

# Design and Generalization Analysis of Orthogonal Matching Pursuit Algorithms

Zakria Hussain, John Shawe-Taylor, *Member, IEEE*, David R. Hardoon, and Charanpal Dhanjal

**Abstract**—We derive generalization error (loss) bounds for orthogonal matching pursuit algorithms, starting with kernel matching pursuit and sparse kernel principal components analysis. We propose (to the best of our knowledge) the first loss bound for kernel matching pursuit using a novel application of sample compression and Vapnik-Chervonenkis bounds. For sparse kernel principal components analysis, we find that it can be bounded using a standard sample compression analysis, as the subspace it constructs is a compression scheme. We demonstrate empirically that this bound is tighter than previous state-of-the-art bounds for principal components analysis, which use global and local Rademacher complexities. From this analysis we propose a novel sparse variant of kernel canonical correlation analysis and bound its generalization performance using the results developed in this paper. We conclude with a general technique for designing matching pursuit algorithms for other learning domains.

**Index Terms**—Kernel methods, matching pursuit, Nyström approximation, principle components analysis, sample compression bounds, sparse kernel canonical correlation analysis, sparsity.

## I. INTRODUCTION

**M**ATCHING pursuit is a family of algorithms that look for a sparse set of bases in a greedy fashion. It was first proposed in the signal processing literature [2], [3] and has recently received considerable attention in machine learning, including [4]–[7]. In this paper, we follow this framework and, to our knowledge, derive the first generalization error bounds for matching pursuit. We also derive new learning algorithms using this approach.

We start by analyzing kernel matching pursuit (KMP); a sparse regression algorithm proposed by Vincent and Bengio [6]. We propose a generalization error analysis (Theorem 4) to upper bound its future loss, using sparsity as a measure

of complexity. We are not aware of any generalization error bounds for kernel matching pursuit. The difficulty in bounding this algorithm in terms of sparsity is the fact that it gives rise to *dual* sparsity (i.e., sparsity in the kernel representation)—and hence requires all of the training inputs in order to construct the final regression function. In our analysis we provide a novel way of combining VC bounds [8], [9] together with sample compression bounds (bounds using sparsity as a measure of complexity). The main technique here relies on noticing that the  $k$  basis vectors chosen by kernel matching pursuit have VC dimension  $k$ , i.e., they are a combination of  $k$  one-dimensional linear threshold functions. From this it is easy to see that we can bound the kernel matching pursuit algorithm by making a further  $\binom{m}{k}$  applications of this VC bound (using a sample compression analysis), which takes care of the  $k$  different linear threshold functions that can be found from a set of  $m \gg k$  linear threshold functions. This simple, yet novel analysis technique is one we have not seen in the literature.

Next we look to analyze sparse kernel principal components analysis (SKPCA) proposed by [5]. We show that this algorithm is related to matching pursuit, and is in fact a matching pursuit style algorithm. However, unlike kernel matching pursuit the analysis of this algorithm is amenable to traditional sample compression bounds [10], [11]—a data-dependent bound which can be found if the final hypothesis can be constructed using *only* a subsample of data points. In order to apply a sample compression bound we must show that an algorithm forms a *compression scheme* [10]–[13]. We prove that sparse kernel principal components analysis is a sample compression scheme (Claim 1) and give a bound (Theorem 6) to upper bound the loss between the original data points and the projections into the sparse kernel principal components analysis. This is the first analysis showing that this algorithm is a compression scheme, and also the first sample compression bound for a domain other than classification and regression. Furthermore, we show our bound to be empirically tighter than two state-of-the-art bounds for kernel principal components analysis, which use Rademacher complexity [14] and local Rademacher complexities [15]. Neither of these bounds take into account the sparsity element of the sparse kernel principal components analysis. Therefore, our results suggests that analyzing sparse algorithms using sparsity as a measure of complexity may result in tighter bounds than more general results.

Given these two analyses and by applying matching pursuit techniques more broadly, we show that other algorithms can have sparse counterparts. Hence, as the title suggests, we design a new sparse algorithm for kernel canonical correlation analysis (Algorithm 3) [16]–[18]. This is the first matching pursuit style algorithm we know of for kernel canonical correlation analysis, and is amenable to the same bounding principle (Theorem 9) as

Manuscript received October 15, 2008; revised March 27, 2011; accepted April 25, 2011. Date of current version July 29, 2011. This work was supported in part by the EU project SMART and in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. The material in this paper was presented at the Neural Information Processing Systems Conference, Vancouver, BC, Canada, December 2008.

Z. Hussain and J. Shawe-Taylor are with the Department of Computer Science, University College London, London, U.K., WC1E 6BT (e-mail: z.hussain@cs.ucl.ac.uk; jst@cs.ucl.ac.uk; jst@ecs.soton.ac.uk).

D. R. Hardoon is with the SAS Singapore and also with the Department of Computer Science, University College London, London, U.K., WC1E 6BT (e-mail: davidrh@me.com).

C. Dhanjal is with the Department of Image and Signal Processing, Telecom ParisTech, Paris, France, and also with the School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, U.K. (e-mail: cd04r@ecs.soton.ac.uk).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Communicated by A. Krzyzak, Associate Editor for Pattern Recognition, Statistical Learning, and Inference.

Digital Object Identifier 10.1109/TIT.2011.2158880

that constructed for kernel matching pursuit, albeit with a different loss function. Furthermore, we also show empirical results on a machine translation problem which shows good behavior against nonsparse kernel canonical correlation analysis.

We finish with a discussion of generic matching pursuit algorithms which can be obtained by carrying out:

- 1) Function maximization; *and*
- 2) Deflation (i.e., orthogonalization).

We discuss applications to other learning domains and also give an example of using this generic algorithm to solve the problem of classification (see Section V). This algorithm is also be amenable to the generalization analysis presented in this paper.

In Section II we discuss the relationship of our results to other work.

## II. RELATED WORK

As aforementioned, we are not aware of any bounds for kernel matching pursuit or sparse kernel principal components analysis. The main issue with bounding KMP is that although it is sparse, it is only sparse in the kernel Reproducing Kernel Hilbert Space (RKHS) and so we *cannot* apply the vanilla sample compression analysis as all of the data points are required in order to reconstruct the final weight vector. It is for this very reason we employ the sample compression argument over a VC bound. Work by Koltchinskii [19] based on Oracle inequalities, and their empirical counterparts called local Rademacher complexities [20], have been published and scale favorably between  $\mathcal{O}\left(\frac{1}{m}\right)$  and  $\mathcal{O}\left(\frac{1}{\sqrt{m}}\right)$ , where  $m$  denotes the size of the sample. We are not aware of any local Rademacher bounds for kernel matching pursuit. However, Blanchard *et al.* [15] have proposed local Rademacher bounds for principal components analysis, showing their bound to be tighter *asymptotically* than the global Rademacher bound of Shawe-Taylor *et al.* [14]. We compare our sample compression bound for SKPCA against this local Rademacher bound and find it to be tighter for a finite sample—however, as  $m \rightarrow \infty$  then the local Rademacher bound becomes tighter. We demonstrate empirically that this does not: (a) help find the basis vectors which capture most of the variance and (b) does not use sparsity as a measure of complexity.

For the algorithmic developments of this paper: namely sparse kernel canonical correlation analysis there have recently been some sparse canonical correlation analysis algorithms proposed in [21]–[24]. However, they are not based on the matching pursuit principle, do not cover dual sparsity in kernel defined feature spaces and also fail to present any generalization analysis. Therefore, our contribution is different on several fronts.

## III. GENERALIZATION ERROR ANALYSIS

We start with the main motivation of our results by describing kernel matching pursuit. We present the low rank matrix approximation algorithm of [5] in Appendix A. This is done to show the relation to our own derivation of the algorithm and to show that it equates to a matching pursuit version of kernel principal components analysis, which forms a compression scheme. We prove that this derivation is equivalent to the low rank matrix approximation algorithm. Also, note that most of the algorithms

described are in kernelized form (see [25] and [26] for introductions to kernels) due to kernel variants being more practically effective at solving nonlinear problems. Nevertheless the bounds presented are applicable to the linear versions of all the algorithms presented in this paper.

### A. Preliminaries

Before beginning our analysis, we state the following Theorem and Definition, of a VC bound and a sample compression scheme, respectively, that will form the basis of our theoretical analysis throughout the paper. Let  $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$  be a training set of  $m$  input-output pairs  $(\mathbf{x}, y)$ , where each  $\mathbf{x} \in \mathbb{R}^n$  and  $y \in \{-1, +1\}$  when dealing with classification and  $y \in \mathbb{R}$  when looking at regression. In this paper, we deal exclusively with regression, however, in order to state the VC bound we need to define the following classification loss  $\ell(\cdot)$  given a function  $f$ . We will assume throughout that  $f$  is found using some learning algorithm  $\mathcal{A}$  when given a sample  $S$ , such that  $f = \mathcal{A}(S)$ .

*Definition 1 (True Loss):* Let  $\mathcal{D}$  be a distribution over the generation of an  $m$  sample  $S$ , and let  $f : \mathbf{x} \mapsto \{-1, +1\}$ , then the *true loss* of  $f$  is defined as

$$\ell(f) = \Pr_{(\mathbf{x}, y) \sim \mathcal{D}} \{|f(\mathbf{x}) - y| > 0\}.$$

Typically,  $\mathcal{D}$  is unknown, so we use the following empirical estimate instead.

*Definition 2 (Empirical Loss):* Given an  $m$  sample  $S$ , and a function  $f : \mathcal{X} \mapsto \{-1, +1\}$  found using  $S$ , we define the *empirical loss*  $\hat{\ell}(f)$  of  $f$  as

$$\begin{aligned} \hat{\ell}(f) &= \Pr_{(\mathbf{x}, y) \sim S} \{|f(\mathbf{x}) - y| > 0\} \\ &= \frac{1}{m} \sum_{i=1}^m \mathbb{1}(|f(\mathbf{x}_i) - y_i| > 0) \end{aligned}$$

where  $\mathbb{1}(a) = 1$  if  $a$  is true and 0 otherwise.

*Theorem 1 (Vapnik-Chervonenkis Bound):* Let  $S$  be a training sample,  $m$  the number of samples in  $S$  and  $\mathcal{E} \in \mathbb{R}$ . Then for sample  $S$ , the probability of observing zero training error  $\hat{\ell}(f)$  but true error  $\ell(f)$  greater than some  $\mathcal{E}$  can be upper bounded by

$$P^m \{S : \hat{\ell}(f) = 0, \ell(f) > \mathcal{E}\} \leq 2B_{\mathcal{H}}(2m)2^{-\mathcal{E}m/2}$$

where

$$B_{\mathcal{H}}(m) = \max_{(\mathbf{x}_1, \dots, \mathbf{x}_m) \in \mathcal{X}^m} |\{h(\mathbf{x}_1), \dots, h(\mathbf{x}_m) : h \in \mathcal{H}\}|$$

is the growth function over  $m$  points.

The following definition is for a sample compression scheme first proposed by [10].

*Definition 3 (Compression Scheme):* The *compression function*  $\Lambda$  induced by a sample compression algorithm  $\mathcal{A}$  on training set  $S$  is the map

$$\Lambda : S \mapsto \Lambda(S)$$

such that the *compression set*  $\Lambda(S) \subset S$  is returned by  $\mathcal{A}$ .

A reconstruction function  $\Phi$  is a mapping from a compression set  $\Lambda(S)$  to a set  $\mathcal{H}$  of hypotheses

$$\Phi : \Lambda(S) \mapsto f \in \mathcal{H}.$$

Let  $\mathcal{A}(S)$  be the function output by learning algorithm  $\mathcal{A}$  on training set  $S$ . A sample compression scheme is a reconstruction function  $\Phi$  mapping a compression set  $\Lambda(S)$  to some set of functions  $\mathcal{H}$  such that

$$\mathcal{A}(S) = \Phi(\Lambda(S)).$$

If  $\mathcal{H}$  is the set of Boolean-valued functions then the sample compression scheme is said to be a classification algorithm. If  $\mathcal{H}$  is the set of Real-valued functions then the sample compression scheme is a regression algorithm.

Given that an algorithm is a compression scheme we can prove the following bound—first proved by [10], [11]—but stated here in the form given by [27].

*Theorem 2 (Sample Compression Bound):* Let  $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$  be a sample and  $\mathcal{A}$  an algorithm which induces a sample compression scheme such that  $\mathcal{A}(S) = \Phi(\Lambda(S))$ . Without loss of generality let  $S$  be reordered so that the first  $k$  points form the compression set  $S_1 = \Lambda(S)$ , the next  $t$  points form the error set  $S_2$  such that  $\mathbb{1}(|f(\mathbf{x}_i) - y_i| > 0) = 1$  for all  $i = k+1, \dots, k+t$ , and let  $\bar{S} = S \setminus (S_1 \cup S_2)$  be the remaining data points. Then for all choices of the indices of  $S_1$  and  $S_2$ , with probability  $1 - \delta$  over the generation of the remaining data  $\bar{S}$  we have the expected loss  $\mathcal{E}[\ell(\mathcal{A}(S))]$ , for  $t = 0$

$$\mathcal{E}[\ell(\mathcal{A}(S))] \leq \frac{1}{m-k} \left[ \ln \binom{m}{k} + \ln \left( \frac{m}{\delta} \right) \right]$$

and for  $t > 0$

$$\mathcal{E}[\ell(\mathcal{A}(S))] \leq \frac{1}{m-k-t} \left[ \ln \binom{m-k}{t} + \ln \binom{m}{k} + \ln \left( \frac{m^2}{\delta} \right) \right].$$

Hence, if there exists an algorithm which forms a small compression set  $k$  and has small empirical error  $t$  then it will result in a tight bound. Note that the sample compression bounds use a union bound argument over the choice of the compression set and error set, and hold with high probability over all such choices.

