Exponential Family Sparse Coding with Applications to Self-taught Learning

Honglak Lee **Rajat Raina** **Alex Teichman** **Andrew Y. Ng**
Computer Science Department
Stanford University
{ hllee, rajatr, teichman, ang }@cs.stanford.edu

Abstract

Sparse coding is an unsupervised learning algorithm for finding concise, slightly higher-level representations of inputs, and has been successfully applied to self-taught learning, where the goal is to use unlabeled data to help on a supervised learning task, even if the unlabeled data cannot be associated with the labels of the supervised task [Raina *et al.*, 2007]. However, sparse coding uses a Gaussian noise model and a quadratic loss function, and thus performs poorly if applied to binary valued, integer valued, or other non-Gaussian data, such as text. Drawing on ideas from generalized linear models (GLMs), we present a generalization of sparse coding to learning with data drawn from any exponential family distribution (such as Bernoulli, Poisson, etc). This gives a method that we argue is much better suited to model other data types than Gaussian. We present an algorithm for solving the L_1-regularized optimization problem defined by this model, and show that it is especially efficient when the optimal solution is sparse. We also show that the new model results in significantly improved self-taught learning performance when applied to text classification and to a robotic perception task.

1 Introduction

Sparse coding is an unsupervised learning algorithm that attempts to learn a concise, slightly higher-level representation using large amounts of unlabeled data [Olshausen and Field, 1996]. Though originally inspired by mammalian perception, the algorithm has recently been applied successfully to machine learning applications where labeled data is scarce.

We consider the "self-taught learning" problem, in which we are given limited labeled data from a classification task, and also large amounts of unlabeled data that is only mildly related to the task [Weston *et al.*, 2006; Raina *et al.*, 2007]. Specifically, the unlabeled data may not share the class labels or arise from the same distribution. Raina et al. show that with a specific sparse coding model, such mildly related unlabeled data can help improve classification accuracy on some tasks. However, their sparse coding model assumes that the inputs consist of real-valued vectors, and that the vectors can be well described using a Gaussian noise model (details in Section 2). In our view, and as demonstrated by our experiments, this severely limits the applicability of sparse coding in a general self-taught learning algorithm.

As a running example, consider the application of self-taught learning to text classification: suppose we would like to classify sports webpages as "Baseball" or "Football" using only very few labeled webpages and many unlabeled text documents, obtained randomly from the Internet (say). The natural representation of text documents is often as a binary "bag-of-words" vector $x \in \{0,1\}^k$, where the i-th feature is 1 if the i-th word in our vocabulary occurred in the document, or as a word-counts vector $x \in \{0,1,2,\dots\}^k$, where the i-th feature represents the number of times the i-th word occurred in the document. In either case, such input vectors are very poorly modeled by a continuous Gaussian distribution (which could take fractional, or negative values). It is thus hardly surprising that when sparse coding is applied to a self-taught learning task involving text data, it only leads to very small improvements in accuracy.

The above problem is not unique to text classification. Sparse coding with the Gaussian noise distribution assumption may be too restrictive to model the wide variety of inputs that we might encounter in machine learning problems, including point clouds or depth maps, discrete data, etc.

To address this problem, we generalize the Gaussian probabilistic model behind sparse coding in a principled way to include most standard distributions. We draw on the widely studied idea of the "exponential family" of distributions. This class of distributions includes the Gaussian, Bernoulli and Poisson distribution, among others, while still providing guarantees useful for efficient learning and inference. Our generalization is analogous to the way in which generalized linear models (GLMs) generalize least squares regression (which relies on a Gaussian assumption) to other kinds of regression, including logistic regression (for binary inputs) or softmax regression (for multivalued inputs) [McCullagh and Nelder, 1989]. We call our model *exponential family sparse coding*, and to differentiate it from the previous model, we henceforth call that model Gaussian sparse coding.

Our generalization makes the parameter learning problem significantly harder. However, we show that the optimization problem can be solved via a series of L_1-regularized least squares problems, each of which can be solved using algorithms similar to the ones used for Gaussian sparse coding. In

fact, our optimization procedure can also be applied to other L_1-regularized optimization problems, and is especially efficient for problems that have very sparse optimal solutions.

We apply our model successfully to two self-taught learning problems—text classification and a robotic perception task—even though Gaussian sparse coding produces poor performance for both.

2 Preliminaries

We consider a classification problem with a small labeled training set $\{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), \ldots, (x_l^{(m)}, y^{(m)})\}$ drawn i.i.d. from an unknown distribution $\mathcal{D}$. Each input $x_l^{(i)} \in \mathcal{X}$ is assigned a class label $y^{(i)} \in \mathcal{Y} = \{1, 2, \ldots, K\}$. We do not place the additional restriction that $\mathcal{X} = \mathbb{R}^k$. For example, text documents can be represented as a binary vector $x_l^{(i)} \in \mathcal{X} = \{0,1\}^k$ or as an integer-valued vector $x_l^{(i)} \in \mathcal{X} = \{0, 1, 2, \ldots\}^k$.

