Self-taught Learning: Transfer Learning from Unlabeled Data

Rajat Raina RAJATR@CS.STANFORD.EDU
Alexis Battle AJBATTLE@CS.STANFORD.EDU
Honglak Lee HLLEE@CS.STANFORD.EDU
Benjamin Packer BPACKER@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department, Stanford University, CA 94305 USA

Abstract

We present a new machine learning framework called "self-taught learning" for using unlabeled data in supervised classification tasks. We do not assume that the unlabeled data follows the same class labels or generative distribution as the labeled data. Thus, we would like to use a large number of unlabeled images (or audio samples, or text documents) randomly downloaded from the Internet to improve performance on a given image (or audio, or text) classification task. Such unlabeled data is significantly easier to obtain than in typical semi-supervised or transfer learning settings, making self-taught learning widely applicable to many practical learning problems. We describe an approach to self-taught learning that uses sparse coding to construct higher-level features using the unlabeled data. These features form a succinct input representation and significantly improve classification performance. When using an SVM for classification, we further show how a Fisher kernel can be learned for this representation.

1. Introduction

Labeled data for machine learning is often very difficult and expensive to obtain, and thus the ability to use unlabeled data holds significant promise in terms of vastly expanding the applicability of learning methods. In this paper, we study a novel use of unlabeled data for improving performance on supervised learning tasks. To motivate our discussion, consider as a running example the computer vision task of classifying images of elephants and rhinos. For this task, it is difficult to obtain many labeled examples of elephants and rhinos; indeed, it is difficult even to obtain many *unlabeled* examples of elephants and rhinos. (In fact, we find it difficult to envision a process for collecting such unlabeled images, that does not immediately also provide the class labels.) This makes the classification task quite hard with existing algorithms for using labeled and unlabeled data, including most semi-supervised learning algorithms such as the one by Nigam et al. (2000). In this paper, we ask how unlabeled images from *other* object classes—which are much easier to obtain than images specifically of elephants and rhinos—can be used. For example, given unlimited access to unlabeled, randomly chosen images downloaded from the Internet (probably none of which contain elephants or rhinos), can we do better on the given supervised classification task?

Our approach is motivated by the observation that even many randomly downloaded images will contain basic visual patterns (such as edges) that are similar to those in images of elephants and rhinos. If, therefore, we can learn to recognize such patterns from the unlabeled data, these patterns can be used for the supervised learning task of interest, such as recognizing elephants and rhinos. Concretely, our approach learns a succinct, higher-level feature representation of the inputs using unlabeled data; this representation makes the classification task of interest easier.

Although we use computer vision as a running example, the problem that we pose to the machine learning community is more general. Formally, we consider solving a supervised learning task given labeled and unlabeled data, where the unlabeled data does not share the class labels or the generative distribution of the labeled data. For example, given unlimited access to natural sounds (audio), can we perform better speaker identification? Given unlimited access to news articles (text), can we perform better email foldering of "ICML reviewing" vs. "NIPS reviewing" emails?

Like semi-supervised learning (Nigam et al., 2000), our algorithms will therefore use labeled and unlabeled data. But unlike semi-supervised learning as it is typically studied in the literature, *we do not assume that the unlabeled data can be assigned to the supervised learning task's class labels.* To thus distinguish our formalism from such forms of semi-supervised learning, we will call our task *self-taught learning*.

There is no prior general, principled framework for incorporating such unlabeled data into a supervised

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

learning algorithm. Semi-supervised learning typically makes the additional assumption that the unlabeled data can be labeled with the same labels as the classification task, and that these labels are merely unobserved (Nigam et al., 2000). Transfer learning typically requires further labeled data from a different but related task, and at its heart typically transfers knowledge from one supervised learning task to another; thus it requires additional labeled (and therefore often expensive-to-obtain) data, rather than unlabeled data, for these other supervised learning tasks.[1] (Thrun, 1996; Caruana, 1997; Ando & Zhang, 2005) Because self-taught learning places significantly fewer restrictions on the type of unlabeled data, in many practical applications (such as image, audio or text classification) it is much easier to apply than typical semi-supervised learning or transfer learning methods. For example, it is far easier to obtain 100,000 Internet images than to obtain 100,000 images of elephants and rhinos; far easier to obtain 100,000 newswire articles than 100,000 articles on ICML reviewing and NIPS reviewing, and so on. Using our running example of image classification, Figure 1 illustrates these crucial distinctions between the self-taught learning problem that we pose, and previous, related formalisms.