All of the above results are for classification. In the kernel matching pursuit analysis we will convert the regression loss of KMP into a classification loss in order to apply the above results. In the regression analysis of sparse kernel principal components analysis we will require the following well-known result.

*Theorem 3 (Hoeffding's Inequality):* If  $x_1, \dots, x_n$  are independent random variables satisfying  $x_i \in [a_i, b_i]$  (meaning  $x_i$  is in the interval between fixed constants  $a_i$  and  $b_i$ ) and if we define the sum of these random variables as  $s_n = \sum_{i=1}^n x_i$ , then it follows that:

$$\Pr\{|s_n - \mathcal{E}[s_n]| \geq \mathcal{E}\} \leq 2 \exp \left( - \frac{2\mathcal{E}^2}{\sum_{i=1}^n (b_i - a_i)^2} \right)$$

where  $\mathcal{E}[\cdot]$  denotes the expectation.

## B. Kernel Matching Pursuit

Let  $(\mathbf{x}, y) \sim S$  be an input-output pair from an  $m$ -sample  $S$  with  $\mathbf{x} \in \mathbb{R}^n$  an  $n$ -dimensional vector and  $y \in \mathbb{R}$ . Given a matrix of inputs  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_m)$  with the corresponding outputs  $\mathbf{y} = (y_1, \dots, y_m)'$ , where  $(\cdot)'$  denotes the transpose of a vector, least squares regression finds the weight vector  $\mathbf{w}$  that minimizes the following quantity:

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}'\mathbf{w}\|^2.$$

Let  $\mathbf{i} = (i_1, \dots, i_k)$  be a vector containing indices (integer valued numbers) where  $k < m$ . Matching pursuit [2] looks to find  $\mathbf{i}$ , i.e., a small number of dimensions (basis vectors) in  $\mathbf{X}$ , in a greedy fashion such that the following quantity is minimized:

$$\min_{\tilde{\mathbf{w}}} \|\mathbf{y} - \mathbf{X}[\mathbf{i}, :]'\tilde{\mathbf{w}}\|^2$$

where  $\mathbf{X}[\mathbf{i}, :]$  denotes the dimensions (features/attributes) indexed by vector  $\mathbf{i}$ . The aim is to find  $|\mathbf{i}| = k < m$  small but still able to generate a good  $\tilde{\mathbf{w}}$  by only using the dimensions indexed by  $\mathbf{i}$ . A measure for the “good”-ness of  $\tilde{\mathbf{w}}$  can be:

- 1) how far is the  $\tilde{\mathbf{w}}$  generated by  $\mathbf{i}$  from the  $\mathbf{w}$  generated by the full dimensionality (stability);
- 2) how well does the regression function computed from  $\tilde{\mathbf{w}}$  generalize in the future (generalization error bounds).

In this paper we concern ourselves with the second property as the first has been extensively studied by [28]–[31] for basis pursuit [32]<sup>1</sup> and recently for the more general and related framework of compressed sensing [33], [34].

Vincent and Bengio [6] have proposed a kernel variant of (orthogonal) matching pursuit [3]<sup>2</sup> where by writing  $\mathbf{w} = \mathbf{X}[:, \mathbf{i}]\boldsymbol{\alpha}$  as a linear combination of the training examples, and substituting into the least squares regression problem they obtain the following minimization problem:

$$\min_{\tilde{\boldsymbol{\alpha}}} \|\mathbf{y} - \mathbf{X}'\mathbf{X}[:, \mathbf{i}]\tilde{\boldsymbol{\alpha}}\|^2 = \min_{\tilde{\boldsymbol{\alpha}}} \|\mathbf{y} - \mathbf{K}[:, \mathbf{i}]\tilde{\boldsymbol{\alpha}}\|^2$$

where  $\mathbf{K}[:, \mathbf{i}] = \mathbf{X}'\mathbf{X}[:, \mathbf{i}]$  is a rectangular kernel matrix defined by  $\mathbf{i}$ . For simplicity we always assume that the examples are already projected into the kernel defined feature space, so that the kernel matrix  $\mathbf{K}$  has entries  $\mathbf{K}[i, j] = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ . The solution for the minimization problem above, for a fixed  $\mathbf{i}$ , is

$$\tilde{\boldsymbol{\alpha}} = \mathbf{K}[:, \mathbf{i}]'\mathbf{K}[:, \mathbf{i}]^{-1}\mathbf{K}[:, \mathbf{i}]\mathbf{y} \quad (1)$$

where we assume the inverse  $\mathbf{K}[:, \mathbf{i}]'\mathbf{K}[:, \mathbf{i}]^{-1}$  exists. Note that in the kernel version  $\mathbf{i}$  denotes the indices of the training examples. Once we have (1) we can make predictions with the KMP regressor  $f_{kmp}$ , for a new example  $\mathbf{x}_j$  like so

$$f_{kmp}(\mathbf{x}_j) = \mathbf{K}[j, \mathbf{i}]\tilde{\boldsymbol{\alpha}}$$

where  $\mathbf{K}[j, \mathbf{i}]$  represents the kernel functions between  $\mathbf{x}_j$  and the examples from the training set indexed by vector  $\mathbf{i}$ . The pseudocode for KMP is given in Algorithm 1. In all the algorithms

<sup>1</sup>Basis pursuit essentially solves the same problem as matching pursuit using a linear program.

<sup>2</sup>Orthogonal matching pursuit updates the entire weight vector after each greedy step using orthogonalization, and can therefore be viewed as a sparse least squares regression algorithm.

to follow, division will be assumed component wise for vectors and the notation  $\mathbf{X} \cdot \mathbf{Y}$  will denote component wise multiplication of matrices  $\mathbf{X}$  and  $\mathbf{Y}$  and  $\mathbf{X} \cdot \mathbf{X} = \mathbf{X}^{\cdot 2}$  will denote component wise squaring.

---

**Algorithm 1:** Kernel matching pursuit with prefitting [6]
 

---

**Input:** kernel  $\mathbf{K}$ , sparsity parameter  $k > 0$ .

1: initialize  $\mathbf{A} = ()$  and  $\mathbf{i} = ()$

2: **for**  $i = 1$  to  $k$  **do**

3: set  $\mathbf{i}_i$  to the index of  $\max \left| \frac{(\mathbf{K}\mathbf{y})}{\sqrt{(\mathbf{K}^{\cdot 2})\mathbf{1}}} \right|$

4:  $\tilde{\boldsymbol{\alpha}}_i = (\mathbf{K}[:, \mathbf{i}_i]' \mathbf{K}[:, \mathbf{i}_i])^{-1} \mathbf{K}[:, \mathbf{i}_i]' \mathbf{y}$

5:  $(\tilde{\boldsymbol{\alpha}}_1, \dots, \tilde{\boldsymbol{\alpha}}_{i-1})' = (\tilde{\boldsymbol{\alpha}}_1, \dots, \tilde{\boldsymbol{\alpha}}_{i-1})' - \tilde{\boldsymbol{\alpha}}_i \mathbf{A}[:, \mathbf{i}_i]$  (if  $\mathbf{A}$  is nonempty)

6: set  $\boldsymbol{\tau} = \mathbf{K}[:, \mathbf{i}_i]$  and  $\mathbf{p} = \frac{\boldsymbol{\tau}' \mathbf{K}}{\boldsymbol{\tau}' \boldsymbol{\tau}}$ , then update (if  $\mathbf{A}$  is nonempty)

$$\mathbf{A} = \mathbf{A} - \mathbf{A}[:, \mathbf{i}_i] \mathbf{p}$$

7: deflate kernel matrix

$$\begin{aligned} \mathbf{K} &= \mathbf{K} - \mathbf{K}[:, \mathbf{i}_i] \mathbf{p} \\ &= \left( \mathbf{I} - \frac{\boldsymbol{\tau} \boldsymbol{\tau}'}{\boldsymbol{\tau}' \boldsymbol{\tau}} \right) \mathbf{K} \end{aligned}$$

8: add row vector  $\mathbf{p}$  to  $\mathbf{A}$  matrix

$$\mathbf{A} = \begin{pmatrix} \mathbf{A} \\ \mathbf{p}_1, \dots, \mathbf{p}_m \end{pmatrix}$$

9: set  $\mathbf{K}[:, \mathbf{i}_i] = 0$

10: **end for**

**Output:** index vector  $\mathbf{i}$  and sparse dual weight vector  $\tilde{\boldsymbol{\alpha}}$

Before proceeding with the analysis of KMP we would like to clarify several aspects regarding the algorithm. Line 3 of Algorithm 1 is the function to maximize—obtained by minimizing the loss of the residual.<sup>3</sup> Line 7 corresponds to deflating the kernel matrix so that the remaining basis vectors are far from those already chosen. Lines 4, 5, 6 (corresponding to matrix  $\mathbf{A}$ ) and 8 are only needed to make rank 1 updates of the  $\tilde{\boldsymbol{\alpha}}$  vector. We would achieve the same algorithm by removing these lines and using the subspace defined by the final  $\mathbf{i}$  to compute  $\tilde{\boldsymbol{\alpha}}$  from (1). This would make the algorithm slightly less efficient but much clearer to interpret. Hence, all other matching pursuit algorithms we describe in this paper will use this simpler (and more general) form.

To our knowledge there does not exist any generalization error bounds for KMP. The fact that KMP does not form a compression scheme means that we cannot simply bound it in terms of the compression set. Furthermore, VC bounds are loose in high dimensional feature spaces. However, we show that an

<sup>3</sup>The residual in this context is the difference between the regression output and the prediction made by the KMP algorithm.

amalgamation of both these theories allows us to bound the generalization error of KMP. The main intuition is that the basis vectors chosen for KMP have equivalent complexity to the set of linear threshold functions and hence their VC dimensions are equivalent to the cardinality of the low dimensional feature space constructed by KMP. Therefore the VC dimension of KMP is simply the number of basis vectors chosen. With this insight we can use a VC bound and count the number of ways of choosing this subspace—a counting principle used in compression schemes—to upper bound the loss of KMP, whilst avoiding the need for the double sample trick and introducing a novel application of VC bounds to high dimensional spaces.

Before we begin we point out that our choice of loss function for the bounds will be the 0/1 loss.

*Definition 4:* Let  $S \sim \mathcal{D}$  be a regression training sample generated iid from a fixed but unknown probability distribution  $\mathcal{D}$ . Given the error  $\ell(f) = |f(\mathbf{x}) - y|$  for a regression function  $f$  between training example  $\mathbf{x}$  and regression output  $y$  we can define, for some fixed positive scalar  $\alpha \in \mathbb{R}$ , the corresponding true classification loss (error) as

$$\ell_\alpha(f) = \Pr_{(\mathbf{x}, y) \sim \mathcal{D}} \{|f(\mathbf{x}) - y| > \alpha\}.$$

Similarly, we can define the corresponding empirical classification loss as

$$\begin{aligned} \hat{\ell}_\alpha(f) &= \ell_\alpha^S(f) = \Pr_{(\mathbf{x}, y) \sim S} \{|f(\mathbf{x}) - y| > \alpha\} \\ &= \mathbb{E}_{(\mathbf{x}, y) \sim S} \{\mathbb{1}(|f(\mathbf{x}) - y| > \alpha)\} \end{aligned}$$

where  $\mathbb{1}$  is the indicator function and  $S$  is suppressed when clear from context.

Let  $\mathcal{X}$  be the set of  $m$  vectors  $\mathbf{x}$ . Given all the above definitions we state the following main result of this section.<sup>4</sup>

*Theorem 4:* Fix  $0 < \alpha \in \mathbb{R}$ . Let  $\mathcal{A}(S) = f$  be the function output by the KMP algorithm  $\mathcal{A}$  when given an  $m$  sample  $S$ . Without loss of generality let  $S$  be reordered so that the first  $k$  points form the compression set  $S_1$  and the next  $t$  points form the training error set  $S_2$  such that  $\mathbb{1}(|f(\mathbf{x}_i) - y_i| > \alpha) = 1$  for all  $i = k + 1, \dots, k + t$ . Then for all choices of the indices of  $S_1$  and  $S_2$ , with probability  $1 - \delta$  over the generation of the remaining data  $\tilde{S} = S \setminus (S_1 \cup S_2)$  we have the following upper bound on the expected loss

$$\begin{aligned} \mathcal{E}[\ell_\alpha(\mathcal{A}(S))] &\leq \frac{2}{m - k - t} \left[ (k + 1) \log \left( \frac{4e(m - k - t)}{k + 1} \right) \right. \\ &\quad \left. + k \log \left( \frac{em}{k} \right) + t \log \left( \frac{e(m - k)}{t} \right) + \log \left( \frac{2m^2}{\delta} \right) \right]. \end{aligned}$$

*Proof:* First consider a fixed size  $k$  for the compression set and number of errors  $t$ . Let  $S_1 = \{\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k}\}$  be the set of  $k$  training points chosen by the KMP regressor,  $S_2 = \{\mathbf{x}_{i_{k+1}}, \dots, \mathbf{x}_{i_{k+t}}\}$  the set of points erred on in training and  $\tilde{S} = S \setminus (S_1 \cup S_2)$  the points outside of the compression set ( $S_1$ ) and training error set ( $S_2$ ). Suppose that the first  $k$  points form the compression set and the next  $t$  are the errors of the KMP regressor. Since the remaining  $m - k - t$  points  $\tilde{S}$  are drawn

<sup>4</sup>Preliminary work appeared at the NIPS conference 2008 [1].