Following the self-taught learning framework, we assume that we are also given a large set of unlabeled examples $\{x^{(1)}, x^{(2)}, \ldots, x^{(r)}\}$ (without the subscript "l"). The inputs $x^{(i)} \in \mathcal{X}$ are only mildly constrained to be of the same "input type" as the labeled examples, but need not belong to any of the labels in the classification task, and need not arise from the same distribution $\mathcal{D}$.

2.1 Gaussian sparse coding

As studied in the literature, Gaussian sparse coding uses unlabeled input vectors $x \in \mathbb{R}^k$ to discover *basis vectors* $b_1, b_2, \ldots, b_n \in \mathbb{R}^k$ such that any input x can be represented approximately as a weighted linear combination of only a small number of the basis vectors: $x \approx \sum_j b_j s_j$. Here, the weights $s \in \mathbb{R}^n$ are called the *activations* corresponding to the input x, and are encouraged to be sparse (i.e., to have many of the activations exactly equal to zero).

In detail, Gaussian sparse coding assumes that the input vector x is generated from a Gaussian distribution with mean $\eta = \sum_j b_j s_j$ and (known) covariance $\sigma^2 I$. The activation vector s is assumed to follow a Laplacian prior $P(s) \propto \prod_j \exp(-\beta|s_j|)$ for some constant β. Given unlabeled examples $\{x^{(1)}, x^{(2)}, \ldots, x^{(r)}\}$, the maximum-a-posteriori estimate of the corresponding activations $\{s^{(1)}, \ldots, s^{(r)}\}$ and the basis vectors $\{b_j\}$ is obtained by solving:

$$\max_{\{b_j\}, \{s^{(i)}\}} \prod_i \; P\left(x^{(i)} | \{b_j\}, \{s^{(i)}\}\right) \; P\left(s^{(i)}\right) \tag{1}$$

This reduces to the following optimization problem:[1]

$$\begin{aligned} \min_{\{b_j\}, \{s^{(i)}\}} \quad & \tfrac{1}{2\sigma^2} \textstyle\sum_i \|x^{(i)} - \sum_{j=1}^n b_j s_j^{(i)}\|^2 + \beta \sum_{i,j} |s_j^{(i)}| \\ \text{subj. to} \quad & \|b_j\|^2 \le c, \; \forall j = 1, ..., n. \end{aligned} \tag{2}$$

This optimization problem is convex in the basis vectors b or the activations s alone (though not jointly convex in both). As the objective function is simply an L_1-regularized quadratic function, an efficient alternating minimization procedure can be devised for this problem [Lee *et al.*, 2007].

Sparse coding forces each input $x^{(i)}$ to be "explained" using a small number of basis vectors, and the activations often capture certain higher-level features of the input. For example, when the inputs $x^{(i)}$ consist of small images (represented as vectors of pixel intensities), the basis vectors learnt using Equation (2) capture various kinds of edges. Effectively, the procedure converts the pixel-based representation $x^{(i)}$ to a succinct, slightly higher-level, edge-based representation $s^{(i)}$. Thus, Raina et al. proposed the following simple self-taught learning algorithm:

1. Use the unlabeled data to learn basis vectors $\{b_j\}$ by solving the problem in Equation (2).
2. Now fix the basis vectors, and compute the "activations" for labeled examples x_l by computing $s^* = \arg\min_s \frac{1}{2\sigma^2}\|x_l - \sum_j b_j s_j\|^2 + \beta \sum_j |s_j|$. This is an L_1-regularized least squares problem, and can be solved efficiently [Efron *et al.*, 2004; Lee *et al.*, 2007].
3. Finally, use the activations s^* as features to train a standard, off-the-shelf classifier (such as an SVM) using the labeled data. The classifier can then be applied to label test data.

2.2 Self-taught Learning for Discrete Inputs

To motivate the algorithms introduced in this paper, consider the application of the above self-taught learning algorithm to binary input vectors $x \in \{0,1\}^k$, say for text classification. The Gaussian sparse coding model makes the probabilistic assumption that $P(x \,|\, \eta = \sum_j b_j s_j)$ is a Gaussian distribution, which is a poor fit to binary data. Stated otherwise, the Gaussian sparse coding model tries to enforce the decomposition $x \approx \sum_j b_j s_j$, even though the unconstrained sum $\sum_j b_j s_j$ is a particularly poor approximation to a binary vector x. Thus, a straightforward application of Gaussian sparse coding does not lead to very useful basis vectors or features.

Instead, we might want to find an approximation of the form $x \approx \sigma(\sum_j b_j s_j)$, where $\sigma(v) = [\frac{1}{1+e^{-v_1}}, \frac{1}{1+e^{-v_2}}, \ldots]$ represents the elementwise logistic function for a vector v. This promises to be a better formulation, since the logistic function always lies in $(0, 1)$.

