Artificial Intelligence 238 (2016) 83–95

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

# Smooth sparse coding via marginal regression for learning sparse representations ☆

Krishnakumar Balasubramanian [a,*], Kai Yu [b], Guy Lebanon [c]

[a] Department of Statistics, University of Wisconsin-Madison, USA
[b] Horizong Robotics, Beijing, China
[c] LinkedIn, USA

ARTICLE INFO

*Article history:*
Received 16 December 2013
Received in revised form 9 April 2016
Accepted 14 April 2016
Available online 16 May 2016

*Keywords:*
Sparse coding
Dictionary learning
Vision

ABSTRACT

We propose and analyze a novel framework for learning sparse representations based on two statistical techniques: kernel smoothing and marginal regression. The proposed approach provides a flexible framework for incorporating feature similarity or temporal information present in data sets via non-parametric kernel smoothing. We provide generalization bounds for dictionary learning using smooth sparse coding and show how the sample complexity depends on the $L_1$ norm of kernel function used. Furthermore, we propose using marginal regression for obtaining sparse codes which significantly improves the speed and allows one to scale to large dictionary sizes easily. We demonstrate the advantages of the proposed approach, both in terms of accuracy and speed by extensive experimentation on several real data sets. In addition, we demonstrate how the proposed approach can be used for improving semi-supervised sparse coding.



## 1. Introduction

Sparse coding is a popular unsupervised paradigm for learning sparse representations of data samples that are subsequently used in classification tasks. In standard sparse coding, each data sample is coded independently with respect to the dictionary. We propose a smooth alternative to traditional sparse coding that incorporates feature similarity, temporal similarity or similar user-specified similarity measures between the samples into the coding process.

The idea of smooth sparse coding is motivated by the relevance weighted likelihood principle. Our approach constructs a code that is efficient in a smooth sense and as a result leads to improved statistical accuracy over traditional sparse coding. The smoothing operation, which can be expressed as non-parametric kernel smoothing, provides a flexible framework for incorporating several types of domain information that might be available for the user. For example, in image classification, one could use: (1) kernels in feature space for encoding similarity information for images and videos and (2) kernels in time space in case of videos for incorporating temporal relationship. Apart from this, the kernel could also be used to encode similarity information in semi-supervised learning setting.

Most sparse coding training algorithms fall under the general category of alternating procedures with a convex lasso regression sub-problem. While efficient algorithms for such cases exist [17], their scalability for large dictionaries remains a challenge. We propose a novel training method for sparse coding based on marginal regression, rather than solving the

☆ This paper is an invited revision of a paper which first appeared at the International Conference on Machine Learning (ICML) 2013.

* Corresponding author.
*E-mail addresses:* krizna@stat.wisc.edu (K. Balasubramanian), yukai@baidu.com (K. Yu), glebanon@gmail.com (G. Lebanon).

http://dx.doi.org/10.1016/j.artint.2016.04.009
0004-3702/

traditional alternating method with lasso sub-problem. Marginal regression corresponds to performing univariate linear regression corresponding to each dimension, followed by a thresholding step to promote sparsity. For large dictionary sizes, this leads to a significant speedup compared to traditional sparse coding methods without sacrificing statistical accuracy.

Note that the notion of speedup we mention should be interpreted appropriately. There are two contributions we make: (i) the smoothing operation and (ii) the use of marginal regression updates in places of lasso updates. It should be noted that the smoothing operation requires additional computation. The source of speedup is replacing the lasso step with marginal regression step in the alternating minimization procedure. For a fair comparison, one needs to look at the performance of smooth sparse coding (or standard sparse coding) when it uses lasso updates and marginal regression updates. We report the overall timing comparison between the different methods in the experimental section for clarification.

We also develop theoretical analysis that extends the sample complexity result of [29] for dictionary learning using standard sparse coding to the smooth sparse coding case. This result specifically shows how the sample complexity depends on the $L_1$ norm of the kernel function used. Our contributions lead to improved classification accuracy in conjunction with significant computational speedup. Below we summarize our main contributions:

1. we propose a framework based on kernel-smoothing for incorporating feature, time or other similarity information between the samples into sparse coding.
2. we provide sample complexity results for dictionary learning using smooth sparse coding.
3. we propose an efficient marginal regression training procedure for sparse coding.
4. We successfully apply the proposed method in various classification tasks and report improved performance in several situations.

## 2. Related work

Our approach is related to the local regression method [19,11]. More recent related work is [21] that uses smoothing techniques in high-dimensional lasso regression in the context of temporal data. Another recent approach [34], achieves code locality by approximating data points using a linear combination of nearby basis points. The main difference is that traditional local regression techniques [19,11,21] do not involve basis learning. In this work, we propose to learn the basis or dictionary along with the regression coefficients locally. This could be viewed as a high dimensional generalization of low dimensional local smoothing problems with no basis learning. Here we argue that one could directly learn the basis simultaneously while using traditional local-smoothing techniques for improved performance. Furthermore, it provides a natural way to incorporate various similarity information constructed from the data samples themselves in to the sparse coding process.

In contrast to previous sparse coding papers we propose to use marginal regression for learning the regression coefficients, which results in a significant computational speedup with no loss of accuracy. Marginal regression is a relatively old technique that has recently reemerged as a computationally faster alternative to lasso regression [8]. See also [10] for a statistical comparison of lasso regression and marginal regression.

## 3. Smooth sparse coding

**Notation:** We use lower case letters, for example $x$, to represent vectors and upper case letters, for example $X$, to represent matrices, in appropriately defined dimensions. We use $\|\cdot\|_p$ to represent the $L_p$ norm of a vector (we use mostly use $p = 1, 2$ in this paper), $\|\cdot\|_F$ to represent the Frobenius norm of a matrix and $|f|_p$ to represent $L_p$ norm of the function $f$ defined as $(\int |f|^p d\mu)^{1/p}$. Data samples are denoted by subscripts, for example $\{x_i\}_{i=1}^n$ corresponds to $n$ data samples, where each sample $x_i$ is a $d$-dimensional vector.

The standard sparse coding problem consists of solving the following optimization problem,

$$\begin{aligned} \min_{\substack{D\in\mathbb{R}^{d\times K}\\ \beta_i\in\mathbb{R}^K, i=1,\ldots,n}} \quad & \sum_{i=1}^{n} \|x_i - D\beta_i\|_2^2 \\ \text{subject to} \quad & \|d_j\|_2 \le 1 \quad j = 1,\ldots K \\ & \|\beta_i\|_1 \le \lambda \quad i = 1,\ldots n, \end{aligned}$$

where $\beta_i \in \mathbb{R}^K$ corresponds to the encoding of sample $x_i$ with respected to the dictionary $D \in \mathbb{R}^{d\times K}$ and $d_j \in \mathbb{R}^d$ denotes the $j$-column of the dictionary matrix $D$. The dictionary is typically over-complete, implying that $K > d$.