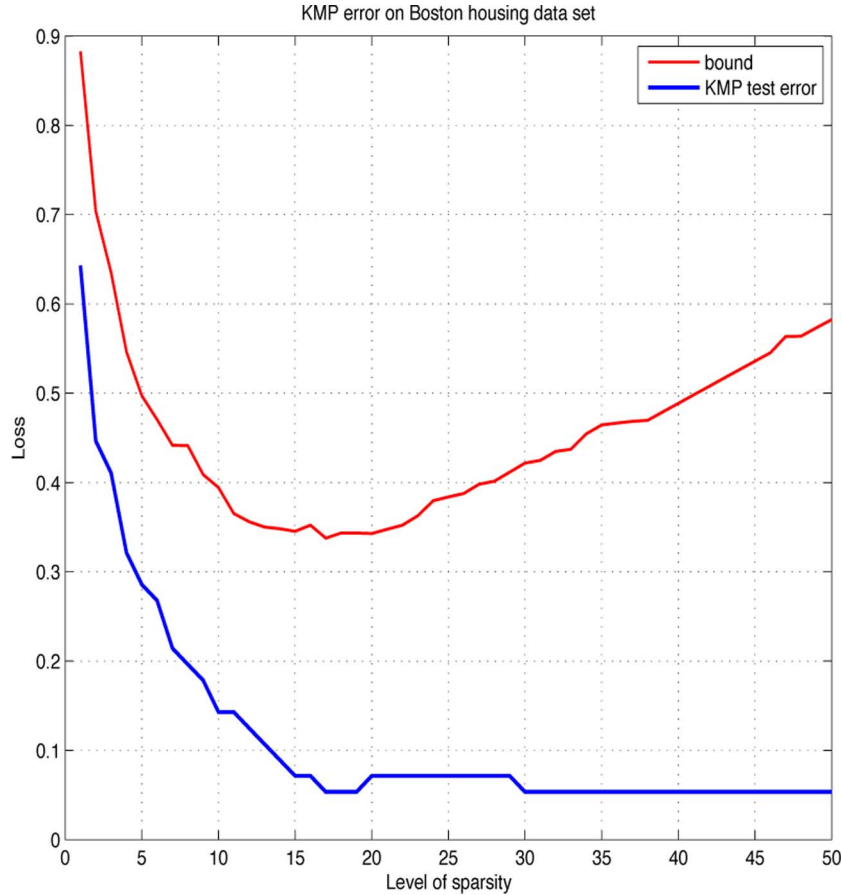


Fig. 1. Plot for KMP bound vs KMP test error.

independently we can apply Theorem 1 to the  $\ell_\alpha$  loss to obtain the bound

$$\Pr \left\{ \bar{S} : \ell_\alpha^{\bar{S}}(f) = 0, \ell_\alpha(f) > \mathcal{E} \right\} \leq 2 \left( \frac{4e(m-k-t)}{k+1} \right)^{k+1} 2^{-\mathcal{E}(m-k-t)/2}$$

where we have made use of a bound on the number of dichotomies that can be generated by parallel hyperplanes due to [35], that states that  $\sum_{i=0}^{k+p-1} \binom{mp-1}{i} \leq \left( \frac{e(mp-1)}{k+p-1} \right)^{k+p-1}$ , where  $k$  is the number of parallel hyperplanes and equals 2 in our case. We now need to consider all of the ways that the  $k$  basis vectors and  $t$  error points might have occurred and apply the union bound over all of these possibilities. This gives the bound

$$\Pr \left\{ S : \exists f \in \text{span}\{S_1\} \text{ s.t. } \ell_\alpha^{S_2}(f) = 1, \ell_\alpha^{\bar{S}}(f) = 0, \ell_\alpha(f) > \mathcal{E} \right\} \leq \binom{m}{k} \binom{m-k}{t} 2 \left( \frac{4e(m-k-t)}{k+1} \right)^{k+1} 2^{-\mathcal{E}(m-k-t)/2}. \quad (2)$$

Finally we need to consider all possible choices of the values of  $k$  and  $t$ . The number of these possibilities is clearly upper bounded by  $m^2$ . Setting  $m^2$  times the right-hand side (RHS) of (2) equal to  $\delta$  and solving for  $\mathcal{E}$  gives the result. ■

This result is unique in the sense that although it uses the VC bound, it does so without the extra complexity of the feature space. It is unaffected by infinite dimensional features spaces such as those defined by the Gaussian kernel, and hence we can generate upper bounds in this scenario. The plot below depicts such a situation.

The bound depicted in Fig. 1 has been scaled down by a factor of 5 to show the plots at similar scales. However, the bound seems to predict well the lowest point of the test error, suggesting that the algorithm could be driven by the bound.

### C. Sparse Kernel Principal Components Analysis

Principal components analysis [25] can be expressed as the following maximization problem:

$$\max_{\mathbf{w}_x} \frac{\mathbf{w}'_x \mathbf{X} \mathbf{X}' \mathbf{w}_x}{\mathbf{w}'_x \mathbf{w}_x} \quad (3)$$

where  $\mathbf{w}_x$  is the weight vector. In a sparse PCA algorithm we may want to find a sparsely represented vector  $\mathbf{w}_x = \mathbf{X}[:, \mathbf{i}] \tilde{\alpha}_x$ , that is a linear combination of a small number of training examples indexed by vector  $\mathbf{i}$ . This corresponds to projections being made into a sparse subspace maximizing the variance of the data. By making the substitution  $\mathbf{w}_x = \mathbf{X}[:, \mathbf{i}] \tilde{\alpha}_x$  into (3) we have the following sparse dual PCA maximization problem:

$$\max_{\tilde{\alpha}_x} \frac{\tilde{\alpha}'_x \mathbf{X}[:, \mathbf{i}]' \mathbf{X} \mathbf{X}' \mathbf{X}[:, \mathbf{i}] \tilde{\alpha}_x}{\tilde{\alpha}'_x \mathbf{X}[:, \mathbf{i}]' \mathbf{X}[:, \mathbf{i}] \tilde{\alpha}_x}$$

which is equivalent to sparse kernel PCA (SKPCA) with  $\mathbf{K}[:, \mathbf{i}] = \mathbf{X}'\mathbf{X}[:, \mathbf{i}]$

$$\begin{aligned}\tilde{\lambda}_x &= \max_{\mathbf{i}, \tilde{\alpha}_x} \frac{\tilde{\alpha}'_x \mathbf{K}[:, \mathbf{i}]' \mathbf{K}[:, \mathbf{i}] \tilde{\alpha}_x}{\tilde{\alpha}'_x \mathbf{K}[\mathbf{i}, \mathbf{i}] \tilde{\alpha}_x} \\ &= \max_{\mathbf{i}, \tilde{\alpha}_x} \frac{\tilde{\alpha}'_x \mathbf{K}^2[\mathbf{i}, \mathbf{i}] \tilde{\alpha}_x}{\tilde{\alpha}'_x \mathbf{K}[\mathbf{i}, \mathbf{i}] \tilde{\alpha}_x}\end{aligned}$$

where  $\tilde{\alpha}_x$  is a sparse vector of length  $k = |\mathbf{i}|$ . Clearly maximizing the quantity above will lead to the maximization of the generalized eigenvalues  $\tilde{\lambda}_x$  corresponding to  $\tilde{\alpha}_x$ —and hence a sparse subset of the original PCA problem.

We would like to find a set of indices  $\mathbf{i}$ , and proceed in a greedy manner (matching pursuit) using a similar approach to [5]. The procedure involves choosing basis vectors that maximize the Rayleigh quotient without the set of eigenvectors. Choosing basis vectors iteratively until some prespecified number of  $k$  vectors are chosen. An orthogonalization of the kernel matrix at each step ensures future potential basis vectors will be orthogonal to those already chosen. We proceed by making the following substitution  $\mathbf{w}_x = \mathbf{X}\mathbf{e}_i$ , where  $\mathbf{e}_i$  is the  $i$ th unit vector, into the primal problem to obtain the following quotient maximization problem:

$$\begin{aligned}\max \rho_i &= \frac{\mathbf{e}'_i \mathbf{X}' \mathbf{X} \mathbf{X}' \mathbf{X} \mathbf{e}_i}{\mathbf{e}'_i \mathbf{X}' \mathbf{X} \mathbf{e}_i} \\ &= \frac{\mathbf{e}'_i \mathbf{K}^2 \mathbf{e}_i}{\mathbf{e}'_i \mathbf{K} \mathbf{e}_i} \\ &= \frac{\mathbf{K}[:, i]' \mathbf{K}[:, i]}{\mathbf{K}[i, i]}\end{aligned}\quad (4)$$

where  $\rho_i$  denotes the solution of quotient (4) computed using the  $i$ th training example.

After this maximization we need to orthogonalize (deflate) the kernel matrix to create a projection into the space orthogonal to the basis vectors chosen to ensure we find the maximum variance of the data in the projected space. The deflation step can be carried out as follows. Let  $\mathbf{K}[:, i] = \mathbf{X}'\mathbf{X}\mathbf{e}_i$ . We know that primal PCA deflation can be carried out with respect to the features in the following way:

$$\hat{\mathbf{X}} = \left( \mathbf{I} - \frac{\mathbf{u}\mathbf{u}'}{\mathbf{u}'\mathbf{u}} \right) \mathbf{X}$$

where  $\mathbf{u}$  is the projection direction defined by the chosen eigenvector,  $\hat{\mathbf{X}}$  is the deflated matrix and  $\mathbf{I}$  is the identity matrix. In SKPCA,  $\mathbf{u} = \mathbf{X}\mathbf{e}_i$  because the projection directions are simply the examples in  $\mathbf{X}$ . Hence

$$\begin{aligned}\hat{\mathbf{X}}'\hat{\mathbf{X}} &= \mathbf{X}' \left( \mathbf{I} - \frac{\mathbf{u}\mathbf{u}'}{\mathbf{u}'\mathbf{u}} \right) \left( \mathbf{I} - \frac{\mathbf{u}\mathbf{u}'}{\mathbf{u}'\mathbf{u}} \right) \mathbf{X} \\ &= \mathbf{X}' \left( \mathbf{I} - \frac{\mathbf{u}\mathbf{u}'}{\mathbf{u}'\mathbf{u}} \right) \mathbf{X} \\ &= \mathbf{X}'\mathbf{X} - \frac{\mathbf{X}'\mathbf{X}\mathbf{e}_i\mathbf{e}_i'\mathbf{X}'\mathbf{X}}{\mathbf{e}'_i\mathbf{X}'\mathbf{X}\mathbf{e}_i} \\ &= \mathbf{X}'\mathbf{X} - \frac{\mathbf{X}'\mathbf{X}[:, i]\mathbf{X}[:, i]'\mathbf{X}}{\mathbf{X}[:, i]'\mathbf{X}[:, i]} \\ &= \mathbf{K} - \frac{\mathbf{K}[:, i]\mathbf{K}[:, i]'}{\mathbf{K}[i, i]}\end{aligned}$$

Therefore, given a kernel matrix  $\mathbf{K}$  the deflated kernel matrix  $\hat{\mathbf{K}}$  can be computed as follows:

$$\hat{\mathbf{K}} = \mathbf{K} - \frac{\boldsymbol{\tau}\boldsymbol{\tau}'}{\mathbf{K}[\mathbf{i}_k, \mathbf{i}_k]} \quad (5)$$

where  $\boldsymbol{\tau} = \mathbf{K}[:, \mathbf{i}_k]$  and  $\mathbf{i}_k$  denotes the latest element in the vector  $\mathbf{i}$ . The algorithm is presented below in Algorithm 2 and uses the notation  $\mathbf{K}^{\cdot 2}$  to denote component wise squaring. Also, division of vectors are assumed to be component wise.

---

**Algorithm 2:** Matching pursuit for kernel principal components analysis (i.e., sparse KPCA)

---

**Input:** kernel  $\mathbf{K}$ , sparsity parameter  $k > 0$ .

1: initialize  $\mathbf{i} = ()$

2: **for**  $i = 1$  to  $k$  **do**

3: set  $\mathbf{i}_i$  to index of  $\max \frac{(\mathbf{K}^{\cdot 2})' \mathbf{1}}{\text{diag}\{\mathbf{K}\}}$

4: set  $\boldsymbol{\tau} = \mathbf{K}[:, \mathbf{i}_i]$  to deflate kernel matrix like so:

$$\mathbf{K} = \mathbf{K} - \frac{\boldsymbol{\tau}\boldsymbol{\tau}'}{\mathbf{K}[\mathbf{i}_i, \mathbf{i}_i]}$$

5: **end for**

6: compute  $\hat{\mathbf{K}}$  using  $\mathbf{i}$  and Equation (8)

**Output:** index vector  $\mathbf{i}$  (i.e., compression set indices)

This algorithm is equivalent to the algorithm proposed by [5] without sub-sampling (See Algorithm 6 in the Appendix). However, their motivation comes from the stance of finding a low rank matrix approximation of the kernel matrix. They proceed by looking for an approximation  $\hat{\mathbf{K}} = \mathbf{K}[:, \mathbf{i}]\mathbf{T}$  for a set  $\mathbf{i}$  and matrix  $\mathbf{T}$  such that the Frobenius norm between the trace residuals  $\text{tr}\{\mathbf{K} - \mathbf{K}[:, \mathbf{i}]\mathbf{T}\} = \text{tr}\{\mathbf{K} - \hat{\mathbf{K}}\}$  is minimal. They use a matching pursuit approach to find the set of indices  $\mathbf{i}$  and the projection matrix  $\mathbf{T}$ . However, the use of  $\mathbf{T}$  in computing the low rank matrix approximation seems to imply the need for additional information from outside of the chosen basis vectors in order to construct this approximation. We show that a projection into the space defined solely by the chosen indices is enough to reconstruct the kernel matrix and does not require any extra information.<sup>5</sup> The projection is the well known Nyström method [36] The following result proves that this projection (Algorithm 2) is in actual fact a *compression scheme* [10], [11].

