}\|, \\ \text{s. t.} \quad & \|\mathbf{b}_j\|_2^2 \leq 1, j = 1, \ldots, p\,. \end{aligned}$$

$\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_p\}$ is called a set of *basis* with each basis $\mathbf{b}_j$ as an n-dimensional vector. $a_j^{(i)}$ is the activation

coefficient associated with the basis $\mathbf{b}_j$ for the sample $\mathbf{x}^i$. Hence $\mathbf{a}^{(i)}$ $(i = l+1, l+2, \ldots, l+m)$ is a set of activation coefficients for the unlabeled sample $\mathbf{x}^i$ with respect to all the basis $\mathbf{b}$. We denote $\mathbf{a}$ as a matrix defined as $(\mathbf{a}^{(l+1)}, \mathbf{a}^{(l+2)}, \ldots, \mathbf{a}^{(l+m)})$.

The above optimization problem tries to represent the unlabeled data in terms of $\mathbf{b}$. In more details, the first term in the optimization function describes the reconstruction error, while the second term with the L_1-norm forces the activation vector to be sparse. It is noted that the above optimization resembles Principal Component Analysis (PCA) [6] if the second term is omitted. However, SC enjoys several advantages over PCA. First, PCA can only generate a limited number of basis components (fewer than n), while SC could generate a large number of basis components whose number might be far larger than n. Second, PCA only results in linear feature extraction, while SC can deliver non-linear representations as imposed by the L_1-norm. With such advantages, SC outperforms PCA in many cases and is actively adopted to learn over-complete representations from data [20], [15].

B. Stage II: Feature Construction from the Basis

In the second stage, STL tries to represent the labeled data with respect to the basis $\mathbf{b}$. This stage is formulated as follows:

$$\min_{\mathbf{a}_L} \quad \sum_{i=1}^{l} \|\mathbf{x}^i - \sum_{j=1}^{p} a_L{}_j^{(i)} \mathbf{b}_j\|_2^2 + \beta \|\mathbf{a}_L^{(i)}\| \ .$$

In this stage, the features or the activation coefficients $\mathbf{a}_L$ for the labeled data are learned over the basis $\mathbf{b}$, which are obtained from the first stage. Similarly, the second term enforces sparsity in the coefficient vector.

C. Stage III: Learning a Classifier from Features

In the third stage, an SVM or other classifier can be exploited to learn the decision function $h = \mathbf{w} \cdot \mathbf{a_z} + c$ ($\mathbf{a_z}$, where $\mathbf{w}$ is the coefficient vector of the future sample $\mathbf{z}$) from the features constructed in Stage II. This is described in the following:

$$\begin{aligned} \min_{\mathbf{w},c} \quad & \sum_{i=1}^{l} \varepsilon_i + \gamma \|\mathbf{w}\|_2^2 \ , \\ \text{s. t.} \quad & y^k(\mathbf{w}.\mathbf{a}_L^{(k)} + c) \geq 1 - \varepsilon_k, \\ & \varepsilon_k \geq 0, k = 1, \ldots, l \ . \end{aligned}$$

Clearly, this optimization problem is the standard L_2-norm Support Vector Machine, except that the input features are the coefficients obtained in the second stage. In real applications, L_1-norm SVM [25] can also be adopted.

As observed from the above optimization, STL extracts the high-level representations from the unlabeled data only. However, these high-level representations may be inappropriate or even misleading for the latter classifier construction. The discriminative information, that proves critical for classification performance, may be discarded in this stage. In the next section, we propose the Supervised Self-taught Learning framework that successfully integrates the above three stages into one optimization problem. In the new framework, the high-level representation optimization is supervised by the classifier learning. The derived representations would be those discriminative patterns that will greatly benefit the classification performance.

IV. Supervised Self-taught Learning Framework

In this section, we present our novel Supervised Self-taught Learning framework. For the purpose of clarity, we first describe the framework in the binary setting. We then present an extension the framework to multi-class classification.