We pose the self-taught learning problem mainly to formalize a machine learning framework that we think has the potential to make learning significantly easier and cheaper. And while we treat any biological motivation for algorithms with great caution, the self-taught learning problem perhaps also more accurately reflects how humans may learn than previous formalisms, since much of human learning is believed to be from *unlabeled* data. Consider the following informal order-of-magnitude argument.[2] A typical adult human brain has about 10^{14} synapses (connections), and a typical human lives on the order of 10^9 seconds. Thus, even if each synapse is parameterized by just a one bit parameter, a learning algorithm would require about $10^{14}/10^9 = 10^5$ bits of information per second to "learn" all the connections in the brain. It seems extremely unlikely that this many bits of labeled information are available (say, from a human's parents or teachers in his/her youth). While this argument has many (known) flaws and is not to be taken too seriously, it strongly suggests that most of human learning is *unsupervised*, requiring only data without any labels (such as whatever natural images, sounds, etc. one may encounter in one's life).

[1]We note that these additional supervised learning tasks can sometimes be created via ingenious heuristics, as in Ando & Zhang (2005).

[2]This argument was first described to us by Geoffrey Hinton (personal communication) but appears to reflect a view that is fairly widely held in neuroscience.

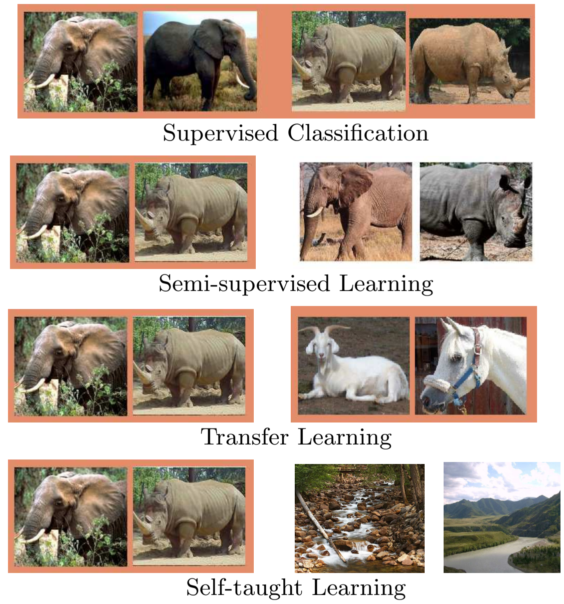

Figure 1. Machine learning formalisms for classifying images of elephants and rhinos. Images on orange background are labeled; others are unlabeled. Top to bottom: Supervised classification uses labeled examples of elephants and rhinos; semi-supervised learning uses additional unlabeled examples of elephants and rhinos; transfer learning uses additional labeled datasets; self-taught learning just requires additional unlabeled images, such as ones randomly downloaded from the Internet.

Inspired by these observations, in this paper we present largely unsupervised learning algorithms for improving performance on supervised classification tasks. Our algorithms apply straightforwardly to different input modalities, including images, audio and text. Our approach to self-taught learning consists of two stages: First we learn a representation using only unlabeled data. Then, we apply this representation to the labeled data, and use it for the classification task. Once the representation has been learned in the first stage, it can then be applied repeatedly to different classification tasks; in our example, once a representation has been learned from Internet images, it can be applied not only to images of elephants and rhinos, but also to other image classification tasks.

2. Problem Formalism

In self-taught learning, we are given a labeled training set of m examples $\{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), \ldots, (x_l^{(m)}, y^{(m)})\}$ drawn i.i.d. from some distribution $\mathcal{D}$. Here, each $x_l^{(i)} \in \mathbb{R}^n$ is an input feature vector (the "l" subscript indicates that it is a labeled example), and $y^{(i)} \in \{1, \ldots, C\}$ is the corresponding class label. In addition, we are given a set of k unlabeled examples $x_u^{(1)}, x_u^{(2)}, \ldots, x_u^{(k)} \in \mathbb{R}^n$. Crucially, we do not assume that the unlabeled data $x_u^{(j)}$ was drawn from the same distribution as, nor that it can be associated with the

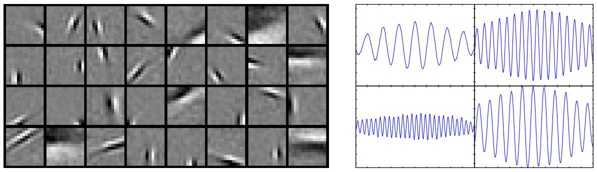

Figure 2. Left: Example sparse coding bases learned from image patches (14x14 pixels) drawn from random grayscale images of natural scenery. Each square in the grid represents one basis. Right: Example acoustic bases learned by the same algorithm, using 25ms sound samples from speech data. Each of the four rectangles in the 2x2 grid shows the 25ms long acoustic signal represented by a basis vector.

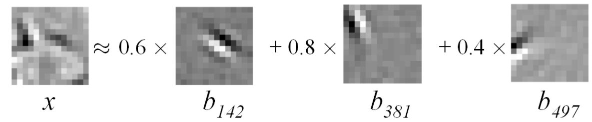

Figure 3. The features computed for an image patch (left) by representing the patch as a sparse weighted combination of bases (right). These features act as robust edge detectors.