*Claim 1:* The sparse kernel principal components analysis algorithm is a compression scheme.

*Proof:* An orthogonal projection  $P_{\mathbf{i}}(\phi(\mathbf{x}_j))$  of a feature vector  $\phi(\mathbf{x}_j)$  into a subspace defined only by the set of indices  $\mathbf{i}$  can be expressed as

$$P_{\mathbf{i}}(\mathbf{x}_j) = \phi(\mathbf{x}_j)' \tilde{\mathbf{X}} (\tilde{\mathbf{X}}' \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}' \quad (6)$$

<sup>5</sup>In their book, Smola and Schölkopf redefine their kernel approximation in the same way as we have done [26], however, they do not make the connection that it is a compression scheme (see Claim 1).

where  $\tilde{\mathbf{X}} = \mathbf{X}[:, \mathbf{i}]$  are the  $\mathbf{i}$  training examples from data matrix  $\mathbf{X}$ . It follows that:

$$\begin{aligned} P_i(\mathbf{x}_j)P_i(\mathbf{x}_j)' &= \phi(\mathbf{x}_j)'\tilde{\mathbf{X}}(\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{X}}(\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\phi(\mathbf{x}_j) \\ &= \mathbf{K}[j, \mathbf{i}]\mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1}\mathbf{K}[\mathbf{i}, j] \end{aligned} \quad (7)$$

with  $\mathbf{K}[j, \mathbf{i}]$  denoting the kernel entries between the index set  $\mathbf{i}$  and the feature vector  $\phi(\mathbf{x}_j)$ . Giving us the following projection into the space defined by  $\tilde{\mathbf{K}}$

$$\tilde{\mathbf{K}} = \mathbf{K}[:, \mathbf{i}]\mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1}\mathbf{K}[:, \mathbf{i}]. \quad (8)$$

Given a data point  $\phi(\mathbf{x}_j)$  in the feature space and a set of chosen indices  $\mathbf{i}$  we can reconstruct the projection using (7), i.e.,  $\mathbf{K}[j, \mathbf{i}]\mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1}\mathbf{K}[j, \mathbf{i}]'$ . Therefore, we only require kernel evaluations between the training examples indexed by  $\mathbf{i}$  and the data point  $\phi(\mathbf{x}_j)$  in order to make this reconstruction. Hence,  $\mathbf{i}$  forms a compression set. ■

Now that we have proved that SKPCA is a compression scheme we give the final result of this section that proves the equivalence of our derivation of SKPCA given in Algorithm 2 and the low rank matrix approximation algorithm (Algorithm 6) given in Appendix A. Note that the algorithm given in Appendix A was first derived in [5], however they further carry out a sub-sampling of 59 examples, which we do not include in Algorithm 6. The following result would also be true if both algorithms 2 and 6 sub-sampled the same examples.

*Theorem 5:* Algorithm 2 is equivalent to Algorithm 6.

*Proof:* Let  $\mathbf{K}$  be the kernel matrix and let  $\mathbf{K}[:, i]$  be the  $i$ th column of the kernel matrix. Assume  $\mathbf{X}$  is the input matrix containing rows of vectors that have already been mapped into a higher dimensional feature space using  $\phi$  such that  $\mathbf{X} = (\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_m))'$ . Smola and Schölkopf [5] state in Section IV-B of their paper that their algorithm 2 finds a low rank approximation of the kernel matrix such that it minimizes the Frobenius norm  $\|\mathbf{X} - \tilde{\mathbf{X}}\|_{\text{Frob}}^2 = \text{tr}\{\mathbf{K} - \tilde{\mathbf{K}}\}$  where  $\tilde{\mathbf{X}}$  is the low rank approximation of  $\mathbf{X}$ . Therefore, we need to prove that Algorithm 6 also minimizes this norm.

We would like to show that the maximum reduction in the Frobenius norm between the kernel  $\mathbf{K}$  and its projection  $\tilde{\mathbf{K}}$  is given by the choice of basis vectors that maximize the Rayleigh quotient together with deflation according to (5). At each stage we deflate by

$$\mathbf{K} = \mathbf{K} - \frac{\boldsymbol{\tau}\boldsymbol{\tau}'}{\mathbf{K}[\mathbf{i}_k, \mathbf{i}_k]}.$$

The trace  $\text{tr}\{\mathbf{K}\} = \sum_{i=1}^m \mathbf{K}[i, i]$  is the sum of the diagonal elements of matrix  $\mathbf{K}$ . Therefore

$$\begin{aligned} \text{tr}\{\mathbf{K}\} &= \text{tr}\{\mathbf{K}\} - \frac{\text{tr}\{\boldsymbol{\tau}\boldsymbol{\tau}'\}}{\mathbf{K}[\mathbf{i}_k, \mathbf{i}_k]} \\ &= \text{tr}\{\mathbf{K}\} - \frac{\text{tr}\{\boldsymbol{\tau}'\boldsymbol{\tau}\}}{\mathbf{K}[\mathbf{i}_k, \mathbf{i}_k]} \\ &= \text{tr}\{\mathbf{K}\} - \frac{\mathbf{K}^2[\mathbf{i}_k, \mathbf{i}_k]}{\mathbf{K}[\mathbf{i}_k, \mathbf{i}_k]}. \end{aligned}$$

The last term of the final equation corresponds exactly to the Rayleigh quotient of (4). Therefore the maximization of the

Rayleigh quotient does indeed correspond to the maximum reduction in the Frobenius norm between the approximated matrix  $\tilde{\mathbf{X}}$  and  $\mathbf{X}$ . ■

Now we can present the compression theory bound for a subspace method (i.e., SKPCA).

*Theorem 6:* Let  $\mathcal{A}_k$  be any learning algorithm having a reconstruction function that maps compression sets to *subspaces*. Let  $m$  be the size of the training set  $S$ ,  $k$  the size of the compression set  $S_1 \subset S$ , and let  $\hat{\mathcal{E}}_{m-k}[\ell(\mathcal{A}_k(S))]$  be the empirical residual loss between the  $m - k$  points outside of the compression set and their projections into a subspace, then for all choices of the indices of  $S_1$  and choices of empirical errors with probability  $1 - \delta$  over the generation of the remaining data, the expected loss  $\mathcal{E}[\ell(\mathcal{A}_k(S))]$  of algorithm  $\mathcal{A}_k$  given any training set  $S$  can be bounded by

$$\begin{aligned} \mathcal{E}[\ell(\mathcal{A}_k(S))] &\leq \min_{1 \leq t \leq k} \left[ \hat{\mathcal{E}}_{m-t}[\ell(\mathcal{A}_t(S))] + \right. \\ &\quad \left. \sqrt{\frac{R}{2(m-t)} \left[ t \ln\left(\frac{em}{t}\right) + \ln\left(\frac{2m}{\delta}\right) \right]} \right] \end{aligned}$$

where  $\ell(\cdot) \geq 0$  and  $R = \sup \ell(\cdot)$ .

*Proof:* Consider the case where we have a compression set of size  $k$ . Then we have  $\binom{m}{k}$  different ways of choosing the compression set. Given  $\delta$  confidence we apply Hoeffding's bound to the  $m - k$  points not in the compression set once for each choice by setting the RHS of Theorem 3 equal to  $\frac{\delta}{\binom{m}{k}}$ . From the definitions  $\ell(\cdot) \geq 0$  and  $R = \sup \ell(\cdot)$  we can set  $b_i = 0$  and  $a_i = \frac{R}{m-k}$ , respectively, in Hoeffding's bound. Using these facts we get

$$\begin{aligned} 2 \exp\left(-\frac{2\mathcal{E}^2}{\sum_{i=1}^{m-k} \left(\frac{R}{m-k} - 0\right)^2}\right) &= \frac{\delta}{\binom{m}{k}} \\ 2 \exp\left(-\frac{2\mathcal{E}^2}{(m-k)R^2/(m-k)^2}\right) &= \frac{\delta}{\binom{m}{k}} \\ 2 \exp\left(-\frac{(m-k)2\mathcal{E}^2}{R^2}\right) &= \frac{\delta}{\binom{m}{k}}. \end{aligned}$$

Solving for  $\mathcal{E}$  and further applying a factor  $1/m$  to  $\delta$  to ensure one application for each possible choice of  $k$  we get

$$\mathcal{E} \leq \sqrt{\frac{R^2}{2(m-k)} \left[ k \ln\left(\frac{em}{k}\right) + \ln\left(\frac{2m}{\delta}\right) \right]}.$$

Hence by Hoeffding's bound  $\Pr\{\mathcal{E}[\ell(\mathcal{A}_k(S))] > \hat{\mathcal{E}}_{m-k}[\ell(\mathcal{A}_k(S))] + \mathcal{E}\} \leq \delta$ . This together with the fact that using more dimensions can only reduce the expected loss on test points gives the result. ■

We now consider the application of the above bound to sparse KPCA. Let the corresponding loss function (residual) be defined as

$$\ell(\mathcal{A}_k(S))(\mathbf{x}) = \|\mathbf{x} - P_{\mathbf{i}_{1..k}}(\mathbf{x})\|^2$$

where  $\mathbf{x}$  is a test point and  $P_{\mathbf{i}_{1\dots k}}(\mathbf{x})$  its projection into the subspace determined by the set  $\mathbf{i}_1, \dots, \mathbf{i}_k$  of indices returned by  $\mathcal{A}_k(S)$ . From (6) it is easy to see that this projection is

$$P_{\mathbf{i}_{1\dots k}}(\mathbf{x}) = \mathbf{x}'\mathbf{X}[:, \mathbf{i}_{1\dots k}]\mathbf{K}[\mathbf{i}_{1\dots k}, \mathbf{i}_{1\dots k}]^{-1}\mathbf{X}[:, \mathbf{i}_{1\dots k}]'$$

Using these definitions, we can give a more specific loss bound in the case where we use a Gaussian kernel in the sparse kernel principal components analysis.

*Corollary 1 (Sample Compression Bound for Sparse KPCA):* Using a Gaussian kernel and all of the definitions from Theorem 6, we get the following bound:

$$\mathcal{E}[\ell(\mathcal{A}_k(S))] \leq \min_{1 \leq t \leq k} \left[ \frac{1}{m-t} \sum_{i=1}^{m-t} \|\mathbf{x}_i - P_{\mathbf{i}_{1\dots t}}(\mathbf{x}_i)\|^2 + \sqrt{\frac{1}{2(m-t)} \left[ t \ln\left(\frac{em}{t}\right) + \ln\left(\frac{2m}{\delta}\right) \right]} \right] \quad (9)$$

Note that  $R$  corresponds to the smallest radius of a ball that encloses all of the training points, and hence, in the case of the Gaussian kernel  $R$  equals 1.

We compare the sample compression bound proposed above with the following two Rademacher complexity bounds. First we define  $P_U(\mathbf{x})$  to be the orthogonal projection of  $\mathbf{x}$  into the subspace  $U_k$  defined by the  $k$  largest principal components found using PCA. We define the residual as  $P_{U_k}^\perp(\mathbf{x}) = \mathbf{x} - P_{U_k}(\mathbf{x})$ . The following global Rademacher complexity kernel principal components analysis (KPCA) bound was proposed by [14].

*Theorem 7 (Global Rademacher Complexity KPCA Bound):* If we perform PCA in the feature space defined by a kernel  $\kappa(\mathbf{x}, \mathbf{z})$  then with probability greater than  $1 - \delta$ , for any  $1 \leq k \leq m$ , we can upper bound the expected squared residual loss  $\ell(\mathcal{A}_k(S))(\mathbf{x}) = \|P_{U_k}^\perp(\mathbf{x})\|^2$  by

$$\mathcal{E}[\ell(\mathcal{A}_k(S))] \leq \min_{1 \leq t \leq k} \left[ \frac{1}{m} \lambda^{>t}(S) + \frac{1 + \sqrt{t}}{\sqrt{m}} \sqrt{\frac{2}{m} \sum_{i=1}^m \kappa(\mathbf{x}_i, \mathbf{x}_i)^2} \right] + R^2 \sqrt{\frac{18 \ln(2m/\delta)}{m}}$$

where the support of the distribution is in a ball of radius  $R$  in the feature space and  $\lambda^{>t}(S) = \sum_{i=t+1}^m \lambda_i$  is the sum of the eigenvalues greater than  $t$  computed from the training data in the feature space.

In the plots we call this bound the ‘‘PCA bound.’’

The second bound we test against is a refinement of the global approach and uses local Rademacher complexity, and was proposed by [15].