A. Two-class Model

The binary SSTL model is formulated as the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{i=1}^{l+m} \|\mathbf{x}^i - \sum_{j=1}^{p} a_j^{(i)} \mathbf{b}_j\|_2^2 + \beta \|\mathbf{a}^{(i)}\| + \lambda \sum_{i=1}^{l} \varepsilon_i + \gamma \|\mathbf{w}\|_2^2 \ , \\ \text{s. t.} \quad & \|\mathbf{b}_j\|_2^2 \leq 1, j = 1, \ldots, p \ , \\ & y^k(\mathbf{w} \cdot \mathbf{a}^{(k)} + c) \geq 1 - \varepsilon_k, \varepsilon_k \geq 0, k = 1, \ldots, l \ . \end{aligned}$$

In the above, $\mathbf{b}_j, j = 1 \ldots, p$ represents p basis extracted from the unlabeled data under the supervision of the labeled data. $a_j^{(i)}$ is the weight or the coefficient for the data point $\mathbf{x}^i$ with respect to the basis $\mathbf{b}_j$. $\{\mathbf{w}, c\}$ defines the classifier boundary[1].

The optimization not only minimizes the reconstruction error among both the labeled data and unlabeled data given by $\sum_{i=1}^{l+m} \|\mathbf{x}^i - \sum_{j=1}^{p} \mathbf{a}_j^{(i)} \mathbf{b}_j\|_2^2$, but also minimizes the error $\sum_{i=1}^{l} \varepsilon_i$ caused by the classifier on the labeled data. An L_1-norm and an L_2-norm is exploited as the regularization terms for $\mathbf{a}$ and $\mathbf{w}$, respectively. One could also use the L_1-norm for $\mathbf{w}$. The basis $\mathbf{b}$ and the classifier $\{\mathbf{w}, c\}$ are optimized simultaneously. This is very different from the original self-taught framework using sparse coding, where the high-level features, determined exclusively by the unlabeled data, might be misleading and detract from classification performance.

In similar fashion to the original sparse coding scheme, the above optimization problem is not convex. However, it is convex in $\mathbf{a}$ (while holding $\{\mathbf{b}, \mathbf{w}, c, \varepsilon\}$ fixed) and also convex in $\{\mathbf{b}, \mathbf{w}, c, \varepsilon\}$ (while holding $\mathbf{a}$ fixed). In the following, we show how to solve the optimization problem iteratively in two steps.

B. Optimization Method

We propose the following iterative algorithm to perform the optimization. When $\mathbf{b}, \mathbf{w}, c, \varepsilon$ are fixed, it is easy to verify that the optimization problem of finding $\mathbf{a}$ is reduced to the following two sub-problems.

Problem I(a):

$$\min_{\mathbf{a}^{(i)}} \quad \|\mathbf{x}^i - \sum_{j=1}^{p} \mathbf{a}_j^{(i)} \mathbf{b}_j\|_2^2 + \beta \|\mathbf{a}^{(i)}\|, \ i = l+1, \ldots, l+m \ .$$

[1]For binary problems, we modify the class labels as {-1,+1}.

Problem I(b):

$$\begin{aligned} \min_{\mathbf{a}^{(i)}} \quad & \|\mathbf{x}^i - \sum_{j=1}^{p} \mathbf{a}_j^{(i)} \mathbf{b}_j\|_2^2 + \beta \|\mathbf{a}^{(i)}\|, i = 1, \ldots, l, \\ \text{s. t.} \quad & y^i(\mathbf{w} \cdot \mathbf{a}^{(i)} + c) \geq 1 - \varepsilon_i, \varepsilon_i \geq 0 \, . \end{aligned}$$

Problem I(a) describes the optimization over unlabeled data, while I(b) presents the optimization over the labeled data points. Problem I(a) is equivalent to a regularized least squares problem. I(b) is similar except that it has a linear constraint. Both problems can be solved by many methods, for example, feature-sign search algorithm [12], the interior point method [14] or a generic convex programming solver (CVX)[2].

Similarly, when $\mathbf{a}$ is fixed, the optimization problem of finding $\mathbf{b}, \mathbf{w}, c, \varepsilon$ is altered to the following two sub problems.