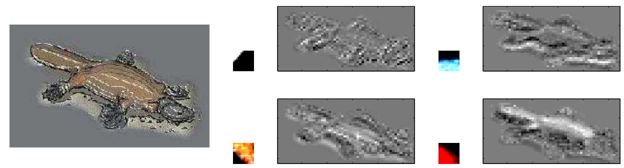

Figure 4. Left: An example platypus image from the Caltech 101 dataset. Right: Features computed for the platypus image using four sample image patch bases (trained on color images, and shown in the small colored squares) by computing features at different locations in the image. In the large figures on the right, white pixels represents highly positive feature values for the corresponding basis, and black pixels represents highly negative feature values. These activations capture higher-level structure of the input image. (Bases have been magnified for clarity; best viewed in color.)

same class labels as, the labeled data. Clearly, as in transfer learning (Thrun, 1996; Caruana, 1997), the labeled and unlabeled data should not be completely irrelevant to each other if unlabeled data is to help the classification task. For example, we would typically expect that $x_l^{(i)}$ and $x_u^{(j)}$ come from the same input "type" or "modality," such as images, audio, text, etc.

Given the labeled and unlabeled training set, a self-taught learning algorithm outputs a hypothesis $h : \mathbb{R}^n \rightarrow \{1, \ldots, C\}$ that tries to mimic the input-label relationship represented by the labeled training data; this hypothesis h is then tested under the same distribution $\mathcal{D}$ from which the labeled data was drawn.

3. A Self-taught Learning Algorithm

We hope that the self-taught learning formalism that we have proposed will engender much novel research in machine learning. In this paper, we describe just one approach to the problem.

We present an algorithm that begins by using the unlabeled data $x_u^{(i)}$ to learn a slightly higher-level, more succinct, representation of the inputs. For example, if the inputs $x_u^{(i)}$ (and $x_l^{(i)}$) are vectors of pixel intensity values that represent images, our algorithm will use $x_u^{(i)}$ to learn the "basic elements" that comprise an image. For example, it may discover (through examining the statistics of the unlabeled images) certain strong correlations between rows of pixels, and therefore learn that most images have many *edges*. Through this, it then learns to represent images in terms of the edges that appear in it, rather than in terms of the raw pixel intensity values. This representation of an image in terms of the edges that appear in it—rather than the raw pixel intensity values—is a *higher level,* or more *abstract,* representation of the input. By applying this learned representation to the labeled data $x_l^{(i)}$, we obtain a higher level representation of the labeled data also, and thus an easier supervised learning task.

3.1. Learning Higher-level Representations

We learn the higher-level representation using a modified version of the sparse coding algorithm due to Olshausen & Field (1996), which was originally proposed as an unsupervised computational model of low-level sensory processing in humans. More specifically, given the unlabeled data $\{x_u^{(1)}, ..., x_u^{(k)}\}$ with each $x_u^{(i)} \in \mathbb{R}^n$, we pose the following optimization problem:

$$\begin{array}{ll} \text{minimize}_{b,a} & \sum_i \|x_u^{(i)} - \sum_j a_j^{(i)} b_j\|_2^2 + \beta \, \|a^{(i)}\|_1 \quad (1) \\ \text{s.t.} & \|b_j\|_2 \leq 1, \quad \forall j \in 1, ..., s \end{array}$$

The optimization variables in this problem are the *basis vectors* $b = \{b_1, b_2, \ldots, b_s\}$ with each $b_j \in \mathbb{R}^n$, and the *activations* $a = \{a^{(1)}, \ldots, a^{(k)}\}$ with each $a^{(i)} \in \mathbb{R}^s$; here, $a_j^{(i)}$ is the activation of basis b_j for input $x_u^{(i)}$. The number of bases s can be much larger than the input dimension n. The optimization objective (1) balances two terms: (i) The first quadratic term encourages each input $x_u^{(i)}$ to be reconstructed well as a weighted linear combination of the bases b_j (with corresponding weights given by the activations $a_j^{(i)}$); and (ii) it encourages the activations to have low L_1 norm. The latter term therefore encourages the activations a to be *sparse*—in other words, for most of its elements to be zero. (Tibshirani, 1996; Ng, 2004)

This formulation is actually a modified version of Olshausen & Field's, and can be solved significantly more efficiently. Specifically, the problem (1) is convex over each subset of variables a and b (though not jointly convex); in particular, the optimization over activations a is an L_1-regularized least squares problem, and the optimization over basis vectors b is an L_2-constrained least squares problem. These two convex sub-problems can be solved efficiently, and the objective in problem (1) can be iteratively optimized over a and b alternatingly while holding the other set of variables fixed. (Lee et al., 2007)

As an example, when this algorithm is applied to small 14x14 images, it learns to detect different edges in the image, as shown in Figure 2 (left). Exactly the same algorithm can be applied to other input types, such as audio. When applied to speech sequences, sparse coding learns to detect different patterns of frequencies, as shown in Figure 2 (right).