Object recognition is a common sparse coding application where $x_i$ corresponds to a set of features obtained from a collection of image patches, for example Scale-Invariant Feature Transform (SIFT) features [20]. The dictionary $D$ corresponds to an alternative coding scheme that is higher dimensional than the original feature representation. The $L_1$ constraint promotes sparsity of the new encoding with respect to $D$. Thus, every sample is now encoded as a sparse vector that is of higher dimensionality than the original representation.

In some cases the data exhibits a structure that is not captured by the sparse coding setting. For example, SIFT features corresponding to samples from the same class are presumably closer to each other compared to SIFT features from other

classes. Similarly in video, neighboring frames are presumably more related to each other than frames that are farther apart. In this paper we propose a mechanism to incorporate such feature similarity and temporal information into sparse coding, leading to a sparse representation with an improved statistical accuracy, for example as measured by classification accuracy.

We consider the following smooth version of the sparse coding problem:

$$\min_{\substack{D\in\mathbb{R}^{d\times K}\\ \beta_i\in\mathbb{R}^K, i=1,\ldots,n}} \sum_{i=1}^{n}\sum_{j=1}^{n} w(x_j,x_i)\|x_j - D\beta_i\|_2^2 \tag{1}$$

$$\text{subject to} \quad \|d_j\|_2 \le 1 \quad j=1,\ldots K \tag{2}$$

$$\|\beta_i\|_1 \le \lambda \quad i=1,\ldots n, \tag{3}$$

where $\sum_{j=1}^{n} w(x_j,x_i)=1$ for all $i$. It is convenient to define the weight function through a smoothing kernel

$$w(x_j,x_i) = \frac{1}{h_1}\mathcal{K}_1\left(\frac{\rho(x_j,x_i)}{h_1}\right)$$

where $\rho(\cdot,\cdot)$ is a distance function that captures the feature similarity, $h_1$ is the bandwidth, and $\mathcal{K}_1$ is a smoothing kernel. Traditional sparse coding minimizes the reconstruction error of the encoded samples. Smooth sparse coding, on the other hand, minimizes the reconstruction of encoded samples with respect to their neighbors (weighted by the amount of similarity).

The smooth sparse coding setting leads to codes that represent a neighborhood rather than an individual sample and that have lower mean square reconstruction error (with respect to a given dictionary), due to lower estimation variance (see for example the standard theory of smoothed empirical process [7]). There are several possible ways to determine the weight function $w$. One common choice for the kernel function is the Gaussian kernel,

$$w(x_j,x_i) = \frac{1}{h_1\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{\|x_j-x_i\|_2}{h_1}\right)^2\right)$$

where bandwidth parameter $h_1$ is selected using cross-validation. The bandwidth may be fixed throughout the input space, or may vary in order to take advantage of non-uniform samples. Other common choices for the kernel are:

$$\begin{aligned}
\text{Tricube} \quad & (1-\|x_j-x_i\|_2^3)^3 1_{\{\|x_j-x_i\|_2<1\}}\\
\text{Triangular} \quad & (1-\|x_j-x_i\|_2) 1_{\{\|x_j-x_i\|_2<1\}}\\
\text{Uniform} \quad & 2^{-1} 1_{\{\|x_j-x_i\|_2<1\}}
\end{aligned}$$

where $1_{\{\|x_j-x_i\|_2<1\}}$ is the indicator function which evaluates to 1, when the condition inside the bracket is true. For more information about smoothing kernels, we refer the reader to [30].

The distance function $\rho(\cdot,\cdot)$ may be one of the standard distance functions, for example, based on the $L_p$ norm. Alternatively, $\rho(\cdot,\cdot)$ may be expressed by domain experts, learned from data before the sparse coding training, or learned jointly with the dictionary and codes during the sparse coding training.

### 3.1. Spatio-temporal smoothing

In spatio-temporal applications, where the subscript $i$ in the data sample $x_i$ denotes time-stamps, we can extend the kernel to include also a term reflecting the distance between the corresponding time or space

$$w(x_j,x_i) = \frac{1}{h_1}\mathcal{K}_1\left(\frac{\rho(x_j,x_i)}{h_1}\right)\frac{1}{h_2}\mathcal{K}_2\left(\frac{j-i}{h_2}\right).$$

Above, $\mathcal{K}_2$ is a univariate symmetric kernel with bandwidth parameter $h_2$. One example is video sequences, where the kernel combines similarity of the frame features and the time-stamp.

Alternatively, the weight function can feature only the temporal component and omit the first term containing the distance function between the feature representation. A related approach for that situation, is based on the Fused lasso which penalizes the absolute difference between codes for neighboring points. The main drawback of that approach is that one needs to fit all the data points simultaneously whereas in smooth sparse coding, the coefficient learning step decomposes as $n$ separate problems and this provides a computational advantage. Also, while fused Lasso penalty is suitable for time-series data to capture relatedness between neighboring frames, it may not be immediately suitable for other situations that the proposed smooth sparse coding method could handle.

**Structured smooth sparse coding:** Another straightforward extension of the proposed smooth sparse coding approach is by using structured regularizers [4,13,12]. For many applications, there naturally exists some known structure on the codes that could be exploited for better sparse modeling. For example, Wavelet-based decompositions lend themselves well to a

tree organization because of their multi scale nature. The proposed smoothing approach to sparse coding could be extended to such scenarios as well.

Specifically, the $L_1$ regularizer in Eq. (3) could be replaced by any norm $R(\beta_i)$ for $i = 1, \ldots, n$. Some choices for the regularizer are the $(p, 1)$-mixed norms when the dictionary exhibits a natural grouping; tree-structured norms when the dictionary is assumed to be embedded in a tree-structure; hierarchical norms which are typically composition of several norms and graph based norms and random field norms [4,13].

## 4. Marginal regression for smooth sparse coding

A standard algorithm for sparse coding is the alternating bi-convex minimization procedure, where one alternates between (i) optimizing for codes (with a fixed dictionary) and (ii) optimizing for dictionary (with fixed codes). Note that step (i) corresponds to regression with $L_1$ constraints and step (ii) corresponds to least squares with $L_2$ constraints. In this section we show how marginal regression could be used to obtain better codes faster (step (i)). In order to do so, we first give a brief description of the marginal regression procedure.