*Theorem 8 (Local Rademacher Complexity KPCA Bound):* If we perform PCA in the feature space defined by a kernel  $\kappa(\mathbf{x}, \mathbf{z})$  then with probability greater than  $1 - \delta$ , for any  $1 \leq$

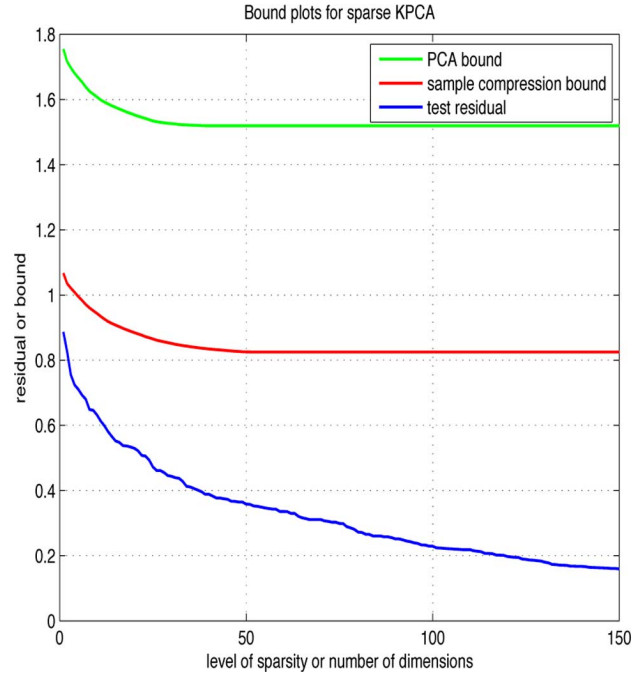


Fig. 2. Bound plots for sparse KPCA comparing the sample compression bound proposed in this paper and the PCA bound of [14]. We plot the residual loss and bound values against the level of sparsity (sample compression bound) and the number of dimensions used (PCA bound). We used the Boston housing data set, available from the UCI repository [37].

$k \leq m$ , we can upper bound the expected squared residual loss  $\ell(\mathcal{A}_k(S))(\mathbf{x}) = \|P_{U_k}^\perp(\mathbf{x})\|^2$  by

$$\mathcal{E}[\ell(\mathcal{A}_k(S))] \leq c \left( \sqrt{\sum_{i>k} \lambda_i(\mathbf{K}) \left( \rho(R, k, m) + \frac{R \ln(1/\delta)}{m} \right)} + \rho(k, m) + \frac{R \ln(1/\delta)}{m} \right)$$

where

$$\rho(R, k, m) = \inf_{t \geq 0} \left\{ R \frac{t}{m} + \sqrt{\frac{k}{m} \sum_{j>t} \lambda_j(\bar{\mathbf{K}})} \right\}$$

and where the support of the distribution is in a ball of radius  $R$  in the feature space,  $c$  is a constant ( $c \leq 80$ ),  $\lambda_j(\mathbf{K})$  denotes the  $j$ th largest eigenvalue of kernel matrix  $\mathbf{K}$  and  $\bar{\mathbf{K}} = (\mathbf{I} - \mathbf{1}\mathbf{1}')\mathbf{K}'\mathbf{K}(\mathbf{I} - \mathbf{1}\mathbf{1}')$  is a centered kernel matrix. We call this the ‘‘local PCA bound.’’

We conducted the experiments using the Gaussian kernel with a width parameter  $\sigma = 0.035$ . We also present the results in several figures. Due to the loose nature of the local Rademacher bound, we plot of all these bounds in Figs. 4 and 5, using log plots. We start the experiments with the global Rademacher bounds.

Fig. 2 plots the test error residuals (for the Boston housing data set) together with its upper bounds computed using Theorem 7 (PCA bound) and the sample compression bound of Corollary 1. The sample compression bound is much tighter than the PCA bound and also nontrivial (unlike the PCA bound).

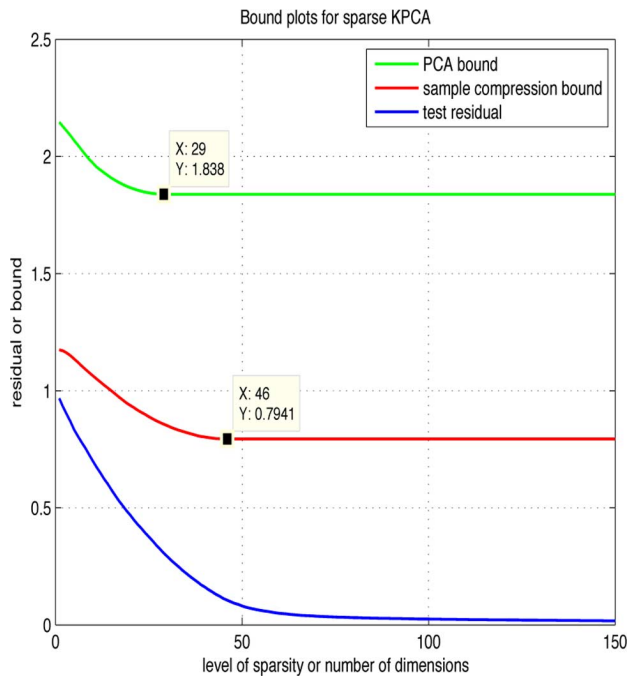


Fig. 3. Plot of the residual loss and bound values against the level of sparsity (sample compression bound) and the number of dimensions used (PCA bound). A Toy experiment with 1000 training examples (and 450 dimensions) drawn randomly from a Gaussian distribution with zero mean and unit variance.

The sample compression bound is at its lowest point after 43 basis vectors have been added. We speculate that at this point the “true” dimensions of the data have been found and that all other dimensions correspond to “noise”. This corresponds to the point at which the plot of residual errors becomes linear, suggesting dimensions with uniform noise. We carry out an extra toy experiment to help assess whether or not this is true and to show that the sample compression bound can help indicate when the principal components have captured most of the actual data. The plot of Fig. 3 depicts the results of a toy experiment where we randomly sampled 1000 examples with 450 dimensions from a Gaussian distribution with zero mean and unit variance. We then ensured that 50 dimensions contained considerably larger eigenvalues than the remaining 400. From Fig. 3 we see that the test residual keeps dropping at a constant rate after 50 basis vectors have been added. The compression bound picks 46 dimensions with the largest eigenvalues, however, the KPCA bound of [14] is much more optimistic and is at its lowest point after 29 basis vectors, suggesting erroneously that SKPCA has captured most of the data in 29 dimensions. Therefore, as well as being tighter and nontrivial (i.e., less than 1), the compression bound appears better at predicting the best choice for the number of dimensions to use with sparse KPCA. Note that we carried out this experiment without randomly permuting the projections into a subspace because SKPCA is rotation invariant and will always choose the principal components with the largest eigenvalues.

In Figs. 2 and 3 we gave bound plots for the global Rademacher complexity bound of Theorem 7 and the sample compression bound of Corollary 1 only. Now using the same setup as Figs. 2 and 3, we add the plot of Theorem 8 (local Rademacher bound) to Figs. 4 and 5, respectively. We call

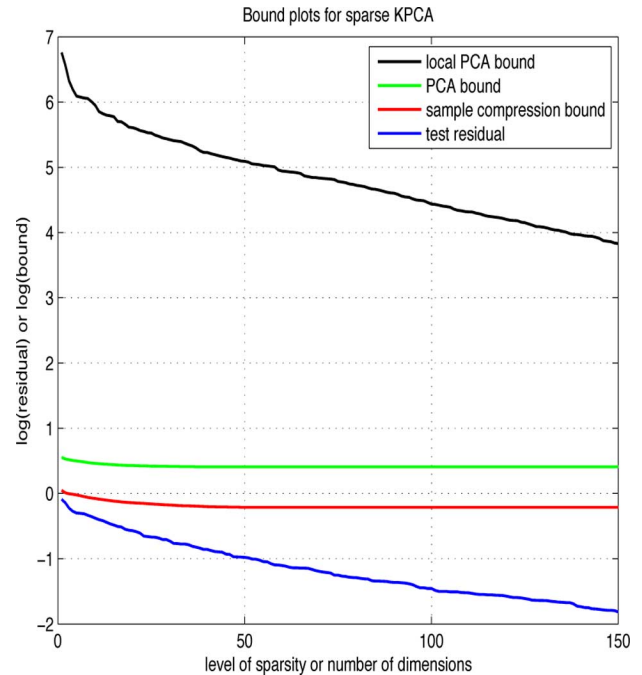


Fig. 4. Log plot of Fig. 2, including the local Rademacher complexity bound of Theorem 8 using  $c = 80$ . The logarithm of the  $y$  axis is given in order to get all curves on the same graph.

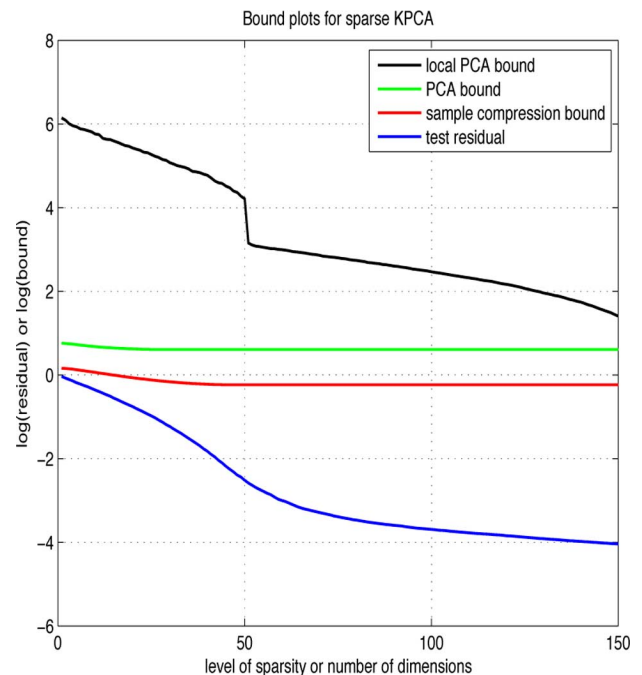


Fig. 5. Log plot of Fig. 3, including the local Rademacher complexity bound of Theorem 8 using  $c = 80$ . The logarithm of the  $y$  axis is given in order to get all curves on the same graph.

this the “local PCA bound,” and due to the large constant of ( $c \leq 80$ ), use a log plot of the  $y$  axis.

As you can see from Fig. 4 the local PCA bound seems to always be monotonically decreasing. It is at its lowest point after 150 dimensions have been chosen. This bound would fail to predict a sparse subspace of sparse kernel PCA as we saw to be the case with the sample compression bound and the (global

Rademacher) PCA bound—however, it is clear that as  $m \rightarrow \infty$  then this bound will be smaller than the PCA and sample compression bound. Fig. 4 corresponds to the toy experiment we conducted with the actual data living in a 50 dimensional space (see Fig. 3)—interestingly the local PCA bound has a large drop in its value after 50 basis vectors. However, it still seems to decrease after this point, and if the minimum was chosen using the local PCA bound to predict the size of the subspace (containing most of the variance of the data) then this would happen much later, at 150 dimensions. Clearly, for finite sample sizes the global Rademacher and sample compression bounds are preferred, and for sparse kernel PCA, the sample compression bound proposed should be preferred.

#### IV. EXTENSIONS: SPARSE KERNEL CANONICAL CORRELATION ANALYSIS

Another form of dimensionality reduction is canonical correlation analysis (CCA)—a natural extension of PCA when two views of the same object are present. Both algorithms maximize a form of the Rayleigh quotient. Therefore we can propose a natural extension of SKPCA (from the last section) to sparse kernel canonical correlation analysis (SKCCA).

##### A. Algorithm

Assume we are given two views  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^m$  and  $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^m$  of the same data where  $\mathbf{x}_i$  and  $\mathbf{y}_i$  are vectors of length  $r \in \mathbb{Z}^+$  and  $s \in \mathbb{Z}^+$  respectively. We then compute projections  $P_x : \mathbf{x} \mapsto \mathbf{x}'\mathbf{w}_x$  and  $P_y : \mathbf{y} \mapsto \mathbf{y}'\mathbf{w}_y$ . The idea of canonical correlation analysis (CCA) is to maximize the correlation  $\text{corr}(P_x(\mathbf{X}), P_y(\mathbf{Y}))$  between the data in their corresponding projection space. Taking the maximum correlation of these two projections reduces to the following maximization problem:

$$\begin{aligned} \lambda_{x,y} &= \max_{\mathbf{w}_x, \mathbf{w}_y} \frac{\mathbf{w}_x' \mathbf{X} \mathbf{Y}' \mathbf{w}_y}{\sqrt{\mathbf{w}_x' \mathbf{X} \mathbf{X}' \mathbf{w}_x \mathbf{w}_y' \mathbf{Y} \mathbf{Y}' \mathbf{w}_y}} \\ &= \max_{\mathbf{w}_x, \mathbf{w}_y} \frac{\mathbf{w}_x' \mathbf{C}_{xy} \mathbf{w}_y}{\sqrt{\mathbf{w}_x' \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y' \mathbf{C}_{yy} \mathbf{w}_y}} \end{aligned} \quad (10)$$

where  $\mathbf{C}_{xy} = \mathbf{C}'_{yx}$  is the covariance matrix between  $\mathbf{X}$  and  $\mathbf{Y}$ ,  $\mathbf{C}_{xx}$  and  $\mathbf{C}_{yy}$  the covariance matrices of  $\mathbf{X}$  and  $\mathbf{Y}$ , respectively, and  $\lambda_{x,y}$  the eigenvalues corresponding to  $\mathbf{w}_x$  and  $\mathbf{w}_y$ .