Problem II(a):

$$\begin{aligned} \min_{\mathbf{w}, c, \varepsilon} \quad & \gamma \|\mathbf{w}\|_2^2 + \lambda \sum_{k=1}^{l} \varepsilon_k, \\ \text{s. t.} \quad & y^k(\mathbf{w} \cdot \mathbf{a}^{(k)} + c) \geq 1 - \varepsilon_k, \varepsilon_k \geq 0, k = 1, \ldots, l \, . \end{aligned}$$

Problem II(b):

$$\begin{aligned} \min_{\mathbf{b}} \quad & \sum_{i=1}^{l+m} \|\mathbf{x}^i - \sum_{j=1}^{p} a_j^{(i)} \mathbf{b}_j\|_2^2 \, , \\ \text{s. t.} \quad & \|\mathbf{b}_j\|_2^2 \leq 1, j = 1, 2, \ldots p \, . \end{aligned}$$

Problem II(a) and II(b) are typical quadratic programming problems. More specifically, II(a) is the standard L_2-norm SVM optimization problem. II(b) is a Quadratically Constrained Quadratic Programming problem (QCQP) [1], [13], [2]. They can be solved either by the SMO algorithm [17] or the dual algorithm proposed in [12].

Since the value of the optimization objective $f(\mathbf{a}, \mathbf{b}, \mathbf{w}, \varepsilon)$ will be decreased after solving each problem, solving the above two problems alternatively will guarantee a convergence to a fixed point. As a summary, we present the optimization algorithm in Algorithm 1.

C. Multi-class Model

In this section we provide details for a one-against-others strategy to extend SSTL to multi-class tasks.

Before we present the problem definition for the multi-class model, we define some notations in the following. Let $\mathbf{I}_o$ be a diagonal matrix with the element 1 at $(0, 0)$ and all the other diagonal elements being -1. Assume that $\{\mathbf{w}_o, c_o\}$ is the decision function associated with the o-th class ($1 \leq o \leq q$, i.e., there are q classes) . We further define $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_q)$ and $\mathbf{c} = (c_0, \ldots, c_q)^T$.

[2]The matlab source codes of the CVX package can be downloaded from http://www.stanford.edu/ boyd/cvx/.

The multi-category SSTL model is defined as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{l+m} \{\|\mathbf{x}^i - \sum_{j=1}^{p} \mathbf{a}_j^{(i)} \mathbf{b}_j\|_2^2 + \beta \|\mathbf{a}^{(i)}\|\} \\ + \quad & \lambda \sum_{i=1}^{l} \sum_{o=1}^{q} \varepsilon_o^i + \gamma \sum_{o=1}^{q} \|\mathbf{w}_o\|_2^2 \qquad (1) \\ \text{s. t.} \quad & \|\mathbf{b}_j\|^2 \leq 1, j = 1, \ldots, p \, , \\ & \mathbf{I}_o(\mathbf{W}^T \mathbf{a}^{(k)} + \mathbf{c}) \geq \mathbf{e} - \varepsilon^k, \\ & \varepsilon^k \geq 0, k = 1, \ldots, l \, . \end{aligned}$$

In the above, ε^k represents a q-dimensional slack vector for the k-th labeled data sample. Each element of ε^k, i.e., ε_o^k ($1 \leq o \leq q$) represents the hinge loss incurred by the classifier $\{\mathbf{w}_o, c_o\}$ with respect to $\mathbf{x}^k$. $\varepsilon^k \geq 0$ means each element of ε^k is not less than 0. $\mathbf{e}$ is a vector with all the elements set to one. Other variables are similarly defined to those in the binary case.

Input: Labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$; unlabeled data $\{(\mathbf{x}_i, y_i)\}_{i=l+1}^{l+m}$

Step 1. Initialize $\mathbf{a}_{(0)}$; set Δ to a small positive value; set the number of iterations $t = 1$.