**Marginal regression:** Consider a regression model $y = X\beta + z$ where $y \in \mathbb{R}^n$, $\beta \in \mathbb{R}^p$, $X \in \mathbb{R}^{n \times p}$ with orthonormal columns (denoted by $x_j$), and $z$ is the noise vector. Marginal regression proceeds as follows:

- Calculate the least squares solution

$$\hat{\alpha}^{(j)} = x_j^T y.$$

- Threshold the least-square coefficients

$$\hat{\beta}^{(j)} = \hat{\alpha}^{(j)} 1_{\{|\hat{\alpha}^{(j)}| > t\}}, \quad j = 1, \ldots, p.$$

Marginal regression requires just $O(np)$ operations compared to $O(p^3 + np^2)$, the typical complexity of lasso algorithms. When $p$ is much larger than $n$, marginal regression provides two orders of speedup over Lasso based formulations. Note that in sparse coding, the speedup occurs for each iteration of the outer loop, thus enabling sparse coding for significantly larger dictionary sizes. Recent studies have suggested that marginal regression is a viable alternative for Lasso given its computational advantage over lasso. A comparison of the statistical properties of marginal regression and lasso is available in [8,10].

**Code update (step (i)):** Applying marginal regression to smooth sparse coding, we obtain the following scheme. The marginal least squares coefficients are

$$\hat{\alpha}_i^{(k)} = \sum_{j=1}^{n} \frac{w(x_j, x_i)}{\|d_k\|_2} d_k^T x_j.$$

We sort these coefficient in terms of their absolute values, and select the top $s$ coefficients whose $L_1$ norm is bounded by $\lambda$:

$$\hat{\beta}_i^{(k)} = \begin{cases} \hat{\alpha}_i^{(k)} & k \in S \\ 0 & k \notin S \end{cases}, \quad \text{where}$$

$$S = \left\{ 1, \ldots, s : s \leq d : \sum_{k=1}^{s} |\hat{\alpha}_i^{(k)}| \leq \lambda \right\}$$

We select the thresholding parameter using cross validation in each of the sparse coding iterations. Note that the same approach could be used with structured regularizers too, for example [4,13].

**Dictionary update (step (ii)):** Marginal regression works well when there is minimal correlation between the different dictionary atoms. In the linear regression setting, marginal regression provides an exact solution with orthogonal data [10]. In the context of sparse coding, this corresponds to having uncorrelated or incoherent dictionaries [28]. One way to measure such incoherence is using the babel function, which bounds the maximum inner product between two different columns $d_i, d_j$:

$$\mu_s(D) = \max_{i \in \{1,\ldots,d\}} \max_{\Lambda \subset \{1,\ldots,d\} \setminus \{i\}; |\Lambda| = s} \sum_{j \in \Lambda} |d_j^\top d_i|.$$

Therefore, one can proceed by imposing conditions on the babel function, which is computationally challenging. An alternative, which leads to easier computation is by adding the term $\|D^T D - I_{K \times K}\|_F^2$ to the reconstruction objective, when optimizing over the dictionary matrix $D$. This leads to the following optimization problem for dictionary update step:

$$\hat{D} = \arg\min_{D \in \mathcal{D}} F(D) \quad \text{where}$$

$$F(D) = \sum_{i=1}^{n} \|x_i - D\hat{\beta}_i\|_2^2 + \gamma \|D^\top D - I\|_F^2$$

and $\mathcal{D} = \{D \in \mathbb{R}^{d \times K} : \|d_j\|_2 \leq 1\}$. The regularization term $\gamma$ controls the level of incoherence enforced.

This optimization problem is of the form of minimizing a differentiable function over a closed convex set. We use the gradient projection method [3,26] for solving the optimization problem. The gradient of the objective with respect to $D$ at each iteration is given by

$$\nabla F(D) = 2\left(D\hat{B}\hat{B}^\top - X\hat{B}^\top\right) + 4\gamma\left(DD^\top D - D\right), \tag{4}$$

where $\hat{B} = [\hat{\beta}_1, \ldots, \hat{\beta}_n]$ is the matrix of codes from the previous code update step, $X \in \mathbb{R}^{d \times n}$ is the data in matrix format. The gradient projection descent iterations are given by

$$D(t+1) = \Pi_{\mathcal{D}}\left(D(t) - \eta_t \nabla F(D(t))\right), \tag{5}$$

where by $\Pi_{\mathcal{D}}$, we denote column-wise projection of the dictionary matrix on to the unit ball and $t$ is the index for sub-iteration count for each dictionary update step. Specifically, for each dictionary update step, we run the gradient projected descent algorithm until convergence. Note that projection of a vector onto the $l_2$ ball is straightforward since we only need to rescale the vector towards the origin, i.e., normalize the vectors with length greater than 1.

Convergence to local point of gradient projection methods for minimizing a differentiable function over a convex set was analyzed in [26]. Similar guarantees could be provided for each of the dictionary update steps. A heuristic approach for dictionary update with incoherence constraint was proposed in [23] and more recently in [25], where the L-BFGS method was used for the unconstrained problem and the norm constraint was enforced at the final step. We found that the proposed gradient projected descent method performed empirically better than both these approaches. Furthermore both approaches are heuristic and do not guarantee local convergence for the dictionary update step.

Finally, a sequence of such updates corresponding to step (i) and step (ii) converges to a stationary point of the optimization problem (this can be shown using Zangwill's theorem [35]). But no provable algorithm that converges to the global minimum of the smooth sparse coding exists yet. Nevertheless, the main idea of this section is to speedup the existing alternating minimization procedure for obtaining sparse representations, by using marginal regression. We leave a detailed theoretical analysis of the individual dictionary update steps and the overall alternating procedure (for codes and dictionary) as future work.

## 5. Sample complexity of smooth sparse coding

In this section, we analyze the sample complexity of the proposed smooth sparse coding framework. We provide uniform convergence bounds over the dictionary space and hence prove a sample complexity result for dictionary learning under smooth spare coding setting. We are mainly interested in studying how the sample complexity scales with respect to the problem parameters. We leverage the analysis for dictionary learning in the standard sparse coding setting [29] and extend it to the smooth sparse coding setting. The main difficulty for the smooth sparse coding setting is obtaining a covering number bound for an appropriately defined class of functions (see Theorem 5.1 for more details).