We would like to construct a sparse kernel canonical correlation analysis (SKCCA) algorithm. By sparsity we mean using a small subset of basis vectors from the sample  $(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, m$ . The sparse index set of basis vectors will be contained in an index vector  $\mathbf{i}$ . Following earlier notation for sparse PCA the sparse CCA problem can be defined by using weight vectors  $\mathbf{w}_x \in \text{span}\{\mathbf{X}[:, \mathbf{i}]\}$  and  $\mathbf{w}_y \in \text{span}\{\mathbf{Y}[:, \mathbf{i}]\}$ . We can convert this primal problem into its dual by rewriting the weight vectors in terms of a linear combination of the training examples and the dual weight vectors

$$\mathbf{w}_x = \mathbf{X}[:, \mathbf{i}] \tilde{\alpha}_x \quad (11)$$

$$\mathbf{w}_y = \mathbf{Y}[:, \mathbf{i}] \tilde{\alpha}_y. \quad (12)$$

Substituting these two expressions into the CCA problem of (10) we get

$$\max_{\mathbf{i}, \tilde{\alpha}_x, \tilde{\alpha}_y} \frac{\tilde{\alpha}_x' \mathbf{X}[:, \mathbf{i}]' \mathbf{X} \mathbf{Y}' \mathbf{Y}[:, \mathbf{i}] \tilde{\alpha}_y}{\sqrt{\tilde{\alpha}_x' \mathbf{X}[:, \mathbf{i}]' \mathbf{X} \mathbf{X}' \mathbf{X}[:, \mathbf{i}] \tilde{\alpha}_x \tilde{\alpha}_y' \mathbf{Y}[:, \mathbf{i}]' \mathbf{Y} \mathbf{Y}' \mathbf{Y}[:, \mathbf{i}] \tilde{\alpha}_y}}.$$

Furthermore, we have  $\mathbf{K}_x[:, \mathbf{i}]' = \mathbf{X}[:, \mathbf{i}]' \mathbf{X}$  and  $\mathbf{K}_y[:, \mathbf{i}]' = \mathbf{Y}[:, \mathbf{i}]' \mathbf{Y}$ , and sparse kernel CCA as

$$\tilde{\lambda}_{x,y} = \max_{\mathbf{i}, \tilde{\alpha}_x, \tilde{\alpha}_y} \frac{\tilde{\alpha}_x' \mathbf{K}_x[:, \mathbf{i}]' \mathbf{K}_y[:, \mathbf{i}] \tilde{\alpha}_y}{\sqrt{\tilde{\alpha}_x' \mathbf{K}_x^2[:, \mathbf{i}] \tilde{\alpha}_x \tilde{\alpha}_y' \mathbf{K}_y^2[:, \mathbf{i}] \tilde{\alpha}_y}}$$

where  $\tilde{\alpha}_x$  and  $\tilde{\alpha}_y$  are sparse dual eigenvectors. This leads to, for fixed  $\mathbf{i}$ , the sparse KCCA generalized eigenproblem<sup>6</sup> of the form

$$\begin{bmatrix} \mathbf{0} & \mathbf{K}_{xy}[\mathbf{i}, \mathbf{i}] \\ \mathbf{K}_{yx}[\mathbf{i}, \mathbf{i}] & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\alpha}_x \\ \tilde{\alpha}_y \end{bmatrix} = \tilde{\lambda}_{x,y} \begin{bmatrix} \mathbf{K}_x^2[\mathbf{i}, \mathbf{i}] & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_y^2[\mathbf{i}, \mathbf{i}] \end{bmatrix} \begin{bmatrix} \tilde{\alpha}_x \\ \tilde{\alpha}_y \end{bmatrix} \quad (13)$$

where  $\mathbf{K}_{xy}[\mathbf{i}, \mathbf{i}] = \mathbf{K}_x[:, \mathbf{i}]' \mathbf{K}_y[:, \mathbf{i}]$  and  $\mathbf{K}_{yx}[\mathbf{i}, \mathbf{i}] = \mathbf{K}_{xy}[\mathbf{i}, \mathbf{i}]'$ .

Using a similar strategy to [5] we now show that maximizing the quotient of the CCA problem leads to a fast method for choosing basis vectors. Also after picking a basis vector we must project into a space orthogonal to it, and describe a deflation step that guarantees future basis vectors chosen are orthogonal to all others.

We take a similar approach to Algorithm 6 which implies the maximization of the generalized Rayleigh quotient

$$\max_i \rho_i = \frac{\mathbf{e}_i' \mathbf{K}_x' \mathbf{K}_y \mathbf{e}_i}{\sqrt{\mathbf{e}_i' \mathbf{K}_x^2 \mathbf{e}_i \mathbf{e}_i' \mathbf{K}_y^2 \mathbf{e}_i}} \quad (14)$$

where  $\mathbf{e}_i$  is the  $i$ th unit vector. At each iteration we look to find the basis vector that maximizes the quotient given by (14). We can rewrite this equation in terms of each individual kernel basis vector by observing that  $\mathbf{K}_x[:, i] = \mathbf{K}_x \mathbf{e}_i$  and  $\mathbf{K}_y[:, i] = \mathbf{K}_y \mathbf{e}_i$ , which gives

$$\begin{aligned} \max_i \rho_i &= \frac{\mathbf{K}_x[:, i]' \mathbf{K}_y[:, i]}{\sqrt{(\mathbf{K}_x[:, i]' \mathbf{K}_x[:, i]) (\mathbf{K}_y[:, i]' \mathbf{K}_y[:, i])}} \\ &= \frac{\mathbf{K}_x[:, i]' \mathbf{K}_y[:, i]}{\sqrt{\mathbf{K}_x^2[i, i] \mathbf{K}_y^2[i, i]}}. \end{aligned}$$

Once the index  $i$  has been chosen such that it maximizes the above equation then the following orthogonality procedure (deflation) is carried out to make sure future chosen bases are sufficiently far (geometrically) from those already added to the set  $\mathbf{i}$ .

Initially, at the first step  $j = 1$ , let  $\hat{\mathbf{K}}_x^j = \mathbf{K}_x$  and  $\hat{\mathbf{K}}_y^j = \mathbf{K}_y$ , and let  $\hat{\mathbf{K}}_x^j, \hat{\mathbf{K}}_y^j$  denote the deflated kernel matrices at the  $j$ th iteration. To find the deflated matrices at step  $j + 1$  we use the following single sided deflation defined for KMP [6]

$$\begin{aligned} \hat{\mathbf{K}}_x^{j+1} &= \left( \mathbf{I} - \frac{\boldsymbol{\tau}_x \boldsymbol{\tau}_x'}{\boldsymbol{\tau}_x' \boldsymbol{\tau}_x} \right) \hat{\mathbf{K}}_x^j = \hat{\mathbf{K}}_x^j - \frac{\boldsymbol{\tau}_x \left( \boldsymbol{\tau}_x' \hat{\mathbf{K}}_x^j \right)}{\boldsymbol{\tau}_x' \boldsymbol{\tau}_x} \\ \hat{\mathbf{K}}_y^{j+1} &= \left( \mathbf{I} - \frac{\boldsymbol{\tau}_y \boldsymbol{\tau}_y'}{\boldsymbol{\tau}_y' \boldsymbol{\tau}_y} \right) \hat{\mathbf{K}}_y^j = \hat{\mathbf{K}}_y^j - \frac{\boldsymbol{\tau}_y \left( \boldsymbol{\tau}_y' \hat{\mathbf{K}}_y^j \right)}{\boldsymbol{\tau}_y' \boldsymbol{\tau}_y} \end{aligned}$$

where  $\boldsymbol{\tau}_x = \hat{\mathbf{K}}_x^j[:, \mathbf{i}_j]$  and  $\boldsymbol{\tau}_y = \hat{\mathbf{K}}_y^j[:, \mathbf{i}_j]$  such that  $j = |\mathbf{i}|$ ,  $j > 0$  and  $\mathbf{i}_j$  is the latest element added to vector  $\mathbf{i}$  and  $\mathbf{I}$  is the identity matrix. This procedure is repeated until  $k$  basis vectors have been chosen. The deflation at each stage can be computed

<sup>6</sup>Given matrices  $\mathbf{A}, \mathbf{B}$  we have the following generalized eigenproblem of the form  $\mathbf{A}\boldsymbol{\alpha} = \lambda\mathbf{B}\boldsymbol{\alpha}$  where  $(\lambda, \boldsymbol{\alpha})$  are the eigenvalue-eigenvector pair, respectively.

in  $\mathcal{O}(m^2)$  time where  $m$  are the number of examples. This is because the bracketed computation in the expression  $\tau_x(\tau'_x \hat{\mathbf{K}}_x^j)$  can be computed first—hence saving us from a cubic computation. This protocol is described in Algorithm 3. Notice the similarities between this algorithm and Algorithm 2 which uses the same procedure of *quotient maximization* and *deflation* in order to evaluate a sparse kernel principal components analysis.

---

**Algorithm 3:** Matching pursuit for kernel canonical correlation analysis (i.e., sparse KCCA)

---

**Input:** two views  $\mathbf{K}_x, \mathbf{K}_y$  and sparsity parameter  $k > 0$ .

1: initialize index vector  $\mathbf{i} = ()$  and an all one vector  $\mathbf{1}$  of size  $m \times 1$ .

2: **for**  $i = 1$  to  $k$  **do**

3: set  $\mathbf{i}_i$  to index of  $\max \frac{((\mathbf{K}_x \cdot \mathbf{K}_y)' \mathbf{1})}{\sqrt{((\mathbf{K}_x^2)' \mathbf{1}) \cdot ((\mathbf{K}_y^2)' \mathbf{1})}}$

4: set  $\tau_x = \mathbf{K}_x[:, \mathbf{i}_i]$  and  $\tau_y = \mathbf{K}_y[:, \mathbf{i}_i]$  to deflate kernel matrices like so:

$$\begin{aligned} \mathbf{K}_x &= \mathbf{K}_x - \frac{\tau_x(\tau'_x \mathbf{K}_x)}{\tau'_x \tau_x} \\ \mathbf{K}_y &= \mathbf{K}_y - \frac{\tau_y(\tau'_y \mathbf{K}_y)}{\tau'_y \tau_y} \end{aligned}$$

5: **end for**

6: project final  $\mathbf{i}$  into orthogonal subspace and carry out (K)CCA to find primal  $\tilde{\mathbf{w}}_x, \tilde{\mathbf{w}}_y$  or dual eigenvectors  $\tilde{\alpha}_x, \tilde{\alpha}_y$  and  $\lambda$ .

**Output:** index vector  $\mathbf{i}$  and eigenvectors  $\tilde{\mathbf{w}}_x, \tilde{\mathbf{w}}_y$  (or  $\tilde{\alpha}_x, \tilde{\alpha}_y$ )

After finding the set of indices  $\mathbf{i}$  we can use the training examples indexed by  $\mathbf{i}$  to make an orthogonal projection within which (K)CCA can be computed. We will compute the projection using the primal feature representation as opposed to the kernel (dual) representation because it will create small  $k \times k$  matrices—running CCA on these matrices will be significantly more efficient than using the  $m \times m$  matrices defined for the full kernel projections. Recall that an orthogonal projection  $P_{\mathbf{i}}$  for any (kernel) basis  $\mathbf{K}[:, \mathbf{i}] = \mathbf{K}[:, i_1], \dots, \mathbf{K}[:, i_k]$  can be defined as  $\mathbf{K}[:, \mathbf{i}] \mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1} \mathbf{K}[:, \mathbf{i}]'$ . Furthermore, if we would like to make a (column) projection of matrix  $\mathbf{K}$  onto this orthogonal projection we get

$$P_{\mathbf{i}}(\mathbf{K}) = \mathbf{K}' \mathbf{K}[:, \mathbf{i}] \mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1} \mathbf{K}[:, \mathbf{i}]'$$

Using the Cholesky decomposition on  $\mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1}$  we obtain a matrix  $\mathbf{R}$  such that  $\mathbf{R}' \mathbf{R} = \mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1}$ . Hence, we can redefine the projection above in terms of  $\mathbf{R}$  as  $\mathbf{K}[:, \mathbf{i}] \mathbf{R}' \mathbf{R} \mathbf{K}[:, \mathbf{i}]'$ . In CCA given that we have two (views) kernel matrices  $\mathbf{K}_x$  and  $\mathbf{K}_y$ , we need two individual projections

$$\begin{aligned} P_{\mathbf{i}}^x(\mathbf{K}_x) &= \mathbf{K}_x' \mathbf{K}_x[:, \mathbf{i}] \mathbf{K}_x[\mathbf{i}, \mathbf{i}]^{-1} \mathbf{K}_x[:, \mathbf{i}]' \\ &= \mathbf{K}_x' \mathbf{K}_x[:, \mathbf{i}] \mathbf{R}'_x \mathbf{R}_x \mathbf{K}_x[:, \mathbf{i}]', \\ P_{\mathbf{i}}^y(\mathbf{K}_y) &= \mathbf{K}_y' \mathbf{K}_y[:, \mathbf{i}] \mathbf{K}_y[\mathbf{i}, \mathbf{i}]^{-1} \mathbf{K}_y[:, \mathbf{i}]' \\ &= \mathbf{K}_y' \mathbf{K}_y[:, \mathbf{i}] \mathbf{R}'_y \mathbf{R}_y \mathbf{K}_y[:, \mathbf{i}]' \end{aligned}$$

where  $\mathbf{R}_x$  and  $\mathbf{R}_y$  have been defined using  $\mathbf{K}_x[\mathbf{i}, \mathbf{i}]^{-1}$  and  $\mathbf{K}_y[\mathbf{i}, \mathbf{i}]^{-1}$ . Let us create matrices  $\tilde{\mathbf{C}}'_x = \mathbf{K}_x[:, \mathbf{i}] \mathbf{R}'_x$  and  $\tilde{\mathbf{C}}'_y = \mathbf{K}_y[:, \mathbf{i}] \mathbf{R}'_y$  giving covariance matrices

$$\begin{aligned} \tilde{\mathbf{C}}_{xx} &= \tilde{\mathbf{C}}_x \tilde{\mathbf{C}}'_x \\ \tilde{\mathbf{C}}_{yy} &= \tilde{\mathbf{C}}_y \tilde{\mathbf{C}}'_y \\ \tilde{\mathbf{C}}_{xy} &= \tilde{\mathbf{C}}_x \tilde{\mathbf{C}}'_y \\ \tilde{\mathbf{C}}_{yx} &= \tilde{\mathbf{C}}'_y \tilde{\mathbf{C}}_x. \end{aligned}$$

These covariance matrices are used to solve CCA (using generalized eigenvalue problem, Gram-Schmidt [25], etc.) to generate the set of eigenvectors  $\tilde{\mathbf{w}}_x$  and  $\tilde{\mathbf{w}}_y$ . After finding these eigenvectors we can project new examples onto them in order to be in the common semantic space.