Step 2. Compute $\{\mathbf{w}_{(t)}, c_{(t)}, \varepsilon_{(t)}, \mathbf{b}\}$.

a. Calculate $\{\mathbf{w}_{(t)}, c_{(t)}, \varepsilon_{(t)}\}$ by solving Problem II(a).

b. Calculate $\mathbf{b}_{(t)}$ by solving Problem II(b).

Step 3. Compute $\{\mathbf{a}_{(t)}^{(i)}\}_{i=1}^{l+m}$.

a. Calculate $\{\mathbf{a}_{(t)}^{(i)}\}_{i=l+1}^{l+m}$ by solving Problem I(a).

b. Calculate $\{\mathbf{a}_{(t)}^{(i)}\}_{i=1}^{l}$ by solving Problem I(b).

Step 4. If $t < T_{\text{MAX}}$ (e.g., $T_{\text{MAX}} = 100$) and $||f_{(t)}(\mathbf{a}, \mathbf{b}, \mathbf{w}, \varepsilon) - f_{(t-1)}(\mathbf{a}, \mathbf{b}, \mathbf{w}, \varepsilon)|| > \Delta$, then $t \leftarrow t + 1$; go to Step 2; otherwise stop.

Output: The classifier $\{\mathbf{w}, c\} \leftarrow \{\mathbf{w}_{(t)}, c_{(t)}\}$

Algorithm 1: Supervised Self-taught Learning Via Sparse Coding

In the following we will focus on interpreting the above multi-class model. First, in binary classification, each labeled sample is used only once. However, in multi-class classification, each labeled sample will be used q times, since there are q classifiers. Hence the hinge loss for each labeled sample is not a scale variable anymore. Instead, it is a q-dimensional vector. Second, the key point of multi-class classification using Sparse Coding is to derive a common set of basis components for all the q classifiers involved. This requires that the single optimization task be formulated for q classifiers. Our model successfully achieves this goal. Finally, as observed from the above model, the optimization can still be optimized in two steps. Moreover, each step is easily verified to be convex as well. Hence it can be solved using a method similar to that presented in the previous section.

V. EXPERIMENTS

In this section, we evaluate our proposed Supervised Self-taught Learning algorithm on various datasets. Specifically, we evaluate the SSTL framework on web text categorization tasks. We adopt four subsets of text documents for the evaluation from three benchmark text collections, namely WebKB [3], Reuters-21578 [4], and Ohsumed [5]. The selected data sets, *course* vs. *non-course*, which are obtained from the WebKB corpus, contain course web pages and non-course web pages from several universities. The *bacterial* vs. *virus* data and the *male* vs. *female* data are extracted from the Ohsumed database that is a set of references from MEDLINE, the on-line medical information database, consisting of titles and/or abstracts from medical journals. The *grain* vs. *wheat* data set is from the Reuters-21578 Text Categorization Collection, which is a collection of documents that appeared on Reuters newswire in 1987. The description of the four selected data sets can be found in Table I.

TABLE I
DESCRIPTIONS FOR THE WEB TEXT DOCUMENTS DATA

Corpus	Labeled Data	# Documents
WebKB	*course* vs. *non-course*	1051
Ohsumed	*bacterial* vs. *virus*	581
	male vs. *female*	871
Reuters	*grain* vs. *wheat*	865

We conducted two sets of experiments. In the first set of experiments, we randomly select 4 labeled documents from each data set to form the training set, and used the remaining documents as the test set. In the second set of experiments, 10 labeled documents are randomly selected to form the training set. In order to generate the unlabeled data for self-taught learning algorithms, we first select the keywords from the given training data and then mine the Internet to get a set of unlabeled web pages using the keywords as the query terms. Here Google is used as the search engine and we select the top 1000 returned web pages as the unlabeled data for each dataset. We then represent each document by a vector of term frequency. We select 500 most informative features according to their correlation to the text categories. Note that, due to both the inaccuracy of query keywords and the ambiguity of the searching engines, the returned web pages contain many irrelevant documents. SSL cannot be directly applied in this task. The parameters are selected based on cross validation. And the final results are the average over 10 runs using the above training and testing procedure.