We begin by re-representing the smooth sparse coding problem in a convenient form for analysis. Let $x_1, \ldots, x_n$ be independent random variables with a common probability measure $\mathbb{P}$ with a density $p$. We denote by $\mathbb{P}_n = n^{-1}\sum_{i=1}^{n} \delta_{x_i}$ the empirical measure over the $n$ samples. Let $\mathcal{K}_{h_1}(\cdot) = \frac{1}{h_1}\mathcal{K}_1(\frac{\cdot}{h})$. With the above notations, the reconstruction error at the point $x$ is given by

$$r_\lambda(x) = \int \min_{\beta \in \mathcal{S}_\lambda} \|x' - D\beta\|_2 \mathcal{K}_{h_1}(\rho(x, x'))\, d\mathbb{P}_n(x')$$

where

$$\mathcal{S}_\lambda = \{\beta : \|\beta\|_1 \leq \lambda\}.$$

The empirical reconstruction error is

$$\mathsf{E}_{\mathbb{P}_n}(r) = \iint \min_{\beta \in \mathcal{S}_\lambda} \|x' - D\beta\|_2 \mathcal{K}_{h_1}(\rho(x, x'))\, d\mathbb{P}_n(x')\, d\mathbb{P}_n(x)$$

and its population version is

$$\mathsf{E}_{\mathbb{P}}(r) = \iint \min_{\beta \in \mathcal{S}_\lambda} \|x' - D\beta\|_2 \mathcal{K}_{h_1}(\rho(x, x'))\, d\mathbb{P}_n(x')\, d\mathbb{P}(x).$$

**Algorithm 1** Smooth Sparse Coding via Marginal Regression.

**Input:** Data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ and kernel/similarity measure $\mathcal{K}_1$ and $d_1$.
**Precompute:** Compute the weight matrix $w(i, j)$ using the kernel/similarity measure and
**Initialize:** Set the dictionary at time zero to be $D_0$.
**Algorithm:**
**repeat**
**Step (i):** For all $i = 1, \ldots, n$, solve marginal regression:

$$\hat{\alpha}_i^{(k)} = \sum_{j=1}^{n} \frac{w(x_j, x_i)}{\|d_k\|_2} d_k^T x_j$$

$$\hat{\beta}_j^{(k)} = \begin{cases} \hat{\alpha}_j^{(k)} & j \in S \\ 0 & j \notin S \end{cases},$$

$$S = \{1, \ldots, s; s \leq d : \sum_{k=1}^{s} |\hat{\alpha}_i^{(k)}| \leq \lambda\}.$$

**Step (ii):** Update the dictionary based on codes from previous step by solving the following optimization problem

$$\hat{D} = \arg\min_{D \in \mathcal{D}} \sum_{i=1}^{n} \|x_i - D\hat{\beta}_i\|_2^2 + \gamma \|D^\top D - I\|_F^2$$

where $\mathcal{D} = \{D \in \mathbb{R}^{d \times K} : \|d_j\|_2 \leq 1\}$,

using the gradient descent step from Equation (5).
**until** convergence
**Output:** Return the learned codes and dictionary.

Our goal is to show that the sample reconstruction error is close to the true reconstruction error. Specifically, to show $\mathsf{E}_{\mathbb{P}}(r_\lambda) \leq (1 + \kappa)\mathsf{E}_{\mathbb{P}_n}(r_\lambda) + \epsilon$ where $\epsilon, \kappa \geq 0$, we bound the covering number of the class of functions corresponding to the reconstruction error. We assume a dictionary of bounded babel function, which holds as a result of the relaxed orthogonality constraint used in the Algorithm 1 (see also [23]). We define the set of $r$ functions with respect to the dictionary $D$ (assuming data lies in the unit $d$-dimensional ball $\mathbb{S}^{d-1}$) by

$$\mathcal{F}_\lambda = \{r_\lambda : \mathbb{S}^{d-1} \to \mathbb{R} : D \in \mathbb{R}^{d \times K}, \|d_i\|_2 \leq 1, \mu_s(D) \leq \gamma\}.$$

The following theorem bounds the covering number of the above function class.

**Theorem 5.1.** *For every $\epsilon > 0$, the metric space $(\mathcal{F}_\lambda, |\cdot|_\infty)$ has a subset of cardinality at most $\left(\frac{4\lambda |\mathcal{K}_{h_1}(\cdot)|_1}{\epsilon(1-\gamma)}\right)^{dK}$, such that every element from the class is at a distance of at most $\epsilon$ from the subset, where $|\mathcal{K}_{h_1}(\cdot)|_1 = \int |\mathcal{K}_{h_1}(x)| \, d\mathbb{P}$.*

**Proof.** Let $\mathcal{F}'_\lambda = \{r'_\lambda : \mathbb{S}^{d-1} \to \mathbb{R} : D \in d \times K, \|d_i\|_2 \leq 1\}$, where $r'_\lambda(x) = \min_{\beta \in \mathcal{S}_\lambda} \|D\beta - x\|$. With this definition we note that $\mathcal{F}_\lambda$ is just $\mathcal{F}'_\lambda$ convolved with the kernel $\mathcal{K}_{h_1}(\cdot)$. By Young's inequality [7] we have,

$$|\mathcal{K}_{h_1} * (s_1 - s_2)|_p \leq |\mathcal{K}_{h_1}|_1 |s_1 - s_2|_p, \quad 1 \leq p \leq \infty$$

for any $L_p$ integrable functions $s_1$ and $s_2$. Using this fact, we see that convolution mapping between metric spaces $\mathcal{F}'$ and $\mathcal{F}$ converts $\frac{\epsilon}{|\mathcal{K}_{h_1}(\cdot)|_1}$ covers into $\epsilon$ covers. From [29], we have that the class $\mathcal{F}'_\lambda$ has $\epsilon$ covers of size at most $(\frac{4\lambda}{\epsilon(1-\gamma)})^{dK}$. This proves the statement of the theorem. □

The above theorem can be used in conjunction with standard statements in the literature for bounding the generalization error of empirical risk minimization algorithms based on covering numbers. We have provided the general statements in the appendix for completeness of this paper. Below, we provide generalization bounds for smooth sparse coding problem, corresponding to slow rates and fast rates.

**Slow rates:** When the theorem on covering numbers for the function class $\mathcal{F}_\lambda$ (Theorem 5.1) is used along with Lemma 1 stated in the appendix (corresponding to slow rate generalization bounds) it is straightforward to obtain the following generalization bounds with slow rates for the smooth sparse coding problem.

**Theorem 5.2.** *Let $\gamma < 1$, $\lambda > e/4$ with distribution $\mathbb{P}$ on $\mathbb{S}^{d-1}$. Then with probability at least $1 - e^{-t}$ over the $n$ samples drawn according to $\mathbb{P}$, for all the $D$ with unit length columns and $\mu_s(D) \leq \gamma$, we have:*

$$E_{\mathbb{P}}(r_\lambda) \leq E_{\mathbb{P}_n}(r_\lambda) + \sqrt{\frac{dK \ln\left(\frac{4\sqrt{n}\lambda |\mathcal{K}_{h_1}(\cdot)|_1}{(1-\gamma)}\right)}{2n}} + \sqrt{\frac{t}{2n}} + \sqrt{\frac{4}{n}}$$

**Table 1**
Time comparison of coefficient learning in Sparse Coding (SC) and Smooth Sparse Coding (SSC) with either Lasso or Marginal regression updates. The dictionary update step was same for all methods. The format correspond to average ± standard error.

| Method | Time (sec) |
|---|---|
| SC + LASSO | $524.5 \pm 12$ |
| SC + MR | $242.2 \pm 10$ |
| SSC + LASSO | $560.2 \pm 12$ |
| SSC + MR | $184.4 \pm 19$ |

The above theorem, establishes that the generalization error scales as $O(n^{-1/2})$ (assuming the other problem parameters are fixed). Note that the term $\kappa$ is exactly zero. In the literature, this is called as slow-rate bound as one could get a much improved result, albeit only for the case when $\kappa$ is strictly greater than zero, though it could taken arbitrarily close to zero.