We now present a systematic generalization of the Gaussian sparse coding model. Our generalization includes both Gaussian sparse coding and our seemingly arbitrary logistic function approximation as special cases, and will directly suggest models for other input types, such as when the inputs consist of nonnegative integer counts $x \in \{0, 1, 2, \ldots\}^k$.

3 Exponential Family Sparse Coding

The exponential family is a widely used class of distributions in statistics, and in its most general form, is represented as: $P(x|\eta) = h(x) \exp(\eta^\top T(x) - a(\eta))$. Here, η represents the natural parameter for the model, and the functions h, T and a together define a particular member of the family. For example, it is easy to verify that the multivariate Gaussian distribution $\mathcal{N}(\mu, I)$ with fixed covariance I and (unknown) mean

[1] Following Lee et al., we use the norm constraint on the bases to disallow degenerate solutions in which the activations can be scaled down as long as the basis vectors are scaled up by the same number.

parameter $\mu \in \mathbb{R}^k$ is equivalent to the exponential family distribution defined by $h(x) = e^{-\|x\|^2/2}/(2\pi)^{k/2}$, $T(x) = x$ and $a(\eta) = \eta^\top \eta/2$ (with natural parameter $\eta = \mu$). The exponential family of distributions is broad enough to include most standard distributions (Gaussian, Bernoulli, Poisson, exponential, and others), but also restrictive enough to provide useful guarantees. Importantly, it guarantees that the log-likelihood is concave in the natural parameter η, making maximum likelihood learning of parameters tractable [McCullagh and Nelder, 1989].

We modify the Gaussian sparse coding model to allow any distribution from the exponential family:

$$P(x|b,s) = h(x)\exp(\eta^\top T(x) - a(\eta)), \quad \eta = \textstyle\sum_j b_j s_j \quad (3)$$

where we use the basis vectors b_j and the activations s_j to construct the natural parameter η for the family. Since the Gaussian distribution is a member of the exponential family, our new generative model includes the earlier Gaussian generative model for $P(x|\eta)$ as a special case. In fact, our model extends Gaussian sparse coding in the same way that generalized linear models (GLMs) extend the notion of regression with least squares loss to other loss functions, including logistic regression and softmax regression as special cases.

Given unlabeled examples $\{x^{(1)}, x^{(2)}, \ldots, x^{(r)}\}$, we can apply Equation (1) to compute the maximum-a-posteriori estimates of the basis vectors b_j and the activations $s^{(i)}$:[2]

$$\min_{B,\{s^{(i)}\}} \quad \textstyle\sum_i -\log h(x^{(i)}) - {s^{(i)}}^\top B^\top T(x^{(i)}) + a(Bs^{(i)})$$

$$+ \beta \textstyle\sum_{i,j} |s_j^{(i)}| \quad (4)$$

$$\text{subj. to} \qquad \|b_j\|^2 \le c, \ \forall j = 1, ..., n, \quad (5)$$

where we define the basis matrix B such that its j-th column is the basis vector b_j, implying that $\eta = \sum_j b_j s_j = Bs$. By definition, we want the model to produce sparse activations, so we set β large enough to produce only a small number of nonzero values per activation vector $s^{(i)}$ on average.

Since the exponential family guarantees convexity of $\log P(x|\eta)$ with respect to η, we can show that the above optimization problem is convex with respect to s for fixed B, and with respect to B for fixed s (though it is not jointly convex). This again suggests an alternating minimization procedure iterating the following two steps till convergence: (i) fix the activations s, and compute the optimal bases B; and, (ii) fix these bases B, and compute the optimal activations s.

Step (i) involves a constrained optimization problem over B with a differentiable objective function. We can thus apply projective gradient descent updates, where at each iteration we perform a line search along the direction of the (negative) gradient, projecting onto the feasible set before evaluating the objective function during the line search. In our case, the projection operation is especially simple: we just need to rescale each basis vector to have norm c if its norm is greater than c. In our experiments, we find that such a projective gradient descent scheme is sufficiently fast for basis learning. We thus focus now on the algorithm for computing the optimal activations in Step (ii).

Step (ii) computes the optimal activation s given fixed basis vectors. The resulting problem involves a non-differentiable L_1-regularized objective function, to which straightforward gradient descent methods are not applicable. Recently, many sophisticated algorithms have been developed for L_1-regularized optimization, including specialized interior point methods [Koh *et al.*, 2007], quasi-Newton methods [Andrew and Gao, 2007; Yu *et al.*, 2008] and coordinate descent methods [Friedman *et al.*, 2007]. When used for computing activations with 1000 basis vectors, these methods find the optimal solution in a few seconds *per unlabeled example*. Since we often need to solve for the activations of tens of thousands of unlabeled examples repeatedly in the inner loop of the overall alternating minimization procedure, these solvers turn out to be too slow for our model.