The experimental results are listed in Table II. As observed from the presented results, STL indeed increases the recognition accuracies of supervised learning in some datasets, e.g. *male* vs. *female* when the training size is equal to 4. However, in many cases, STL demonstrates much worse performance than purely supervised learning, e.g. in *course* vs. *non-course* and *grain* vs. *wheat* when the training size is equal to 4. Because web documents are usually of both high-dimension and of high sparsity, without supervision from the labeled data, it is very likely that STL extracts non-important or even noisy basis components from the unlabeled data. This explains why STL sometimes degrades performance. In comparison, our proposed SSTL successfully avoids this problem. SSTL attempts to detect the most discriminative patterns as the basis by supervising the self-taught learning process via the labeled data. As clearly seen in Table II, SSTL is consistently better than or equivalent to STL and SVM in all the four data sets. The difference between SSTL and the other two algorithms is more distinct in the *course* vs. *non-course* data set: the accuracy of SSTL is almost double that of SVM, and is also significantly higher than STL. The experimental results clearly demonstrate the advantages of our proposed learning framework.

[3] http://www.cs.cmu.edu/~webkb/

[4] http://www.daviddlewis.com/resources/testcollections/

[5] ftp://medir.ohsu.edu/pub/ohsumed

VI. DISCUSSION

We now discuss some issues raised by this project in this section. First, the Self-taught Learning framework is very different from other current learning paradigms. The core idea of STL is to boost classification performance when the labeled data is limited by appropriately transferring knowledge from seemingly irrelevant unlabeled data. This is very different from the semi-supervised learning algorithms in that SSL requires the unlabeled data to follow the same distribution as the labeled data. It is also different from Transfer Leaning algorithms in that TL only transfers knowledge from labeled data. Our proposed Supervised Self-taught Learning algorithm is still positioned within the self-taught learning paradigm, but it focuses on transferring the knowledge from unlabeled data in a supervised or discriminative way. In other words, SSTL proposes to extract "useful" information from unlabeled data which could improve classification performance. This is very different from the original Self-taught Learning algorithm in that STL transfers knowledge from unlabeled data in an unstructured fashion.

Second, it is standard to combine discriminative learning algorithms with generative or unsupervised learning methods [11], [8], [7], [10], [9], [5]. Our proposed SSTL is also motivated by this idea. However, these methods are still supervised learning algorithms because they perform such hybrid learning only for the labeled data. In contrast, our proposed algorithm tries to learn discriminative information from unlabeled data. This is the major difference between our algorithm and these hybrid methods. In addition, we believe the hybrid techniques specially designed for supervised learning could also be applied in the SSTL framework. More specifically, we notice that the discriminative sparse coding algorithm proposed in [7] might be used to further improve the classification accuracy. We leave this topic as future work.

Third, we only focus on studying the Supervised Self-taught Learning framework by applying the sparse coding algorithm. Obviously, there are a lot of other algorithms that could be applied to this new learning framework. It is

TABLE II

COMPARISONS ON WEB TEXT CATEGORIZATION TASKS. STL PERFORMS WORSE THAN SVM DUE TO INAPPROPRIATE HIGH-LEVEL REPRESENTATIONS. SSTL PRESENTS THE BEST RESULTS CONSISTENTLY BY INCORPORATING KNOWLEDGE SELECTIVELY AND DISCRIMINATIVELY.

	Training Size= 4			**Training Size**= 10		
Data Set	SVM	STL	SSTL	SVM	STL	SSTL
course vs. *non-course*	39.48	34.39	**78.19**	45.18	87.48	**91.21**
bacterial vs. *virus*	61.82	53.42	**62.49**	**73.14**	72.79	**73.14**
male vs. *female*	52.49	64.70	**65.52**	63.41	53.66	**68.25**
grain vs. *wheat*	57.63	51.93	**61.52**	65.39	67.02	**69.38**

interesting to investigate how other existing algorithms can be adapted to the SSTL framework.