Importantly, by using an L_1 regularization term, we obtain extremely sparse activations—only a few bases are used to reconstruct any input $x_u^{(i)}$; this will give us a succinct representation for $x_u^{(i)}$ (described later). We note that other regularization terms that result in most of the $a_j^{(i)}$ being non-zero (such as that used in the original Olshausen & Field algorithm) do not lead to good self-taught learning performance; this is described in more detail in Section 4.

3.2. Unsupervised Feature Construction

It is often easy to obtain large amounts of unlabeled data that shares several salient features with the labeled data from the classification task of interest. In image classification, most images contain many edges and other visual structures; in optical character recognition, characters from different scripts mostly comprise different pen "strokes"; and for speaker identification, speech even in different languages can often be broken down into common sounds (such as phones).

Building on this observation, we propose the following approach to self-taught learning: We first apply sparse coding to the unlabeled data $x_u^{(i)} \in \mathbb{R}^n$ to learn a set of bases b, as described in Section 3.1. Then, for each training input $x_l^{(i)} \in \mathbb{R}^n$ from the classification task, we compute features $\hat{a}(x_l^{(i)}) \in \mathbb{R}^s$ by solving the following optimization problem:

$$\hat{a}(x_l^{(i)}) = \arg\min_{a^{(i)}} \|x_l^{(i)} - \textstyle\sum_j a_j^{(i)} b_j\|_2^2 + \beta\, \|a^{(i)}\|_1 \quad (2)$$

This is a convex L_1-regularized least squares problem and can be solved efficiently (Efron et al., 2004; Lee et al., 2007). It approximately expresses the input $x_l^{(i)}$ as a sparse linear combination of the bases b_j. The sparse vector $\hat{a}(x_l^{(i)})$ is our new representation for $x_l^{(i)}$.

Using a set of 512 learned image bases (as in Figure 2, left), Figure 3 illustrates a solution to this optimization problem, where the input image x is approximately expressed as a combination of three basis vectors $b_{142}, b_{381}, b_{497}$. The image x can now be represented via the vector $\hat{a} \in \mathbb{R}^{512}$ with $\hat{a}_{142} = 0.6$, $\hat{a}_{381} = 0.8$, $\hat{a}_{497} = 0.4$. Figure 4 shows such features $\hat{a}$ computed for a large image. In both of these cases, the computed features capture aspects of the higher-level structure of the input images. This method applies equally well to other input types; the features computed on audio samples or text documents similarly detect useful higher-level patterns in the inputs.

We use these features as input to standard supervised classification algorithms (such as SVMs). To classify a test example, we solve (2) to obtain our representation $\hat{a}$ for it, and use that as input to the trained classifier. Algorithm 1 summarizes our algorithm for self-taught learning.

Algorithm 1 Self-taught Learning via Sparse Coding

input Labeled training set
$T = \{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), \ldots, (x_l^{(m)}, y^{(m)})\}$.
Unlabeled data $\{x_u^{(1)}, x_u^{(2)}, \ldots, x_u^{(k)}\}$.

output Learned classifier for the classification task.

algorithm Using unlabeled data $\{x_u^{(i)}\}$, solve the optimization problem (1) to obtain bases b.
Compute features for the classification task to obtain a new labeled training set $\hat{T} = \{(\hat{a}(x_l^{(i)}), y^{(i)})\}_{i=1}^m$, where
$\hat{a}(x_l^{(i)}) = \arg\min_{a^{(i)}} \|x_l^{(i)} - \sum_j a_j^{(i)} b_j\|_2^2 + \beta\, \|a^{(i)}\|_1$.
Learn a classifier $\mathcal{C}$ by applying a supervised learning algorithm (e.g., SVM) to the labeled training set $\hat{T}$.
return the learned classifier $\mathcal{C}$.

3.3. Comparison with Other Methods

It seems that any algorithm for the self-taught learning problem must, at some abstract level, detect structure using the unlabeled data. Many unsupervised learning algorithms have been devised to model different aspects of "higher-level" structure; however, their application to self-taught learning is more challenging than might be apparent at first blush.

Principal component analysis (PCA) is among the most commonly used unsupervised learning algorithms. It identifies a low-dimensional subspace of maximal variation within unlabeled data. Interestingly, the top $T \leq n$ principal components $b_1, b_2, \ldots, b_T$ are a solution to an optimization problem that is cosmetically similar to our formulation in (1):

$$\begin{aligned} \text{minimize}_{b,a} \quad & \textstyle\sum_i \|x_u^{(i)} - \sum_j a_j^{(i)} b_j\|_2^2 \\ \text{s.t.} \quad & b_1, b_2, \ldots, b_T \text{ are orthogonal} \end{aligned} \quad (3)$$

PCA is convenient because the above optimization problem can be solved efficiently using standard numerical software; further, the features $a_j^{(i)}$ can be computed easily because of the orthogonality constraint, and are simply $a_j^{(i)} = b_j^T x_u^{(i)}$.