We now present a simple optimization algorithm for L_1-regularized optimization problems. We later show that, despite the simplicity of the algorithm, when the optimal solution is very sparse (as in Equation 4-5) our method is faster than state-of-the-art solvers for L_1-regularized problems.

3.1 Computing optimal activations

We first note that since the optimal values for the activation vectors $s^{(i)}$ do not depend on each other, and can be optimized separately, it is sufficient to consider the following optimization problem for a single input x and its activation s:

$$\min_s \quad \ell(s) + \beta\|s\|_1 \quad (6)$$

where s corresponds to a vector of activations, and $\ell(s)$ is a specific convex function of s.

In the case of Gaussian sparse coding, $\ell(s)$ is simply a quadratic function, and the optimization problem is a L_1-regularized least squares problem that can be solved efficiently [Efron *et al.*, 2004; Lee *et al.*, 2007]. This suggests an iterative algorithm for the general case: at each iteration, we compute a local quadratic approximation $\hat{\ell}(s)$ to the function $\ell(s)$, and optimize the objective function $\hat{\ell}(s) + \beta\|s\|_1$ instead.[3] Using this insight, Lee et al. proposed the IRLS-LARS algorithm for the case of L_1-regularized logistic regression, using Efron et al.'s LARS algorithm in the inner loop to solve the approximated problem.

This method can be applied to other L_1-regularized optimization problems for which a local quadratic approximation can be efficiently computed. Indeed, for the case of the L_1-regularized exponential family in Equation (4-5), we can show that the local quadratic approximation at a point s is:

$$\hat{\ell}(s) \quad = \quad \left\|\Lambda^{1/2}Bs' - \Lambda^{1/2}z\right\|^2 \quad (7)$$

where $\Lambda_{ii} = a''((Bs)_i)$ for a diagonal matrix Λ, and $z = \Lambda^{-1}(T(x) - a'(Bs)) + Bs$.[4]

We note that if the objective function $\ell(s)$ is reasonably approximated by a quadratic function, the solutions to the

[2] As in Gaussian sparse coding, we assume a Laplacian prior on s: $P(s) \propto \prod_j \exp(-\beta|s_j|)$, and a uniform prior on b_j.

[3] This method is an instance of a general method called Iteratively Reweighted Least Squares (IRLS) in the literature [Green, 1984].

[4] Proof Sketch: ℓ and $\hat{\ell}$ have the same gradient and Hessian at s: $\nabla\ell = \nabla\hat{\ell} = -B^\top T(x) + B^\top a'(Bs)$, and $\nabla^2\ell = \nabla^2\hat{\ell} = B^\top \Lambda B$.

Dataset	Small1	Small2	Small3	Med1	Med2	Med3	Large1	Large2	Large3
IRLS-LARS	4.6	4.9	4.3	12.8	12.5	13.2	1131	1270	1214
l1-logreg	18.3	18.9	17.7	181	188	185	3277	3101	3013
CoordDescent	3.6	3.4	5.6	20.7	20.7	31.0	787	653	723
OWL-QN	7.1	7.0	10.3	27.1	31.4	25.6	1018	739	906
SubLBFGS	33.0	22.3	23.1	101	142	57.2	1953	2717	1627
IRLS-FS	**2.5**	**2.3**	**2.2**	**5.3**	**5.5**	**5.4**	**117**	**108**	**109**

Table 1: Total running time in seconds for computing activations for 50 input examples for 9 problems (one per column). There are 3 problems each of 3 different sizes, and they are labeled Small1 to Small3, Med1 to Med3, or Large1 to Large3 based on the size of the problem. The "Small" problems had 200 basis vectors and input dimension 369, "Med" problems had 600 basis vectors and dimension 369, and "Large" problems had 1000 basis vectors and dimension 3891.

Dataset	Col	Alon	Duln	Duer	Arr	Mad	Hep	Spf	Prom	Wbc	Ion	Spl	Spc	Spam
Sparsity (% nonzero)	0.2	0.5	0.8	1.4	3.5	3.6	26.3	29.5	31.6	40.0	45.5	56.7	63.6	66.7
IRLS-LARS	2.1	3.3	6.2	35.6	2.2	25.6	0.5	5.0	2.1	5.0	3.5	18.3	2.6	57.8
l1-logreg	18.3	16.8	13.6	14.4	34.8	509	1.0	3.0	2.0	3.8	2.7	12.8	2.0	37.1
CoordDescent	83.3	54.1	63.8	129	7.7	101	0.2	**2.0**	0.6	**2.6**	**1.4**	**4.5**	**0.8**	**14.2**
OWL-QN	27.4	29.4	16.9	79.6	7.7	634	**0.1**	3.4	**0.4**	13.4	1.9	7.1	0.9	39.3
SubLBFGS	114	80.8	60.5	311	24.3	261	0.7	9.3	2.7	14.4	4.5	13.4	2.1	43.0
IRLS-FS	**1.9**	**1.9**	**2.5**	**7.1**	**1.5**	**14.0**	0.3	2.3	1.3	2.9	2.0	10.4	1.9	50.8