– **Fast rates:** Under further assumptions ($\kappa > 0$), it is possible to obtain faster rates of $O(n^{-1})$ for smooth sparse coding, similar to the ones obtained for general learning problems in [1]. It should be noted that even though the rates obtained are better, the constant in front of the empirical reconstruction error is strictly greater than one and can never be exactly one. The following theorem gives the precise statement.

**Theorem 5.3.** *Let $\gamma < 1$, $\lambda > e/4$, $dK > 20$ and $n \geq 5000$. Then with probability at least $1 - e^{-t}$, we have for all $D$ with unit length and $\mu_s(D) \leq \gamma$,*

$$E_{\mathbb{P}}(r_\lambda) \leq 1.1 E_{\mathbb{P}_n}(r_\lambda) + 9\frac{dK \ln\left(\frac{4n\lambda|\mathcal{K}_{h_1}(\cdot)|_1}{(1-\gamma)}\right) + t}{n}.$$

The above theorem follows from the theorem on covering number bound (Theorem 5.1) above and Lemma 2 from the appendix. In both statements the definition of $r_\lambda(x)$ differs from (1) by a square term. Note that it could be incorporated easily into the above bounds resulting in an additive factor of 2 inside the logarithm term. We are mainly interested in studying the scaling of generalization error with respect to problem parameters and the above step has no effect on that. We refer the interested reader to [29] for more details.

## 6. Experiments

We demonstrate the advantage of the proposed approach both in terms of speedup and accuracy over standard sparse coding. A detailed description of all real-world data sets used in the experiments are given in the appendix. As discussed before, the overall optimization procedure is non-convex. The stopping criterion was chosen as follows: stop iterating when the value of the reconstruction error does not change by more than 0.001%. Though this does not guarantee convergence to a global optimum, according to the experimental results, we see that the points of convergence invariably resulted in a good local optimum as reflected by the good empirical performance. Furthermore, in all the experiments, we ran 10 iterations of the projected gradient descent algorithm for each dictionary update step. We fixed the learning rate for all iterations of gradient projection descent algorithm as $\eta = \eta_t = 0.01$ as it was found to perform well in the experiments. The parameters $\gamma$ and $\lambda$ are set for each experiment based on cross-validation (we first tuned for $\gamma$ and then for $\lambda$) for classification results on training set as is done in the literature [32]. Furthermore, for each experiment, the time taken for training is also reported for comparison. In all our experiments we use the tricube kernel. We use classification accuracy (when linear SVM was used) as a measure of performance in all experiments, except for the following section on speed comparison, where we use reconstruction error. Furthermore, we use the LIBSVM [5] in our experiments and use cross-validation to set regularization parameter of SVM.

### 6.1. Speed comparison

We conducted synthetic experiments to examine the speedup provided by sparse coding with marginal regression. The data was generated from a 100-dimensional mixture of two Gaussian distribution that satisfies $\|\mu_1 - \mu_2\|_2 = 3$ (with identity covariance matrices). The dictionary size was fixed at 1024. The number of data point was 1000.

We compare the proposed smooth sparse coding algorithm, standard sparse coding with lasso [17] and marginal regression updates respectively, with a relative reconstruction error $\|X - \hat{D}\hat{B}\|_F / \|X\|_F$ convergence criterion. We experimented with 200 different values of the relative reconstruction error (all values less than 10%) and report the average time. From Table 1, we see that smooth sparse coding with marginal regression takes significantly less time to achieve a fixed reconstruction error. This is due to the fact that it takes advantage of the spatial structure and use marginal regression updates. It is worth mentioning that standard sparse coding with marginal regression, performs faster than the other two methods which uses Lasso as expected.

**Table 2**
Face recognition: Test set classification accuracy based on coding at Pixel Level using sparse coding and proposed smoothed sparse coding.

| Method | SC | SSC-tricube |
|---|---|---|
| Train error | 8.382 | 9.342 |
| Test error | 6.212 | 6.942 |

**Table 3**
Test set accuracy for face recognition on the CMU-Multi-PIE data set (left) 15 scene (middle) and Caltech-101 (right) respectively. The performance of the smooth sparse coding approach is better than the standard sparse coding and LLC in all cases. The number reported in bracket corresponds to training time in seconds. The format corresponds to average $\pm$ standard error in all cases.

| | CMU-Multi-PIE | 15 scene | Caltech-101 |
|---|---|---|---|
| SC | $92.70 \pm 1.21$ $(489 \pm 36)$ | $80.28 \pm 2.12$ $(501 \pm 39)$ | $73.20 \pm 1.14$ $(591 \pm 32)$ |
| LLC | $93.70 \pm 2.22$ $(532 \pm 39)$ | $82.28 \pm 1.98$ $(585 \pm 32)$ | $74.82 \pm 1.65$ $(611 \pm 32)$ |
| SSC | $95.05 \pm 2.33$ $(321 \pm 32)$ | $84.53 \pm 2.57$ $(287 \pm 21)$ | $77.54 \pm 2.59$ $(410 \pm 22)$ |

**Table 4**
Effect of dictionary size on classification accuracy using smooth sparse coding and marginal regression on 15 scene and Caltech-101 data set. The format corresponds to average $\pm$ standard error.

| Dictionary size | 15 scene | Caltech-101 |
|---|---|---|
| 1024 | $84.42 \pm 2.01$ | $77.14 \pm 2.23$ |
| 2048 | $87.92 \pm 2.35$ | $79.75 \pm 1.44$ |
| 4096 | $90.22 \pm 2.91$ | $81.01 \pm 1.17$ |

### *6.2. Experiments with kernel in feature space*

We conducted several experiments demonstrating the advantage of the proposed coding scheme in different settings. Concentrating on face and object recognition from static images, we evaluated the performance of the proposed approach along with standard sparse coding and LLC [34], another method for obtaining sparse features based on locality. Following [32], we used the following approach for generating sparse image representation: we densely sampled $16 \times 16$ patches from images at the pixel level on a grid with step size 8 pixels, computed SIFT features [20], and then computed the corresponding sparse codes over a 1024-size dictionary. We used max-pooling to get the final representation of the image based on the codes for the patches. Max-pooling involves taking max over each entry of the computed codes. It is motivated by the neural activation structure of the human brain and has been successfully applied in several image classification tasks [32]. The process was repeated with different randomly selected training and testing images and we report the average per-class recognition rates (together with its standard error estimate) based on one-vs-all SVM classification.