When compared with sparse coding as a method for constructing self-taught learning features, PCA has two limitations. First, PCA results in *linear* feature extraction, in that the features $a_j^{(i)}$ are simply a linear function of the input.[3] Second, since PCA assumes

[3]As an example of a nonlinear but useful feature for im-

Table 1. Details of self-taught learning applications evaluated in the experiments.

Domain	Unlabeled data	Labeled data	Classes	Raw features
Image classification	10 images of outdoor scenes	Caltech101 image classification dataset	101	Intensities in 14x14 pixel patch
Handwritten character recognition	Handwritten digits ("0"–"9")	Handwritten English characters ("a"–"z")	26	Intensities in 28x28 pixel character/digit image
Font character recognition	Handwritten English characters ("a"–"z")	Font characters ("a"/"A" – "z"/"Z")	26	Intensities in 28x28 pixel character image
Song genre classification	Song snippets from 10 genres	Song snippets from 7 *different* genres	7	Log-frequency spectrogram over 50ms time windows
Webpage classification	100,000 news articles (Reuters newswire)	Categorized webpages (from DMOZ hierarchy)	2	Bag-of-words with 500 word vocabulary
UseNet article classification	100,000 news articles (Reuters newswire)	Categorized UseNet posts (from "SRAA" dataset)	2	Bag-of-words with 377 word vocabulary

Table 2. Classification accuracy on the Caltech 101 image classification dataset. For PCA and sparse coding results, each image was split into the specified number of regions, and features were aggregated *within* each region by taking the maximum absolute value.

Features	Number of regions			
	1	4	9	16
PCA	20.1%	30.6%	36.8%	37.0%
Sparse coding	30.8%	40.9%	46.0%	**46.6%**
Published baseline (Fei-Fei et al., 2004)				16%

the bases b_j to be orthogonal, the number of PCA features cannot be greater than the dimension n of the input. Sparse coding does not have either of these limitations. Its features $\hat{a}(x)$ are an inherently nonlinear function of the input x, due to the presence of the L_1 term in Equation (1).[4] Further, sparse coding can use more basis vectors/features than the input dimension n. By learning a large number of basis vectors but using only a small number of them for any particular input, sparse coding gives a higher-level representation in terms of the many possible "basic patterns," such as edges, that may appear in an input. Section 6 further discusses other unsupervised learning algorithms.

4. Experiments

We apply our algorithm to several self-taught learning tasks shown in Table 3.3. Note that the unlabeled data in each case cannot be assigned the labels from the labeled task. For each application, the raw input examples x were represented in a standard way: raw pixel intensities for images, the frequency spectrogram for audio, and the bag-of-words (vector) representation for text. For computational reasons, the unlabeled data was preprocessed by applying PCA to reduce its dimension;[5] the sparse coding basis learning algorithm was then applied in the resulting principal component space.[6] Then, the learned bases were used to construct features for each input from the supervised classification task.[7] For each such task, we report the result from the better of two standard, off-the-shelf supervised learning algorithms: a support vector machine (SVM) and Gaussian discriminant analysis (GDA). (A classifier specifically customized to sparse coding features is described in Section 5.)

We compare our self-taught learning algorithm against two baselines, also trained with an SVM or GDA: using the raw inputs themselves as features, and using principal component projections as features, where the principal components were computed on the unlabeled

ages, consider the phenomenon called *end-stopping* (which is known to occur in biological visual perception) in which a feature is maximally activated by edges of only a specific orientation and length; increasing the length of the edge further significantly decreases the feature's activation. A linear response model cannot exhibit end-stopping.

[4]For example, sparse coding can exhibit end-stopping (Lee et al., 2007). Note also that even though sparse coding attempts to express x as a *linear* combination of the bases b_j, the optimization problem (2) results in the activations a_j being a *non-linear* function of x.

[5]We picked the number of principal components to preserve approximately 96% of the unlabeled data variance.

[6]Reasonable bases can often be learned even using a smooth approximation such as $(\sum_j \sqrt{a_j^2+\epsilon})$ to the L_1-norm sparsity penalty $\|a\|_1$. However, such approximations do *not* produce sparse features, and in our experiments, we found that classification performance is significantly worse if such approximations are used to compute $\hat{a}(x)$. Since the labeled and unlabeled data can sometimes lead to very different numbers of non-zero coefficients a_i, in our experiments β was also recalibrated prior to computing the labeled data's representations $\hat{a}(x_l)$.

[7]Partly for scaling and computational reasons, an additional feature aggregation step was applied to the image and audio classification tasks (since a single image is several times larger than the individual/small image patch bases that can be learned tractably by sparse coding). We aggregated features for the large image by extracting features for small image patches in different locations in the large image, and then aggregating the features per-basis by taking the feature value with the maximum absolute value. The aggregation procedure effectively looks for the "strongest" occurrence of each basis pattern within the image. (Even better performance is obtained by aggregating features over a KxK grid of regions, thus looking for strong activations separately in different parts of the large image; see Table 3.3.) These region-wise aggregated features were used as input to the classification algorithms (SVM or GDA). Features for audio snippets were similarly aggregated by computing the maximum activation per basis vector over 50ms windows in the snippet.