Table 2: Total running time in seconds for 50 trials of learning parameters of various L_1-regularized logistic regression benchmarks (obtained from Lee *et al.*, 2006). The datasets are ordered left-to-right by the increasing fraction of nonzero parameters at the optimal solution (e.g., the leftmost problem Col had only 0.2% nonzero parameters at the optimal solution).

successive quadratic approximations should be close to each other. However, the LARS algorithm used in IRLS-LARS cannot be initialized at an arbitrary point, and thus has to rediscover the solution from scratch while solving each successive approximation. In contrast, the "feature-sign search" algorithm (originally proposed in the context of Gaussian sparse coding) can be initialized at an arbitrary point [Lee *et al.*, 2007], and can thus potentially solve the successive approximations much faster. We propose to use the feature-sign search algorithm to optimize each quadratic approximation.

The final algorithm, which we call IRLS-FS, is described below. The algorithm is guaranteed to converge to the global optimum in a finite number of iterations. (Proof similar to IRLS-LARS.)

Algorithm 1: IRLS-FS algorithm for L_1-regularized exponential family problems

Input: $B \in \mathbb{R}^{k \times n}$, $x \in \mathbb{R}^k$, threshold ϵ. Initialize $s := \vec{0}$.
while decrease in objective value at last step $> \epsilon$ **do**
  Compute diagonal matrix Λ with $\Lambda_{ii} = a''((Bs)_i)$,
  Compute vector $z = \Lambda^{-1}(T(x) - a'(Bs)) + Bs$.
  Initializing feature-sign search at s, compute:
    $\hat{s} = \arg\min_{s'} \left\|\Lambda^{1/2}Bs' - \Lambda^{1/2}z\right\|^2 + \beta\|s'\|_1$
  Set $s := (1-t)s + t\hat{s}$, where t is found by backtracking line-search [Boyd and Vandenberghe, 2004] to minimize the objective function in Eqn (4).
end while

4 Computational Efficiency

We compare the IRLS-FS algorithm against state-of-the-art algorithms for optimizing the activations, focusing on the case of binary sparse coding (i.e., $x \in \{0,1\}^k$). This case is especially interesting because this leads to an L_1-regularized logistic regression problem, where the number of "features" is equal to the number of basis vectors used, but is *independent* of the dimensionality of inputs x in the original problem. This problem is of general interest (e.g., see Ng, 2004), and customized algorithms have also been developed for it.

We consider five recent algorithms: the IRLS-LARS algorithm [Lee *et al.*, 2006] and the l1-logreg interior point method [Koh *et al.*, 2007] specifically for logistic regression, the Coordinate Descent method [Friedman *et al.*, 2007] with successive IRLS approximations (constructed locally as in IRLS-FS), and the OWL-QN [Andrew and Gao, 2007] and SubLBFGS [Yu *et al.*, 2008] algorithms for L_1-regularized convex optimization problems.[5] All algorithms were evaluated on nine L_1-regularized logistic regression problems which arise in the course of solving Equations (4-5) for unlabeled text documents with binary sparse coding (details in Section 5.1). The sparsity parameter β was set to produce 20 nonzero activations per example on average.[6] We measured the running time taken by each algorithm to converge within a specified tolerance of the optimal objective value.[7]

[5]Baseline algorithms: Lee et al. show that IRLS-LARS outperforms several previous algorithms, including grafting [Perkins and Theiler, 2003], SCGIS [Goodman, 2004], GenLasso [Roth, 2004] and Gl1ce [Lokhorst, 1999]. IRLS-FS, IRLS-LARS, l1-logreg and CoordDescent were implemented in Matlab, and OWL-QN and SubLBFGS were compiled in C++ with optimization flags.

[6]In our experiments, such β values produced reasonable basis vectors during basis learning.

[7]Details: Since IRLS-LARS solves the dual or Lasso version of our problem (i.e., with a constraint C on the L_1 norm of the activations rather than a penalty β), we follow Lee et al.'s method of

Table 1 shows the running times computed over 50 trials. IRLS-FS outperforms the other algorithms on this task, showing that it well-suited to exponential family sparse coding. When a large number of basis vectors are used, IRLS-FS can be 5 to 7 times faster than the best baseline algorithm.