#### *6.2.1. Smoothing at pixel level versus feature level*

In this section we investigate the performance of smooth sparse coding with pixels as features on the CMU Multi-PIE data set. We extract the patches directly on the images represented as pixels as opposed to using the SIFT features. From Tables 2 and 3, we note that the performance of smooth sparse coding directly on the pixels is not as good as smooth sparse coding with SIFT features, i.e., sparse coding performs better at pixel level than smooth sparse coding. One possible reason is that the pixels themselves might exhibit significant variations which are not captured by the smoothing operation. This observation is consistent with our intuition that smoothing only helps when there is some structure in points used for coding. Raw images might not have much discriminative information as such, but extracting SIFT features helps to get a set of points with rich structure. The proposed smooth sparse coding tries to exploit that structure by explicitly encoding the similarity.

#### *6.2.2. Image classification*

We conducted image classification experiments on the CMU-Multi-PIE, 15 Scene and Caltech-101 data sets. The dictionary size was 1024. We also report the timing results for the training stage. As Table 3 indicates, our smooth sparse coding algorithm resulted in significantly higher classification accuracy than standard sparse coding and LLC. In fact, the reported performance is better than previous reported results using unsupervised sparse coding techniques [32].

*Dictionary size:* In order to demonstrate the use of scalability of the proposed method with respect to dictionary size, we report classification accuracy with increasing dictionary sizes using smooth sparse coding. The main advantage of the proposed marginal regression training method is that one could easily run experiments with larger dictionary sizes, which typically takes a significantly longer time for other algorithms. For both the Caltech-101 and 15-scene data set, classification accuracy increases significantly with increasing dictionary sizes as seen in Table 4.

**Table 5**
Action recognition (accuracy) for cited method (left), Hog3d+ SC (middle) and Hog3d+ SSC (right): the KTH data set (top) YouTube action dataset (bottom). The numbers in bracket correspond to training time. The format corresponds to average ± standard error.

| Cited method | SC | SSC |
|---|---|---|
| 92.10 [31] | $92.423 \pm 2.12$ $(721 \pm 23)$ | $94.393 \pm 3.21$ $(433 \pm 32)$ |
| 71.2 [18] | $72.640 \pm 2.53$ $(698 \pm 23)$ | $75.022 \pm 2.01$ $(361 \pm 29)$ |

*6.2.3. Action recognition*

We further conducted an experiment on activity recognition from videos with the KTH action and YouTube datasets. Similar to the static image case, we follow the standard approach for generating sparse representations for videos as in [31]. We densely sample $16 \times 16 \times 10$ blocks from the video and extract HoG-3d [15] features from the sampled blocks. We then use smooth sparse coding and max-pooling to generate the video representation (dictionary size was fixed at 1024 and cross-validation was used to select the regularization and bandwidth parameters). Previous approaches include sparse coding, vector quantization, and $k$-means on top of the HoG-3d feature set (see [31] for a comprehensive evaluation). As indicated by Table 5, smooth sparse coding results in higher classification accuracy than previously reported state-of-the-art and standard sparse coding on both datasets (see [31,18] for a description of the alternative techniques).

*6.2.4. Discriminatory power*

In this section, we describe another experiment that contrasts the codes obtained by sparse coding and smooth sparse coding in the context of a subsequent classification task. We set the dictionary size to be 1024. We first compute the codes based on patches and combine them with max-pooling to obtain the image level representation. Note that a smiler study was undertaken in [33]. We then compute the fisher discriminant score (ratio of within-class variance to between-class variance) for each dimension as a measure of discrimination power realized by the representations.

Fig. 1, graphs a histogram of the ratio of smooth sparse coding Fisher score over standard sparse coding Fisher score $R(d) = F_1(d)/F_2(d)$ for the 15-scene dataset (top) and the Youtube dataset (bottom). Note that the patches used are common for both the procedures and the features obtained are more discriminatory for smooth sparse coding. The histograms demonstrate the improved discriminatory power of smooth sparse coding over regular sparse coding.

*6.3. Experiments using temporal smoothing*

In this section we describe an experiment conducted using the temporal smoothing kernel on the Youtube persons dataset. We extracted SIFT descriptors for all $16 \times 16$ patches sampled on a grid of step size 8 and used smooth sparse coding with time kernel to learn the codes and max pooling to get the final video representation. We avoided pre-processing steps such as face extraction or face tracking. The dictionary size was set at 1024. Note that in the previous action recognition video experiment, video blocks were densely sampled and used for extracting HoG-3d features. In this experiment, on the other hand, we extracted SIFT features from individual frames and used the time kernels to incorporate the temporal information into the sparse coding process.

For this case, we also compared to the more standard fused-lasso based approach [27]. Note that in fused Lasso based approach, in addition to the standard $L_1$ penalty, an additional $L_1$ penalty on the difference between the neighboring frames for each dimensions is used. This tries to enforce the assumption that in a video sequence, neighboring frames are more related to one another as compared to frames that are farther apart.

Table 6 shows that smooth sparse coding achieved higher accuracy than fused lasso and standard sparse coding. Smooth sparse coding has comparable accuracy on person recognition tasks to other methods that use face-tracking, for example [14]. Another advantage of smooth sparse coding is that it is significantly faster than sparse coding and the fused lasso.

## 7. Semi-supervised smooth sparse coding

One of the primary difficulties in several image classification tasks is the lack of availability of labeled data. This motivated a semi-supervised learning approach for dictionary learning [22]. The motivation for such an approach is that unlabeled data from a related domain might have useful visual patterns that might be similar to the problem at hand. Hence, learning a high-level dictionary based on data from a different domain aids the classification task at hand.

We propose that the smooth sparse coding approach might be useful in this setting. The motivation is as follows: in semi-supervised learning, typically not all samples from a different data set might be useful for the task at hand. Using smooth sparse coding, one can provide more weight to the useful points more than the other points (the weights being calculated based on feature/time similarity kernel) to obtain better dictionaries and sparse representations. Other approach to handle a lower number of labeled samples include collaborative modeling or multi-task approaches which impose a shared structure on the codes for several tasks and use data from all the tasks simultaneously, for example group sparse coding [2]. The proposed approach provides an alternative when such collaborative modeling assumptions do not hold, by using relevant unlabeled data samples that might help the task at hand via appropriate weighting.