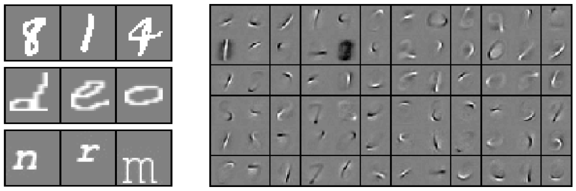

Figure 5. Left: Example images from the handwritten digit dataset (top), the handwritten character dataset (middle) and the font character dataset (bottom). Right: Example sparse coding bases learned on handwritten digits.

Table 3. Top: Classification accuracy on 26-way handwritten English character classification, using bases trained on handwritten digits. Bottom: Classification accuracy on 26-way English font character classification, using bases trained on English handwritten characters. The numbers in parentheses denote the accuracy using raw and sparse coding features together. Here, sparse coding features alone do not perform as well as the raw features, but perform significantly better when used in combination with the raw features.

Digits → English handwritten characters			
Training set size	Raw	PCA	Sparse coding
100	**39.8%**	25.3%	39.7%
500	54.8%	54.8%	**58.5%**
1000	61.9%	64.5%	**65.3%**

Handwritten characters → Font characters			
Training set size	Raw	PCA	Sparse coding
100	8.2%	5.7%	7.0% (**9.2%**)
500	17.9%	14.5%	16.6% (**20.2%**)
1000	25.6%	23.7%	23.2% (**28.3%**)

data (as described in Section 3.3). In the PCA results presented in this paper, the number of principal components used was always fixed at the number of principal components used for preprocessing the raw input before applying sparse coding. This control experiment allows us to evaluate the effects of PCA preprocessing and the later sparse coding step separately, but should therefore not be treated as a direct evaluation of PCA as a self-taught learning algorithm (where the number of principal components could then also be varied).

Tables 3.3-4 report the results for various domains. Sparse coding features, possibly in combination with raw features, significantly outperform the raw features alone as well as PCA features on most of the domains.

On the 101-way Caltech 101 image classification task with 15 training images per class (Table 3.3), sparse coding features achieve a test accuracy of 46.6%. In comparison, the first published supervised learning algorithm for this dataset achieved only 16% test accuracy even with computer vision specific features (instead of raw pixel intensities).[8]

Table 4. Accuracy on 7-way music genre classification.

Training set size	Raw	PCA	Sparse coding
100	28.3%	28.6%	**44.0%**
1000	34.0%	26.3%	**45.5%**
5000	38.1%	38.1%	**44.3%**

Table 5. Text bases learned on 100,000 Reuters newswire documents. Top: Each row represents the basis most active on average for documents with the class label at the left. For each basis vector, the words corresponding to the largest magnitude elements are displayed. Bottom: Each row represents the basis that contains the largest magnitude element for the word at the left. The words corresponding to other large magnitude elements are displayed.

Design	design, company, product, work, market
Business	car, sale, vehicle, motor, market, import
vaccine	infect, report, virus, hiv, decline, product
movie	share, disney, abc, release, office, movie, pay

Figure 5 shows example inputs from the three character datasets, and some of the learned bases. The learned bases appear to represent "pen strokes." In Table 4, it is thus not surprising that sparse coding is able to use bases ("strokes") learned on digits to significantly improve performance on handwritten characters—it allows the supervised learning algorithm to "see" the characters as comprising strokes, rather than as comprising pixels.

For audio classification, our algorithm outperforms the original (spectral) features (Table 4).[9] When applied to text, sparse coding discovers word relations that might be useful for classification (Table 5). The performance improvement over raw features is small (Table 4).[10] This might be because the bag-of-words representation of text documents is already sparse, unlike the raw inputs for the other applications.[11]

We envision self-taught learning as being most useful when labeled data is scarce. Table 4 shows that with small amounts of labeled data, classification performance deteriorates significantly when the bases (in sparse coding) or principal components (in PCA) are

[8]Since the time we ran our experiments, other researchers have reported better results using *highly specialized* computer vision algorithms (Zhang et al., 2006: 59.1%; Lazebnik et al., 2006: 56.4%). We note that our algorithm was until recently state-of-the-art for this well-known dataset, even with almost no explicit computer-vision engineering, and indeed it significantly outperforms many carefully hand-designed, computer-vision specific methods published on this task (E.g., Fei-Fei et al., 2004: 16%; Serre et al., 2005: 35%; Holub et al., 2005: 40.1%).

[9]Details: We learned bases over songs from 10 genres, and used these bases to construct features for a music genre classification over songs from 7 *different* genres (with different artists, and possibly different instruments). Each training example comprised a labeled 50ms song snippet; each test example was a 1 second song snippet.