This poses the question: can IRLS-FS be a useful algorithm for general L_1-regularized optimization problems (not necessarily ones generated by the sparse coding problem)? We compare the algorithms above on 14 moderate-size benchmark classification datasets, and apply L_1-regularized logistic regression to them. The value of β on each benchmark was picked to optimize the generalization error of the resulting logistic regression classifier; unlike the earlier experiment, β was not set explicitly to obtain sparse solutions. Table 2 shows the running time of the algorithms to compute the optimal parameters. IRLS-FS outperforms the other algorithms on 6 out of 14 of these benchmark datasets; more specifically, it performs best when the optimal parameters have a very small number of nonzero values.

5 Application to self-taught learning

The model presented in this paper generalizes Gaussian sparse coding. It is also closely related to exponential family PCA [Collins *et al.*, 2001], which corresponds to setting the sparsity penalty β to zero, and additionally constraining the basis matrix B to have orthogonal columns. We now show that the exponential family sparse coding model can produce better self-taught learning performance than either of these previous methods.

5.1 Text classification

We first apply the exponential family sparse coding algorithms to two self-taught learning problems in text classification: one using binary-valued input vectors, and another using integer-valued (word count) vectors.

We test on five standard webpage classification problems [Do and Ng, 2006], and a newsgroup classification problem (20 newsgroups dataset). We used 470,000 unlabeled news articles (from the Reuters corpus) to learn basis vectors according to the binary and Poisson sparse coding models.[8] Table 3 gives examples of basis vectors obtained from Poisson sparse coding. Many basis vectors appear to encode related words and capture various "topics."

using the KKT conditions to convert between the constraint value C and the equivalent penalty β for that problem. We ran all algorithms until they reached an objective value of $(1+\epsilon)f^{opt}$ where f^{opt} is the optimal objective value (we used $\epsilon = 10^{-6}$).

[8]Details: The webpage classification problems were created using subcategories of the Arts, Business, Health, Recreation and Sports categories of the DMOZ hierarchy. Each of them consisted of 10 separate binary classification problems over a 500 word vocabulary, with stopword removal and stemming. The newsgroup classification problem consisted of 10 binary classification problems constructed using the 20 newsgroups dataset. We used 600 basis vectors, and picked β to achieve roughly 10% nonzero activations. We did not tune these numbers. For learning, we used stochastic updates with mini-batches of 2000 randomly sampled examples, and stopped learning when the objective value did not decrease for 10 consecutive mini-batch iterations.

free	share	subscrib	paint	actor	novel
market	exchange	online	pictur	actress	literari
polici	stock	servic	portrait	film	poet
power	secur	server	museum	comedi	fiction
peopl	commiss	databas	rule	star	univers

Table 3: Examples of basis vectors trained for the Business (left 3 columns) and Arts (right 3 columns) problems, using unlabeled Reuters news data with Poisson sparse coding. Each column shows the five word stems that were most highly active (i.e., had the highest weight) for some basis vector.

Using the learnt basis vectors, we computed features for each classification task using the binary and Poisson sparse coding model. We call our model "ExpSC" and compare against several baselines: the raw words themselves ("Raw"), Gaussian sparse coding ("GSC"), exponential family PCA with the same binary or Poisson exponential family assumption ("ExpPCA"), and also Latent Dirichlet Allocation (LDA), a widely-known topic model for text documents [Blei *et al.*, 2003]. All baselines (except the raw features) were trained using the same unlabeled data as our model. For LDA, we tried 20, 50 and 100 topics, and picked the best. We also consider combinations of the raw word features with the other types of features (e.g., "ExpSC+Raw" indicates a combination of the ExpSC features and the raw features). All features were evaluated using standard supervised-learning classifiers over 100 trials each for 4, 10 and 20 training documents.[9]

Table 4 shows the average test error over all problems for the binary and Poisson case. The exponential family sparse coding features alone frequently outperform the other features, and produce slightly better results when used in combination with the raw features (ExpSC+Raw). The results for Poisson sparse coding are particularly striking, showing 20-30% error reduction in some cases. The other three methods for using unlabeled data (GSC, ExpPCA, LDA) perform poorly in many cases. Figure 1 shows the test errors (with standard error bars) plotted against the training set size for word count (Poisson) data.

5.2 Robotic perception

We also applied the exponential family sparse coding algorithm to a very different self-taught learning problem: object recognition in 3D range data. The data was collected with a laser range finder (Velodyne lidar) in a parking lot environment. Given a 3D box in this space, the task is to predict whether the box contains a car or not.

A standard, robust represenation for such 3D point cloud data is the "spin-image" representation [Johnson and Hebert, 1999]. A detailed description of spin-images is beyond the scope of this paper; but informally, for our application, a spin-image can be thought of as a sheet spinning about the vertical z vector at a point, accumulating counts of other points in

[9]We focused on three standard classifiers—SVM, GDA and kernel dependency estimation (KDE)—that performed best for the raw bag-of-words features out of several generic classifiers, including k-NN and decision trees. We report average results of the best performing classifier for each feature. We picked the β value used for computing the features by cross-validation.