Finally, although we have successfully integrated the three isolated optimization problems of STL into a single optimization task, it introduces several extra parameters in order to balance the reconstruction errors in the unlabeled data and the optimization values contributed by the classifier learning. Currently, these parameters are tuned manually or by cross validation. It is preferable to devise a more efficient algorithm to speed up the parameter selection process. We leave this task as an open problem.

VII. CONCLUSION

In this paper, we have presented a study on a Supervised Self-taught Learning framework, which can transfer knowledge from unlabeled data actively. This framework successfully integrates the three-step optimization into a single optimization problem. By integrating classifier optimization with choosing the high-level representations, the proposed model is able to select those discriminant features or representations, which are more appropriate for classification. Hence this may benefit the classification performance greatly. To our knowledge, this is the first work that performs Self-taught Learning in a supervised way. We have demonstrated that the novel framework reduces to solving four sub optimization problems iteratively, each of them being convex. Moreover, the final optimization can be iteratively solved with convergence guaranteed. Extensive evaluations on web data have shown that our proposed algorithm can improve the classification performance against the original Self-taught Learning algorithm and a purely supervised learning algorithm when the amount of labeled data is limited.

REFERENCES

[1] M. S. Bazaraa. *Nonlinear Programming: Theory and Algorithms*. New York: Wiley, 2nd edition, 1993.

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massashusetts, 2nd edition, 1999.

[3] O. Chapelle, A. Zien, and Schölkopf B. *Semi-supervised learning*. MIT Press, Cambridge, MA, 2006.

[4] W. Dai, Q. Yang, G. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning (ICML-2007)*, pages 193–200, New York, NY, USA, 2007. ACM Press.

[5] G. Druck, C. Pal, A. McCallum, and X. Zhu. Semi-supervised classification with hybrid generativediscriminative methods. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD-2007)*, pages 280–289, New York, NY, USA, 2007. ACM.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2000.

[7] Stephan Hasler, Heiko Wersing, and Edgar Korner. Combining reconstruction and discrimination with class-specific sparse coding. *Neural Computation*, 19:1897–1918, 2007.

[8] K. Huang, I. King, and M. R. Lyu. Discriminative training of bayesian chow-liu tree multinet classifiers. In *Proceedings of International Joint Conference on Neural Network (IJCNN-2003), Oregon, Portland, U.S.A.*, volume 1, pages 484–488, 2003.

[9] K. Huang, H. Yang, I. King, and M. R. Lyu. *Machine Learning: Modeling Data Locally and Gloablly*. Springer Verlag, ISBN 3-5407-9451-4, 2008.

[10] K. Huang, H. Yang, I. King, and M. R. Lyu. Maxi-min margin machine: Learning large margin classifiers globally and locally. *IEEE Transactions on Neural Networks*, 19:260–272, 2008.

[11] T. Jebara. *Machine Learning: Discriminative and Generative*. Kluwer, ISBN 1-4020-7647-9, 2003.

[12] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 82–90, 2007.

[13] Olvi L. Mangasarian. *Nonlinear programming*. Philadelphia : Society for Industrial and Applied Mathematics, 1994.

[14] Y. Nesterov and A. Nemirovsky. *Interior point polynomial methods in convex programming: Theory and applications*. Studies in Applied Mathematics. Philadelphia, 1994.

[15] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, (381):607–609, 1996.

[16] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37:3311–3325, 1997.

[17] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Technical Report MSR-TR-98-14*, 1998.

[18] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of International Conference on Machine Learning (ICML-2007)*, pages 759 – 766, 2007.

[19] Rajat Raina, Andrew Y. Ng, and Daphne Koller. Constructing informative priors using transfer learning. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 713–720, New York, NY, USA, 2006. ACM Press.

[20] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, (290):123–137, 2000.

[21] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[22] S. Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems (NIPS)*, 1996.

[23] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 2nd edition, 2000.

[24] T. Zhang and R. Ando. Analysis of spectral kernel design based semi-supervised learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems (NIPS 18)*, pages 1601–1608. MIT Press, Cambridge, MA, 2006.

[25] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Advances in Neural Information Processing Systems (NIPS 16)*, 2003.

[26] Xiaojin Zhu. Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2005.