[10]Details: Learned bases were evaluated on 30 binary webpage category classification tasks. PCA applied to text documents is commonly referred to as latent semantic analysis. (Deerwester et al., 1990)

[11]The results suggest that algorithms such as LDA (Blei et al., 2002) might also be appropriate for self-taught learning on text (though LDA is specific to a bag-of-words representation and would not apply to the other domains).

Table 6. Classification accuracy on webpage classification (top) and UseNet article classification (bottom), using bases trained on Reuters news articles.

Reuters news → Webpages			
Training set size	Raw	PCA	Sparse coding
4	62.8%	63.3%	**64.3%**
10	73.0%	72.9%	**75.9%**
20	79.9%	78.6%	**80.4%**

Reuters news → UseNet articles			
Training set size	Raw	PCA	Sparse coding
4	61.3%	60.7%	**63.8%**
10	**69.8%**	64.6%	68.7%

Table 7. Accuracy on the self-taught learning tasks when sparse coding bases are learned on unlabeled data (third column), or when principal components/sparse coding bases are learned on the labeled training set (fourth/fifth column). Since Tables 3.3-4 already show the results for PCA trained on unlabeled data, we omit those results from this table. The performance trends are qualitatively preserved even when raw features are appended to the sparse coding features.

Domain	Training set size	Unlabeled SC	Labeled PCA	Labeled SC
Handwritten characters	100	**39.7%**	36.2%	31.4%
	500	**58.5%**	50.4%	50.8%
	1000	**65.3%**	62.5%	61.3%
	5000	73.1%	**73.5%**	73.0%
Font characters	100	**7.0%**	5.2%	5.1%
	500	**16.6%**	11.7%	14.7%
	1000	**23.2%**	19.0%	22.3%
Webpages	4	**64.3%**	55.9%	53.6%
	10	**75.9%**	57.0%	54.8%
	20	**80.4%**	62.9%	60.5%
UseNet	4	**63.8%**	60.5%	50.9%
	10	**68.7%**	67.9%	60.8%

learned on the labeled data itself, instead of on large amounts of additional unlabeled data.[12] As more and more labeled data becomes available, the performance of sparse coding trained on labeled data approaches (and presumably will ultimately exceed) that of sparse coding trained on unlabeled data.

Self-taught learning empirically leads to significant gains in a large variety of domains. An important theoretical question is characterizing how the "similarity" between the unlabeled and labeled data affects the self-taught learning performance (similar to the analysis by Baxter, 1997, for transfer learning). We leave this question open for further research.

5. Learning a Kernel via Sparse Coding

A fundamental problem in supervised classification is defining a "similarity" function between two input examples. In the experiments described above, we used the regular notions of similarity (i.e., standard SVM kernels) to allow a fair comparison with the baseline algorithms. However, we now show that the sparse coding model also suggests a specific specialized similarity function (kernel) for the learned representations.

The sparse coding model (1) can be viewed as learning the parameter b of the following linear generative model, that posits Gaussian noise on the observations x and a Laplacian (L_1) prior over the activations:

$$\begin{aligned} P(x = \textstyle\sum_j a_j b_j + \eta \mid b, a) &\propto \exp(-\|\eta\|_2^2/2\sigma^2), \\ P(a) &\propto \exp(-\beta \textstyle\sum_j |a_j|) \end{aligned}$$

Once the bases b have been learned using unlabeled data, we obtain a complete generative model for the input x. Thus, we can compute the Fisher kernel to measure the similarity between new inputs. (Jaakkola & Haussler, 1998) In detail, given an input x, we first compute the corresponding features $\hat{a}(x)$ by efficiently solving optimization problem (2). Then, the Fisher kernel implicitly maps the input x to the high-dimensional feature vector $U_x = \nabla_b \log P(x, \hat{a}(x)|b)$, where we have used the MAP approximation $\hat{a}(x)$ for the random variable a.[13] Importantly, for the sparse coding generative model, the corresponding kernel has a particularly intuitive form, and for inputs $x^{(s)}$ and $x^{(t)}$ can be computed efficiently as:

$$K(x^{(s)}, x^{(t)}) = \left(\hat{a}(x^{(s)})^T \hat{a}(x^{(t)})\right) \cdot \left({r^{(s)}}^T r^{(t)}\right),$$

where $r = x - \sum_j \hat{a}_j b_j$ represents the residual vector corresponding to the MAP features $\hat{a}$. Note that the first term in the product above is simply the inner product of the MAP feature vectors, and corresponds to using a linear kernel over the learned sparse representation. The second term, however, compares the two residuals as well.

We evaluate the performance of the learned kernel on the handwritten character recognition domain, since it does not require any feature aggregation. As a baseline, we compare against all choices of standard kernels (linear/polynomials of degree 2 and 3/RBF) and features (raw features/PCA/sparse coding features). Table 5 shows that an SVM with the new kernel outperforms the *best* choice of standard kernels and features, even when that best combination was picked on the test data (thus giving the baseline a slightly unfair advantage). In summary, using the Fisher kernel derived from the generative model described above, we obtain a classifier customized specifically to the distribution of sparse coding features.