Training set size		Raw	ExpSC	ExpSC+Raw	GSC	GSC+Raw	ExpPCA	ExpPCA+Raw	LDA+Raw
Binary	4	34.3%	**27.9%**	30.0%	-	-	29.1%	31.3%	-
	10	23.0%	20.5%	**20.4%**	-	-	22.3%	22.66%	-
	20	17.7%	17.4%	**16.1%**	-	-	19.3%	17.7%	-
Poisson	4	29.6%	25.4%	**25.0%**	31.2%	26.4%	32.9%	30.4%	33.2%
	10	24.3%	18.9%	**18.6%**	26.1%	22.7%	26.3%	23.8%	24.7%
	20	18.4%	15.2%	**14.9%**	21.4%	17.6%	24.2%	21.2%	17.6%

Table 4: Aggregate test error across all text classification problems (webpages/newsgroups), represented either using binary vectors (Binary) or word count vectors (Poisson). Gaussian sparse coding (GSC) and LDA work better with word counts, so we show results only for this case. LDA+Raw performed better than LDA alone.

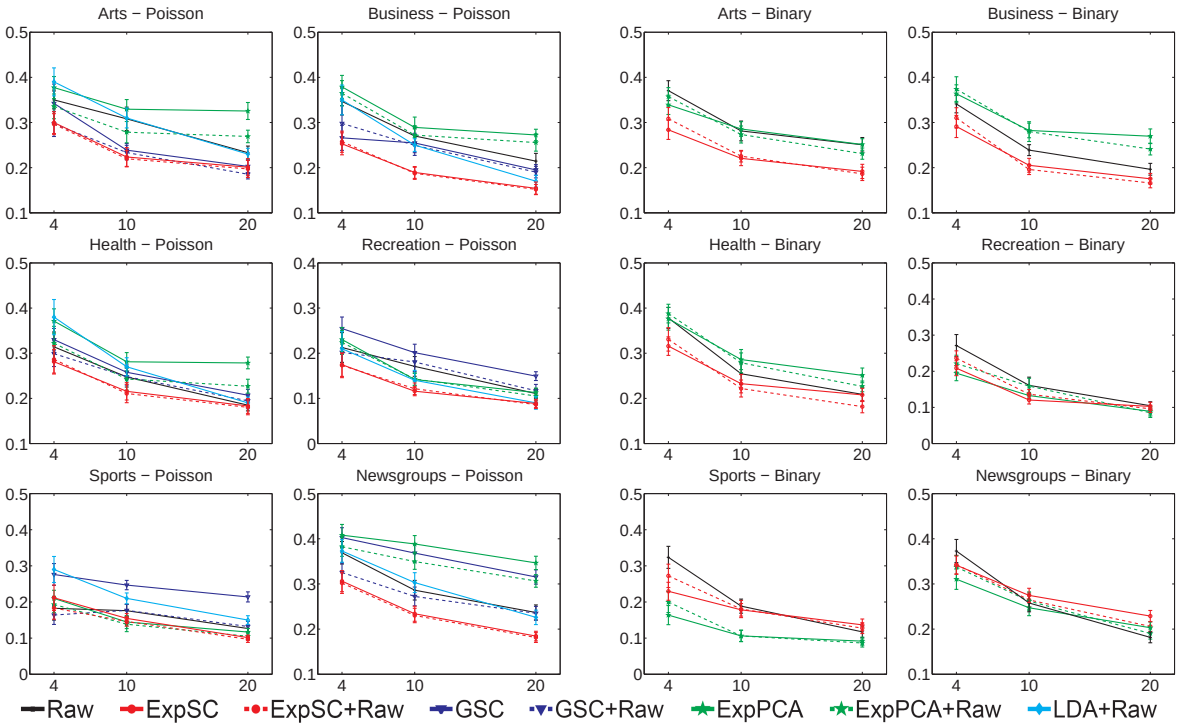

Figure 1: Test error vs. training set size for the webpage and newsgroup classification tasks. See text for explanation of algorithms, and Table 4 for full test errors. Our algorithm is plotted in red color. (Best viewed in color.)

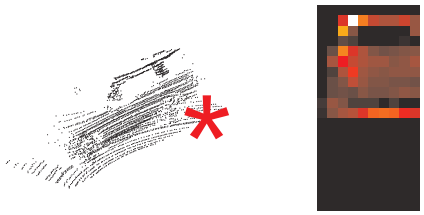

Figure 2: Left: A view of a point cloud for a car produced by the laser range finder. Right: A spin-image generated around the point marked with a red star on the left.

each pixel as it rotates (see Figure 2). Spin-images robustly describe the 3D surfaces around a reference point using a 2D array of counts (20x10 array in our case).

Given a large number of unlabeled spin-images (which can be extracted very easily) and a small number of spin-images from regions manually labeled as car or non-car, our task is to predict whether a new spin image represents a car or non-car region. Since the spin-image is a count-based representation, we apply Poisson sparse coding to this problem. Basis vectors were learnt using 31,000 unlabeled spin-images, and manual examination of the result reveals that certain basis vectors capture various 3D features of cars, roads, trees, and other objects.