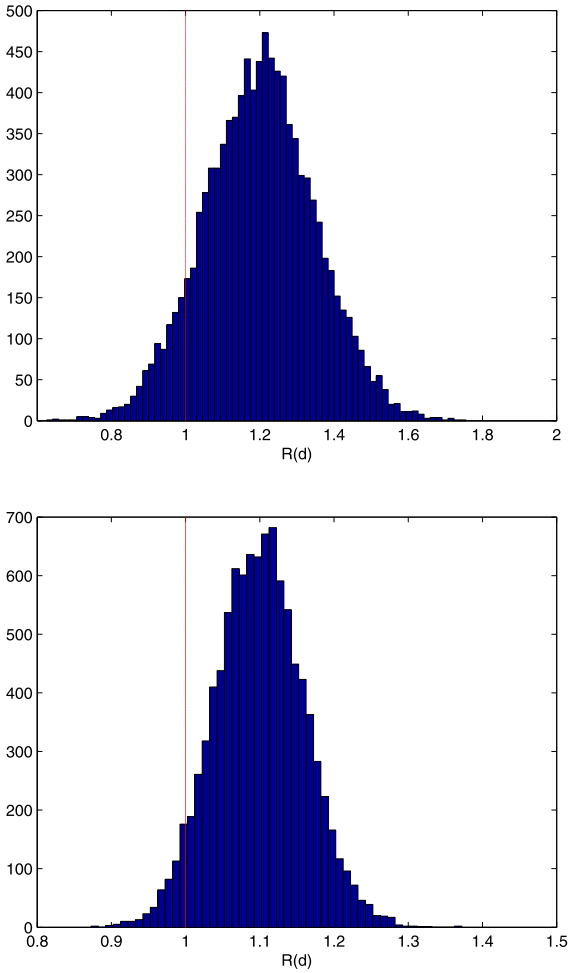


**Fig. 1.** Comparison between the histograms of Fisher discriminant score realized by sparse coding and smooth sparse coding. The images represent the histogram of the ratio of smooth sparse coding Fisher score over standard sparse coding Fisher score (top: image data set; bottom: video). A value greater than 1 implies that smooth sparse coding is more discriminatory.

**Table 6**
Linear SVM accuracy for person recognition task from YouTube face video dataset. The number in bracket corresponds to training time. The format corresponds to average $\pm$ standard error.

| Method | Fused Lasso | SC | SSC-tricube |
|---|---|---|---|
| Accuracy | $68.59 \pm 1.91$ $(703 \pm 36)$ | $65.53 \pm 2.64$ $(691 \pm 36)$ | $71.21 \pm 3.12$ $(382 \pm 26)$ |

We now describe an experiment that examines the proposed smoothed sparse coding approach in the context of semi-supervised dictionary learning. We use data from both the CMU Multi-PIE dataset (session 1) and faces-on-tv dataset (treated as frames) to learn a dictionary using a feature similarity kernel. We follow the same procedure described in the previous experiments to construct the dictionary. The dictionary size was set at 1024. In the test stage we use the obtained dictionary for coding data from sessions 2, 3, 4 of the CMU-Multi-PIE data set, using smooth sparse coding. Note that semi-supervision was used only in the dictionary learning stage and the classification stage used supervised SVM.

Table 7 shows the test set error rate and compares it to standard sparse coding and LLC [34]. Smooth sparse coding achieves significantly lower test error rate than the two alternative techniques. Based on the experiments, we conclude that the smoothing approach described in this paper may be useful in cases where there is a small set of labeled data, such as semi-supervised learning.

## 8. Discussion and future work

We propose a simple framework for incorporating similarity in feature space and space or time into sparse coding. We also propose in this paper modifying sparse coding by replacing the lasso optimization stage by marginal regression and

**Table 7**
Semi-supervised learning test set error: Dictionary learned from both CMU Multi-PIE and faces-on-tv data set using feature similarity kernel, used to construct sparse codes for CMU Multi-PIE data set. The format correspond to average $\pm$ standard error.

| Method | SC | LLC | SSC-tricube |
|---|---|---|---|
| Test error | $6.345 \pm 0.32$ | $6.003 \pm 0.43$ | $4.975 \pm 0.10$ |

adding a constraint to enforce incoherent dictionaries. The resulting algorithm is significantly faster. This facilitates scaling up the sparse coding framework to large dictionaries, an area which is usually restricted due to intractable computation.

This work leads to several interesting follow-up directions. On the theoretical side, local convergence of Lasso-based sparse coding has been analyzed recently – preliminary examination suggests that the proposed marginal-regression based sparse coding algorithm might be more favorable for the local convergence analysis. It is also interesting to explore tighter generalization error bounds by directly analyzing the solutions of the marginal regression based iterative algorithm. Methodologically, it is interesting to explore if using an adaptive or non-constant kernel bandwidth leads to higher accuracy. Furthermore alternative methods for imposing incoherence constraints that may lead to easier optimization is an interesting direction to investigate.

## Appendix A. Data set description

### *A.1. CMU Multi-PIE face recognition*

The CMU Multi-PIE dataset is one of the standard data sets used for face recognition experiments. The data set contains 337 subjects across simultaneous variations in pose, expression, and illumination. We ignore the 88 subjects that were considered as outliers in [32] and used the rest of the images for our face recognition experiments. We follow [32] and use the 7 frontal extreme illuminations from session one as train set and use other 20 illuminations from Sessions 2–4 as test set.

### *A.2. 15 scenes categorization*

The 15-Scenes data set [16] consists of 4485 images from 15 categories, with the number of images each category ranging from 200 to 400. The categories correspond to scenes from various settings like kitchen, living room etc.

### *A.3. Caltech-101 data set*

The Caltech-101 data set [9] consists of images from 101 classes like animals, vehicles, flowers, etc. The number of images per category varies from 30 to 800. Most images are of medium resolution ($300 \times 300$). All images are used gray-scale images. We use 30 images per category and test on the rest.

### *A.4. Activity recognition*

The KTH action dataset [24] consists of 6 human action classes. Each action is performed several times by 25 subjects and is recorded in four different scenarios. In total, the data consists of 2391 video samples. The YouTube actions data set has 11 action categories and is more complex and challenging [18]. It has 1168 video sequences of varied illumination, background, resolution etc.

### *A.5. Youtube person data set*

The YouTube person data set [14] contains 1910 sequences of 47 subjects, mostly actors/actresses and politicians, from YouTube. As most of the videos are low resolution and recorded at high compression rates, they are noisy and contain low-quality image frames.

### *A.6. Faces-on-TV data set*

The data was first used in [6]. Here we replicate the data set description from [6] for the sake of completeness. The dataset contains approximately 3,000 face images extracted from 8 episodes of the TV-show LOST, annotated with ground-truth names, along with approximately registered faces and frame information if one wants to re-extract the faces from the videos. It also contains automatically extracted names using a screenplay aligned with the video closed captions.

## Appendix B. Generalization bounds for learning problems

In this section, we reproduce the specific generalization bounds we used from the literature, for the sake of completeness. We first state the following general lemma regarding generalization error bounds with slow rates for a learning problem with given covering number bounds.