Let  $\mathbf{K}[j, \mathbf{i}]$  denote the kernel matrix between a new example in feature space  $\phi(\mathbf{x}_j)$  and the examples indexed by  $\mathbf{i}$ . As above we project  $\mathbf{K}[j, \mathbf{i}]$  into the space defined by  $\mathbf{R}$  so that  $\mathbf{k}_j = \mathbf{R} \mathbf{K}[j, \mathbf{i}]'$ . Similarly in (K)CCA with two views we would have

$$\begin{aligned} \mathbf{k}_j^x &= \mathbf{R}_x \mathbf{K}_x[j, \mathbf{i}]' \\ \mathbf{k}_j^y &= \mathbf{R}_y \mathbf{K}_y[j, \mathbf{i}]'. \end{aligned}$$

Finally, we can project  $\mathbf{k}_j^x$  and  $\mathbf{k}_j^y$  onto  $\tilde{\mathbf{w}}_x$  and  $\tilde{\mathbf{w}}_y$  by computing  $P_x(\mathbf{k}_j^x) = \tilde{\mathbf{w}}_x' \mathbf{k}_j^x$  and  $P_y(\mathbf{k}_j^y) = \tilde{\mathbf{w}}_y' \mathbf{k}_j^y$ . The difference  $\|P_x(\mathbf{k}_j^x) - P_y(\mathbf{k}_j^y)\|$  between these two projections is the reconstruction error of sparse KCCA for this particular paired example  $(\mathbf{x}_j, \mathbf{y}_j)$ .

Fig. 6 depicts the reconstruction error of SKCCA against KCCA when tested on an English-Spanish text corpora data set [38]. The KCCA implementation uses the Gram-Schmidt orthogonalization procedure as described in [18].

We can see from this figure that SKCCA is competitive with KCCA which uses several different regularization parameters. We conjecture that sparsity is indeed a more robust form of regularization when compared to the regularized version of KCCA. The figure presented is indicative of this conjecture and so we also give a zoomed in plot of the first 50 dimensions used for KCCA and the first 50 basis vectors chosen for SKCCA as the bottom plot of Fig. 6. The following bound also implies such good behavior, by upper bounding the reconstruction error that may be incurred by SKCCA on a single dimension.

## B. Generalization Error Analysis

The compression bound for SKPCA cannot be applied in the SKCCA setting because the basis vectors chosen to produce the common subspace require the entire information from the training set. However, from the type of analysis we made earlier for KMP we can upper bound the future loss of sparse KCCA.

To help keep the notation consistent with the bound introduced for KMP we make the following definitions. We denote the  $\mathcal{X}$  view sample as  $S^{\mathcal{X}}$  and similarly the  $\mathcal{Y}$  view sample as  $S^{\mathcal{Y}}$ . Therefore, two training samples consisting of paired data sets from the joint space  $\mathcal{X} \times \mathcal{Y}$  will be denoted as  $S^{\mathcal{X} \times \mathcal{Y}} = S^{\mathcal{X}} \cup S^{\mathcal{Y}}$ . We denote the index set of the chosen basis vectors as  $\mathbf{i}$  and  $S_{\mathbf{i}}^{\mathcal{X} \times \mathcal{Y}}$  as the paired samples indexed by vector  $\mathbf{i}$ . Earlier we denoted  $P_x$  to be the projection for the  $\mathcal{X}$  view onto the eigenvector but here use the notation  $f_x$  to denote the same projection function. We make the same change in the notation of the  $P_y$  function.

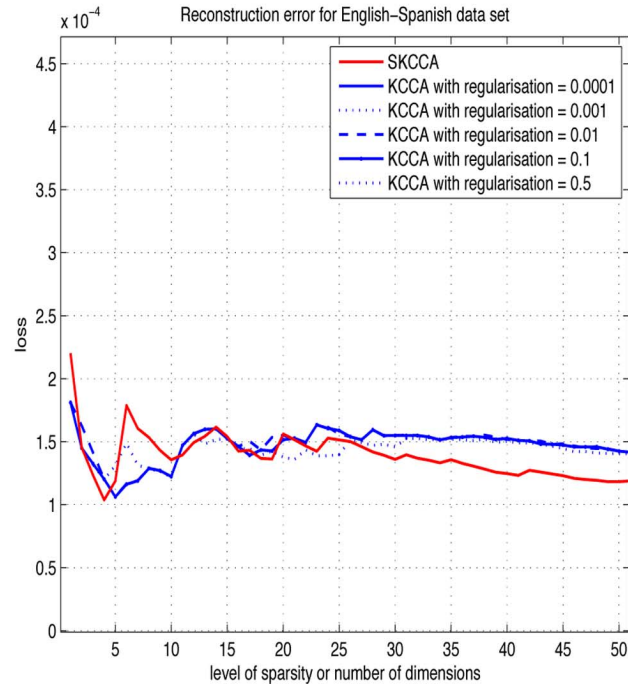
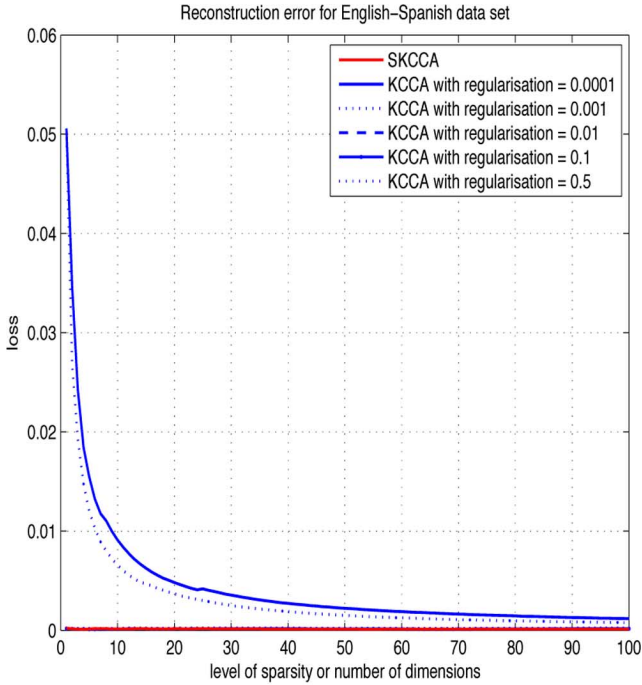


Fig. 6. Top: plot for all dimensions. Bottom: a closer look at the top plot for the first 50 dimensions or basis vectors.

We would also like to remind the reader that the primal weight vectors  $\mathbf{w}_x$  and  $\mathbf{w}_y$  for each view are 1-dimensional weight vectors for some given dimension  $i \in \{1, \dots, k\}$ . Finally, similarly to Definition 4, we define the following loss function for SKCCA.

*Definition 5:* Let  $S^{\mathcal{X} \times \mathcal{Y}} \sim \mathcal{D}$  be a paired training sample from a fixed but unknown distribution  $\mathcal{D}$ . Given the projection functions  $f_x = f_x(\mathbf{x}) = \mathbf{w}'_x \mathbf{x}$  and  $f_y = f_y(\mathbf{y}) = \mathbf{w}'_y \mathbf{y}$  and the error  $\ell(f_x, f_y) = |\mathbf{w}'_x \mathbf{x} - \mathbf{w}'_y \mathbf{y}| = |f_x - f_y|$  for the paired

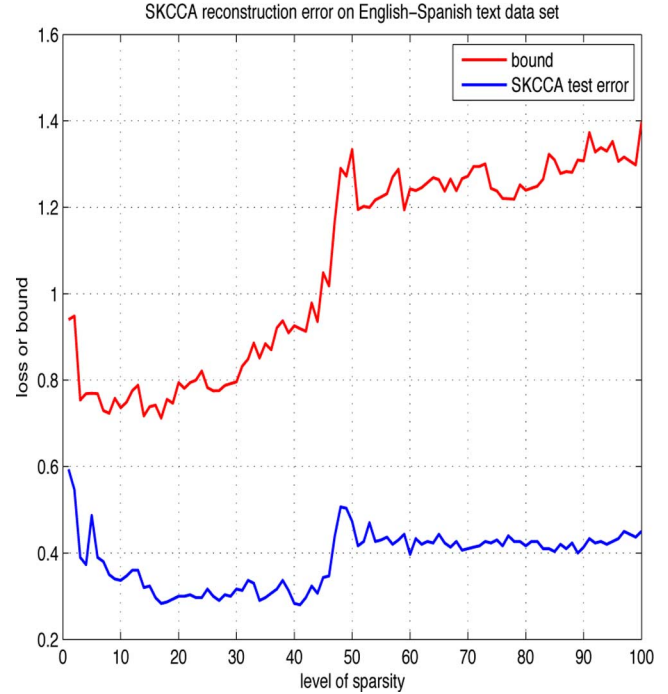


Fig. 7. Bound plot for sparse KCCA using 1-dimension with  $\alpha = 0.0095$  and scaled down by a factor of 5.

data points  $\mathbf{x}$  and  $\mathbf{y}$  we can define, for some fixed positive scalar  $\alpha \in \mathbb{R}$ , the corresponding true classification loss as

$$\ell_\alpha(f_x, f_y) = \Pr_{(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}} \{|f_x(\mathbf{x}) - f_y(\mathbf{y})| > \alpha\}.$$

Similarly, we can define the corresponding empirical classification loss as

$$\begin{aligned} \hat{\ell}_\alpha(f_x, f_y) &= \Pr_{(\mathbf{x}, \mathbf{y}) \sim S^{\mathcal{X} \times \mathcal{Y}}} \{|f_x(\mathbf{x}) - f_y(\mathbf{y})| > \alpha\} \\ &= \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim S^{\mathcal{X} \times \mathcal{Y}}} \{\mathbb{1}(|f_x(\mathbf{x}) - f_y(\mathbf{y})| > \alpha)\}, \end{aligned}$$

where  $\mathbb{1}$  is the indicator function and  $S^{\mathcal{X} \times \mathcal{Y}}$  is suppressed when clear from context.

Given this definition we would like to upper bound the true classification loss with the information gained from the empirical classification loss. We can proceed in much the same way as was done for KMP but with the difference that the weight vectors  $\mathbf{w}_x$  and  $\mathbf{w}_y$  are the vectors that allow a projection into a single dimension. Giving a bound on the loss of the two corresponding projections  $f_x$  and  $f_y$  onto each dimension. By making this style of analysis we can adapt the KMP bound as follows.

*Theorem 9:* Fix  $0 < \alpha \in \mathbb{R}$ . Let  $\mathcal{A}$  be the SKCCA algorithm such that  $\mathcal{A}(S^{\mathcal{X}}) = f_x$  and  $\mathcal{A}(S^{\mathcal{Y}}) = f_y$ , and let  $m$  be the size of the paired training set  $S^{\mathcal{X} \times \mathcal{Y}}$ . Without loss of generality let  $S^{\mathcal{X} \times \mathcal{Y}}$  be reordered so that the first  $k$  points form the compression set  $S_1^{\mathcal{X} \times \mathcal{Y}}$  and the next  $t$  points form the training error set  $S_2^{\mathcal{X} \times \mathcal{Y}}$  such that  $\mathbb{1}(|f_x(\mathbf{x}_i) - f_y(\mathbf{y}_i)| > \alpha) = 1$  for all  $i = k + 1, \dots, k + t$ . Then for all choices of the indices of  $S_1^{\mathcal{X} \times \mathcal{Y}}$  and  $S_2^{\mathcal{X} \times \mathcal{Y}}$ , with probability  $1 - \delta$  over the generation

of the remaining data  $\bar{S}^{\mathcal{X} \times \mathcal{Y}} = S^{\mathcal{X} \times \mathcal{Y}} \setminus (S_1^{\mathcal{X} \times \mathcal{Y}} \cup S_2^{\mathcal{X} \times \mathcal{Y}})$  we have the following upper bound on the expected loss:

$$\mathcal{E}[\ell_\alpha(\mathcal{A}(S^{\mathcal{X}}), \mathcal{A}(S^{\mathcal{Y}}))] \leq \frac{2}{m-k-t} \left[ (k+1) \log \left( \frac{4e(m-k-t)}{k+1} \right) + k \log \left( \frac{em}{k} \right) + t \log \left( \frac{e(m-k)}{t} \right) + \log \left( \frac{2m^2}{\delta} \right) \right].$$

*Proof:* We can treat the loss between  $f_x(\mathbf{x}) = \mathbf{w}'_x \mathbf{x}$  and  $f_y(\mathbf{y}) = \mathbf{w}'_y \mathbf{y}$  as a regression loss

$$(\mathbf{w}_x - \mathbf{w}_y) \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \mathbf{w}'_x \mathbf{x} - \mathbf{w}'_y \mathbf{y} \quad (15)$$

where we would want (15) to be 0 (or the *zero function*) for all  $\mathbf{x}$  and  $\mathbf{y}$  and every dimension projection  $\mathbf{w}_x$  and  $\mathbf{w}_y$ . This regression loss can be mapped into its corresponding classification loss using Definition 5. Therefore by constraining the weight vectors  $\mathbf{w}_x, \mathbf{w}_y$  to be 1—dimensional and using this loss function we can proceed in the same manner as the proof of Theorem 4. ■

The following plot is taken from the reconstruction error of a one-dimensional projection for SKCCA. We conducted the experiment using a Spanish-English text corpora database, where we had paired texts in English and Spanish, and used the TFIDF scores as features. We projected each paired English and Spanish text using the one-dimensional SKCCA projection direction and computed the empirical classification loss described in Definition 5 for some fixed  $\alpha$ . The bound has been scaled down by a factor of 5 to get both plots on the same figure.