Table 5 shows the test error for varying training set size. When compared with using the raw spin-image alone, the Poisson sparse coding features reduce test error by 30% with 4 training examples, and by 8-10% for the other cases.

6 Discussion

Our extension to Gaussian sparse coding is conceptually similar to the extension proposed for PCA by Collins *et al.*, 2001. However, PCA itself defines a linear transform of the features, unlike Gaussian sparse coding, and does not work as well on

Train set size	Raw	ExpSC	ExpSC+Raw	GSC	GSC+Raw	ExpPCA	ExpPCA+Raw
4	34.2%	**23.7%**	25.2%	36.8%	36.0%	38.5%	34.6%
10	22.4%	**20.2%**	20.7%	31.3%	27.5%	31.4%	22.2%
20	18.4%	19.2%	**16.9%**	28.7%	19.5%	23.7%	18.0%

Table 5: Test error for various algorithms and training set sizes for classifying spin-images from the laser range finder.

self-taught learning tasks. It is thus not surprising that the exponential family extension of sparse coding also outperforms exponential family PCA.

Compared to LDA, our model is more general because it can be applied to non-text and non-exchangeable domains. Further, it has been pointed out that when LDA topics are viewed as defining a simplex over word distributions, the exact topics learnt by LDA can vary a lot—the only constraint is that the simplex defined by the topics still spans all the training documents [Blei *et al.*, 2003]. In our view, the additional sparsity constraints in the sparse coding model help reduce this ambiguity, and allow large numbers of "topics" to be learnt robustly.

7 Conclusion

In this paper, we presented a general method for self-taught learning, that extends previous models in a natural way. The extensions can still be solved efficiently using the IRLS-FS algorithm, which we showed to be an efficient algorithm for medium-sized L_1-regularized learning problems with sparse optimal solutions. We showed that our model achieves better self-taught learning performance than Gaussian sparse coding or exponential family PCA on two very different tasks. We believe our model will extend the applicability of self-taught learning to new, previously difficult problems.

Acknowledgments

This work was supported by the DARPA transfer learning program under contract number FA8750-05-2-0249, and by the Office of Naval Research under MURI N000140710747.

References

[Andrew and Gao, 2007] G. Andrew and J. Gao. Scalable training of L_1-regularized log-linear models. In *ICML*, 2007.

[Blei *et al.*, 2003] D. Blei, A. Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[Boyd and Vandenberghe, 2004] S. Boyd and L. Vandenberghe. *Convex Optimization*. Camb. Univ. Press, 2004.

[Collins *et al.*, 2001] Michael Collins, Sanjoy Dasgupta, and Robert E. Schapire. A generalization of principal component analysis to the exponential family. In *NIPS*, 2001.

[Do and Ng, 2006] C. Do and A. Y. Ng. Transfer learning for text classification. In *NIPS*, 2006.

[Efron *et al.*, 2004] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Stat.*, 32, 2004.

[Friedman *et al.*, 2007] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 2, 2007.

[Goodman, 2004] J. Goodman. Exponential priors for maximum entropy models. In *ACL*, 2004.

[Green, 1984] P. J. Green. Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives. *JRSS, Ser. B, Meth.*, 46:149–192, 1984.

[Johnson and Hebert, 1999] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE PAMI*, 21(5):433–449, 1999.

[Koh *et al.*, 2007] K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale L_1-regularized logistic regression. *JMLR*, 8:1519–1555, 2007.

[Lee *et al.*, 2006] S.-I. Lee, H. Lee, P. Abbeel, and A. Y. Ng. Efficient L_1 regularized logistic regression. In *AAAI*, 2006.

[Lee *et al.*, 2007] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2007.

[Lokhorst, 1999] J. Lokhorst. *The Lasso and Generalised Linear Models*. Hons. Project, Department of Statistics, The University of Adelaide, Australia, 1999.

[McCullagh and Nelder, 1989] P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapman & Hall, 1989.

[Ng, 2004] A. Y. Ng. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *ICML*, 2004.

[Olshausen and Field, 1996] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583), 1996.

[Perkins and Theiler, 2003] S. Perkins and J. Theiler. Online feature selection using grafting. In *ICML*, 2003.

[Raina *et al.*, 2007] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML*, 2007.

[Roth, 2004] V. Roth. The generalized lasso. *IEEE Trans. Neural Networks*, 15(1):16–28, 2004.

[Weston *et al.*, 2006] J. Weston, R. Collobert, F. Sinz, L. Bottou, and V. Vapnik. Inference with the universum. In *ICML*, 2006.

[Yu *et al.*, 2008] J. Yu, S. V. N. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-Newton approach to nonsmooth convex optimization. In *ICML*, 2008.