6. Discussion and Other Methods

In the semi-supervised learning setting, several authors have previously constructed features using data from the *same* domain as the labeled data (e.g., Hinton & Salakhutdinov, 2006). In contrast, self-taught learning

[12] For the sake of simplicity (and due to space constraints), we performed this comparison only for the domains that the basic sparse coding algorithm applies to, and that do not require the extra feature aggregation step.

[13] In our experiments, the marginalized kernel (Tsuda et al., 2002), that takes an expectation over a (computed by MCMC sampling) rather than the MAP approximation, did not perform better.

Table 8. Classification accuracy using the learned sparse coding kernel in the Handwritten Characters domain, compared with the accuracy using the *best* choice of standard kernel and input features. (See text for details.)

Training set size	Standard kernel	Sparse coding
100	35.4%	**41.8%**
500	54.8%	**62.6%**
1000	61.9%	**68.9%**

poses a harder problem, and requires that the structure learned from unlabeled data be "useful" for representing data from the classification task. Several existing methods for unsupervised and semi-supervised learning can be applied to self-taught learning, though many of them do not lead to good performance. For example, consider the task of classifying images of English characters ("a"—"z"), using unlabeled images of digits ("0"—"9"). For such a task, manifold learning algorithms such as ISOMAP (Tenenbaum et al., 2000) or LLE (Roweis & Saul, 2000) can learn a low-dimensional manifold using the unlabeled data (digits); however, these manifold representations do not generalize straightforwardly to the labeled inputs (English characters) that are dissimilar to any single unlabeled input (digit). We believe that these and several other learning algorithms such as auto-encoders (Hinton & Salakhutdinov, 2006) or non-negative matrix factorization (Hoyer, 2004) might be modified to make them suitable for self-taught learning.

We note that even though semi-supervised learning was originally defined with the assumption that the unlabeled and labeled data follow the same class labels (Nigam et al., 2000), it is sometimes conceived as "learning with labeled and unlabeled data." Under this broader definition of semi-supervised learning, self-taught learning would be an instance (a particularly widely applicable one) of it.

Examining the last two decades of progress in machine learning, we believe that the self-taught learning framework introduced here represents the natural extrapolation of a sequence of machine learning problem formalisms posed by various authors—starting from purely supervised learning, through semi-supervised learning, to transfer learning—where researchers have considered problems making increasingly little use of expensive labeled data, and using less and less related data. In this light, self-taught learning can also be described as "unsupervised transfer" or "transfer learning from unlabeled data." Most notably, Ando & Zhang (2005) propose a method for transfer learning that relies on using hand-picked heuristics to generate labeled secondary prediction problems from unlabeled data. It might be possible to adapt their method to several self-taught learning applications. It is encouraging that our simple algorithms produce good results across a broad spectrum of domains. With this paper, we hope to initiate further research in this area.

Acknowledgments

We give warm thanks to Bruno Olshausen, Geoff Hinton and the anonymous reviewers for helpful comments. This work was supported by the DARPA transfer learning program under contract number FA8750-05-2-0249, and by ONR under award number N00014-06-1-0828.

References

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, *6*, 1817–1853.

Baxter, J. (1997). Theoretical models of learning to learn. In T. Mitchell and S. Thrun (Eds.), *Learning to learn.*

Blei, D., Ng, A. Y., & Jordan, M. (2002). Latent dirichlet allocation. *NIPS*.

Caruana, R. (1997). Multitask learning. *ML Journal*, *28.*

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *J. Am. Soc. Info. Sci.*, *41*, 391–407.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Ann. Stat.*, *32*, 407–499.

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. *CVPR Workshop on Gen.-Model Based Vision.*

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*, 504–507.

Holub, A., Welling, M., & Perona, P. (2005). Combining generative models and Fisher kernels for object class recognition. *ICCV*.

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *JMLR*, *5*, 1457–1469.

Jaakkola, T., & Haussler, D. (1998). Exploiting generative models in discriminative classifiers. *NIPS*.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *CVPR*.

Lee, H., Battle, A., Raina, R., & Ng, A. Y. (2007). Efficient sparse coding algorithms. *NIPS*.

Ng, A. Y. (2004). Feature selection, L_1 vs. L_2 regularization, and rotational invariance. *ICML*.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*, 103–134.

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

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290.*

Serre, T., Wolf, L., & Poggio, T. (2005). Object recognition with features inspired by visual cortex. *CVPR*.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? *NIPS*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. B.*, *58*, 267–288.

Tsuda, K., Kin, T., & Asai, K. (2002). Marginalized kernels for biological sequences. *Bioinformatics*, *18.*

Zhang, H., Berg, A., Maire, M., & Malik, J. (2006). SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. *CVPR*.