**Lemma 1** *(see [29]). Let $\mathcal{Q}$ be a function class of $[0, B]$ functions with covering number $(\frac{C}{\epsilon})^d > \frac{e}{B^2}$ under $|\cdot|_\infty$ norm. Then for every $t > 0$ with probability at least $1 - e^{-t}$, for all $f \in \mathcal{Q}$, we have:*

$$Ef \leq E_n f + B\left(\sqrt{\frac{d\ln(C\sqrt{n})}{2n}} + \sqrt{\frac{t}{2n}}\right) + \sqrt{\frac{4}{n}}.$$

Next, we state a general lemma regarding generalization error bounds with fast rates:

**Lemma 2** *(see [29]). Let $\mathcal{Q}$ be a function class of $[0, 1]$ functions that can be covered for any $\epsilon > 0$ by at most $(C/\epsilon)^d$ balls of radius $\epsilon$ in the $|\cdot|_\infty$ metric, where $C \geq e$ and $\beta > 0$. Then with probability at least $1 - \exp(-t)$ we have for all functions $f \in \mathcal{Q}$,*

$$Ef \leq (1+\beta)E_n f + K(d, m, \beta)\frac{d\ln(Cm) + t}{n},$$

*where* $K(d, m, \beta) = \sqrt{2\left(\frac{9}{\sqrt{n}} + 2\right)\left(\frac{d+3}{3d}\right) + 1} + \left(\frac{9}{\sqrt{n}} + 2\right) + \left(\frac{d+3}{3d}\right) + 1 + \frac{1}{2\beta}$.

Note that $K(d, m, \beta)$ is non-increasing in $d, m$ as a consequence of which we immediately have the following corollary, which we use in the statement of our main theorem for fast rates.

**Corollary 1.** *Let $\mathcal{Q}$ be as above. For $d \geq 20$, $m \geq 5000$ and $\beta = 0.1$, we have with probability at least $1 - \exp(-t)$ for all functions $f \in \mathcal{Q}$,*

$$Ef \leq (1.1)E_n f + 9\frac{d\ln(Cm) + t}{n}.$$

The proofs of Lemma 1 and Lemma 2 could be found in [29].

## References


[1] P.L. Bartlett, O. Bousquet, S. Mendelson, Local Rademacher complexities, Ann. Stat. (2005).
[2] S. Bengio, F. Pereira, Y. Singer, D. Strelow, Group sparse coding, in: Annual Conference on Neural Information Processing Systems, 2009.
[3] D. Bertsekas, On the Goldstein–Levitin–Polyak gradient projection method, IEEE Trans. Autom. Control (1976).
[4] A. Bronstein, P. Sprechmann, G. Sapiro, Learning efficient structured sparse models, in: International Conference on Machine Learning, 2012.
[5] C. Chang, C. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. (2011).
[6] T. Cour, B. Sapp, C. Jordan, B. Taskar, Learning from ambiguously labeled images, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009.
[7] L. Devroye, G. Lugosi, Combinatorial Methods in Density Estimation, Springer, 2001.
[8] J. Fan, J. Lv, Sure independence screening for ultrahigh dimensional feature space, J. R. Stat. Soc., Ser. B, Stat. Methodol. (2008).
[9] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories, Comput. Vis. Image Underst. (2007).
[10] C.R. Genovese, J. Jin, L. Wasserman, Z. Yao, A comparison of the lasso and marginal regression, J. Mach. Learn. Res. (2012).
[11] T. Hastie, C. Loader, Local regression: automatic kernel carpentry, Stat. Sci. (1993).
[12] J. Huang, T. Zhang, D. Metaxas, Learning with structured sparsity, J. Mach. Learn. Res. (2011).
[13] R. Jenatton, J. Mairal, G. Obozinski, F. Bach, Proximal methods for sparse hierarchical dictionary learning, in: International Conference on Machine Learning, 2010.
[14] M. Kim, S. Kumar, V. Pavlovic, H. Rowley, Face tracking and recognition with visual constraints in real-world videos, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008.
[15] A. Kläser, M. Marszałek, C. Schmid, A spatio-temporal descriptor based on 3d-gradients, in: British Machine Vision Conference, 2008.
[16] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: spatial pyramid matching for recognizing natural scene categories, in: IEEE Conference on Computer Vision and Pattern Recognition, 2006.
[17] H. Lee, A. Battle, R. Raina, A.Y. Ng, Efficient sparse coding algorithms, in: Annual Conference on Neural Information Processing Systems, 2007.
[18] J. Liu, J. Luo, M. Shah, Recognizing realistic actions from videos in the wild, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009.
[19] C. Loader, Local Regression and Likelihood, Springer, 1999.
[20] D.G. Lowe, Object recognition from local scale-invariant features, in: IEEE Conference on Computer Vision and Pattern Recognition, 1999.
[21] L. Meier, P. Bühlmann, Smoothing l1-penalized estimators for high-dimensional time-course data, Electron. J. Stat. 1 (2007) 597–615.
[22] R. Raina, A. Battle, H. Lee, B. Packer, A.Y. Ng, Self-taught learning: transfer learning from unlabeled data, in: International Conference on Machine Learning, 2007.
[23] I. Ramırez, F. Lecumberry, G. Sapiro, Sparse modeling with universal priors and learned incoherent dictionaries, Tech report, IMA, University of Minnesota, 2009.

[24] C. Schüldt, I. Laptev, B. Caputo, Recognizing human actions: a local SVM approach, in: International Conference on Pattern Recognition, 2004.
[25] C.D. Sigg, D. Dikk, J.M. Buhmann, Learning dictionaries with bounded self-coherence, IEEE Trans. Signal Process. (2012).
[26] M.V. Solodov, Convergence analysis of perturbed feasible descent methods, J. Optim. Theory Appl. (1997).
[27] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, K. Knight, Sparsity and smoothness via the fused lasso, J. R. Stat. Soc., Ser. B, Stat. Methodol. 67 (2005).
[28] J.A. Tropp, Greed is good: algorithmic results for sparse approximation, IEEE Trans. Inf. Theory (2004).
[29] D. Vainsencher, S. Mannor, A.M. Bruckstein, The sample complexity of dictionary learning, J. Mach. Learn. Res. (2011).
[30] M.P. Wand, M.C. Jones, Kernel Smoothing, CRC Press, 1994.
[31] H. Wang, M.M. Ullah, A. Klaser, I. Laptev, C. Schmid, Evaluation of local spatio-temporal features for action recognition, in: British Machine Vision Conference, 2009.
[32] J. Yang, K. Yu, T. Huang, Supervised translation-invariant sparse coding, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010.
[33] K. Yu, Y. Lin, J. Lafferty, Learning image representations from the pixel level via hierarchical sparse coding, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011.
[34] K. Yu, T. Zhang, Y. Gong, Nonlinear learning using local coordinate coding, in: Annual Conference on Neural Information Processing Systems, 2009.
[35] W.I. Zangwill, Nonlinear Programming: A Unified Approach, Prentice-Hall International, 1969.