## V. EXTENSIONS: OTHER LEARNING DOMAINS

The results of this paper point to a general algorithm for matching pursuit and also a general VC bound upper bounding the loss of matching pursuit. This framework equates to maximizing some loss function and then orthogonalizing the remaining basis functions, in order to construct a small subspace in which to carry out learning. Therefore, matching pursuit can be viewed as a meta-scheme for learning algorithms.

It may not be clear how all of the algorithms presented so far are connected. We now describe a generic matching pursuit algorithm that is in the form of the sparse KPCA and sparse KCCA algorithms presented so far. We then give an algorithm for kernel matching pursuit (KMP) in this generic form. Hopefully convincing the reader of the relationship between these three algorithms.

---

### Algorithm 4: A generic matching pursuit algorithm

---

**Input:** data  $\mathbf{X}$ , sparsity parameter  $k > 0$ .

- 1: initialize  $\mathbf{i} = ()$
- 2: **for**  $i = 1$  to  $k$  **do**
- 3: set  $\mathbf{i}_i$  to index of the maximization of a loss function that uses  $\mathbf{X}$ .
- 4: deflate vectors in  $\mathbf{X}$  according to basis vector chosen.

TABLE I  
FUNCTIONS TO MAXIMIZE FOR SEVERAL DIFFERENT MATCHING PURSUIT ALGORITHMS

|          | KMP                                                                     | SKPCA                                                | SKCCA                                                                                                  |
|----------|-------------------------------------------------------------------------|------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| MAXIMISE | $\left  \frac{(\mathbf{K}\mathbf{y})}{\sqrt{(\mathbf{K}^2)'1}} \right $ | $\frac{(\mathbf{K}^2)'1}{\text{diag}\{\mathbf{K}\}}$ | $\frac{((\mathbf{K}_x \cdot \mathbf{K}_y)'1)}{\sqrt{((\mathbf{K}_x^2)'1) \cdot ((\mathbf{K}_y^2)'1)}}$ |

5: **end for**

6: use final  $\mathbf{i}$  to construct subspace and carry out learning in this low dimensional subspace to find (sparse) parameters.

**Output:** final set  $\mathbf{i}$  and (sparse) parameters to make predictions, create low dimensional projection, etc.

In the generic matching pursuit algorithm described in Algorithm 4 we have two lines in the main For loop; One to maximize some function (line 3) and another to carry out deflation (line 4). We formulate in Table I the three distinct functions that need to be maximized in order to retrieve the three algorithms described so far; namely, kernel matching pursuit (KMP), sparse kernel principal components analysis (SKPCA) and sparse kernel canonical correlation analysis (SKCCA). For instance, KMP maximizes  $\left| \frac{(\mathbf{K}\mathbf{y})}{\sqrt{(\mathbf{K}^2)'1}} \right|$ . Earlier we presented an algorithm for KMP that doesn't quite fit into the generic algorithm described in Algorithm 4. Therefore, we give a more generic version of KMP in Algorithm 5 that maximizes a function, deflates and then uses the low dimensional subspace to carry out learning.

---

### Algorithm 5: Kernel matching pursuit (generic version)

---

**Input:** kernel  $\mathbf{K}$ , sparsity parameter  $k > 0$ .

- 1: initialize  $\mathbf{i} = ()$  to be empty
- 2: **for**  $i = 1$  to  $k$  **do**
- 3: set  $\mathbf{i}_i$  to the index of  $\max \left| \frac{(\mathbf{K}\mathbf{y})}{\sqrt{(\mathbf{K}^2)'1}} \right|$
- 4: set  $\boldsymbol{\tau} = \mathbf{K}[:, \mathbf{i}_i]$  to deflate kernel matrix like so:

$$\mathbf{K} = \left( \mathbf{I} - \frac{\boldsymbol{\tau}\boldsymbol{\tau}'}{\boldsymbol{\tau}'\boldsymbol{\tau}} \right) \mathbf{K}$$

5: **end for**

6: solve (1) with final  $\mathbf{i}$  to find  $\tilde{\boldsymbol{\alpha}}$

**Output:** index vector  $\mathbf{i}$  and sparse dual weight vector  $\tilde{\boldsymbol{\alpha}}$

The algorithms presented so far imply a general matching pursuit framework, and so we discuss now a potential matching pursuit style algorithm for classification—we will not give any experimental results or a complete algorithm, but rather use this section to show that the derivation of other matching pursuit algorithms is possible within the framework described.

Assume we have a sample  $S$  containing examples  $\mathbf{x} \in \mathbb{R}^n$  and labels  $y \in \{-1, 1\}$ . Then we can have a matching pursuit algorithm for Fisher discriminant analysis (see [25] for details) in

the following way. Initially, we may pick one example  $\mathbf{i} = \{i_1\}$  and project the remaining training examples into the space defined by  $\mathbf{i}$ . We would then attempt to find the index that maximizes the Fisher discriminant analysis (FDA) loss. After which a deflation of the data matrix  $\mathbf{X}$  (or kernel  $\mathbf{K}$ ) would be carried out to allow new training examples to be chosen. Finally giving us a set  $\mathbf{i}$  of training examples that can be used to compute the final weight vector  $\mathbf{w}$ , together with the FDA decision function  $f(\mathbf{x}) = \text{sgn}(\mathbf{w}'\mathbf{x} + b)$  where  $b$  is the bias and  $\mathbf{x}$  an example.

Recall that  $\mathbf{X}$  contains examples as column vectors. Using the notation from [25], we have the following maximization problem for FDA:

$$\mathbf{w} = \max_{\mathbf{w}} \frac{2m^+m^-}{m^3} \frac{\mathbf{w}'\mathbf{X}\mathbf{y}\mathbf{y}'\mathbf{X}'\mathbf{w}}{\mathbf{w}'\mathbf{X}\mathbf{B}\mathbf{X}'\mathbf{w}}$$

where  $m^+$  are the number of positive examples,  $m^-$  the number of negative examples and  $\mathbf{B} = \mathbf{D} - \mathbf{C}^+ - \mathbf{C}^-$  where  $\mathbf{D}$  is a diagonal matrix with entries

$$\mathbf{D}[i, i] = \begin{cases} 2m^-/m & \text{if } y_i = +1 \\ 2m^+/m & \text{if } y_i = -1 \end{cases}$$

and  $\mathbf{C}^+$  and  $\mathbf{C}^-$  are given by

$$\mathbf{C}^+[i, j] = \begin{cases} 2m^-/(mm^+) & \text{if } y_i = +1 = y_j \\ 0 & \text{otherwise} \end{cases}$$

and

$$\mathbf{C}^-[i, j] = \begin{cases} 2m^+/(mm^-) & \text{if } y_i = -1 = y_j \\ 0 & \text{otherwise.} \end{cases}$$

Following our pattern throughout the paper we can define the following maximization problem for a dual sparse version of LDA by setting  $\mathbf{w} = \mathbf{X}\mathbf{e}_i$  and substituting into the LDA problem described above (ignoring constants) to yield

$$\begin{aligned} \max_i \rho_i &= \frac{\mathbf{e}_i'\mathbf{X}'\mathbf{X}\mathbf{y}\mathbf{y}'\mathbf{X}'\mathbf{X}\mathbf{e}_i}{\mathbf{e}_i'\mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{X}'\mathbf{X}\mathbf{e}_i} \\ &= \frac{\mathbf{K}[:, i]'\mathbf{y}\mathbf{y}'\mathbf{K}[:, i]}{\mathbf{K}[:, i]'\mathbf{B}\mathbf{K}[:, i]}. \end{aligned}$$

After finding the best index  $i$ , we would orthogonalize the matrix  $\mathbf{K}$  using the same technique utilized by KMP and SKCCA, by setting  $\tilde{\boldsymbol{\tau}} = \mathbf{K}[:, i]$ , and deflating like so

$$\mathbf{K} = \left( \mathbf{I} - \frac{\boldsymbol{\tau}\boldsymbol{\tau}'}{\boldsymbol{\tau}'\boldsymbol{\tau}} \right) \mathbf{K}.$$

After choosing the  $k$  training examples, giving  $\mathbf{i} = (i_1, \dots, i_k)$ , we can define

$$\mathbf{K}[:, \mathbf{i}]\mathbf{R}$$

where  $\mathbf{R}$  is the Cholesky decomposition of  $\mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1}$ , and train FDA in this new projected space to find a sparse weight vector  $\tilde{\mathbf{w}}$  and make predictions. This work has been published (with a slightly different bound to those presented here) in the AISTATS conference of 2009 [39].

In this way, matching pursuit can be applied to other learning problems, by maximizing some function and then deflating, in order to find a set of basis vectors to create a low dimensional subspace in which to carry out learning.

## VI. CONCLUSION

In this paper we have addressed one important theoretical issue for matching pursuit, missing from the current literature, on how to upper bound the loss that may occur in the future (generalization error bounds). We proposed a novel bounding technique for kernel matching pursuit using traditional learning theory bounds and sample compression schemes. The bounds were valid for infinite dimensional feature spaces and could be used in a matching pursuit version of kernel canonical correlation analysis. We also proved that sparse kernel principal components analysis is a compression scheme and showed on some real world and simulated data sets that our bound is much tighter than the state-of-the-art bounds available for (kernel) principal components analysis. Finally, we described how to apply matching pursuit to other learning domains such as kernel canonical correlation analysis, and gave an example for classification using Fisher discriminant analysis.

We feel that matching pursuit can be applied to various other learning domains such as novelty detection, ranking, multiview learning, etc. We hope to have given the reader a flavour of the different domains that can be tackled and an approach for tackling new learning problems using matching pursuit. The bounds presented also theoretically validate the matching pursuit approach as a viable meta-learning procedure.

For future research we would like to tighten the bound presented for KMP, and feel perhaps a PAC-Bayesian [40] argument may result in tighter bounds. Another future direction for the sparse kernel canonical correlation analysis is to apply the bound to more than 1-dimension without having to resort to a union bound over all of the dimensions—which would loosen the bound. Finally, we did not discuss the computational complexity of the algorithms as more efficient techniques for finding the final weight vectors would certainly improve the speeds of the algorithms presented (as is the case with KMP with prefitting due to [6]). Ultimately though, we chose to present the algorithms without these speed-ups, to make clearer the connections between them and to state them in a general form.

## APPENDIX

### SMOLA AND SCHÖLKOPF'S LOW RANK MATRIX APPROXIMATION ALGORITHM

The following Algorithm 6 was first proposed by Smola and Schölkopf [5]. In Theorem 5 we proved its equivalence to Algorithm 2 and add it here for completeness.

---

**Algorithm 6:** A matching pursuit algorithm for learning a low rank matrix approximation [5] (i.e., sparse KPCA)

---

**Input:** kernel  $\mathbf{K}$ , sparsity parameter  $k > 0$ ,

1: an  $m \times 1$  column vector  $\hat{\mathbf{K}} = (0, \dots, 0)'$ , a  $1 \times m$  row vector  $\mathbf{T} = (0, \dots, 0)$  and  $\mathbf{i} = \emptyset$ .

2: **for**  $i = 1$  to  $k$  **do**

3: compute matrix approximation  $\tilde{\mathbf{K}} = \hat{\mathbf{K}}\mathbf{T}$

4: set  $\mathbf{i}_i$  to the index of  $\max \frac{\mathbf{1}'(\mathbf{K} - \tilde{\mathbf{K}})^2}{\text{diag}\{\mathbf{K}\} - \text{diag}\{\tilde{\mathbf{K}}\}}$

5: **if**  $i == 1$  **then**

$$6: \mathbf{T} = \frac{\mathbf{K}[:, \mathbf{i}_i]'}{\mathbf{K}[\mathbf{i}_i, \mathbf{i}_i]}$$

$$7: \hat{\mathbf{K}} = \mathbf{K}[:, \mathbf{i}_i]$$

8: else

$$9: \mathbf{t} = \left( \frac{-\mathbf{T}[:, \mathbf{i}_i]'}{\mathbf{K}[\mathbf{i}_i, \mathbf{i}_i] - \hat{\mathbf{K}}[\mathbf{i}_i, \mathbf{i}_i]}, \frac{1}{\mathbf{K}[\mathbf{i}_i, \mathbf{i}_i] - \hat{\mathbf{K}}[\mathbf{i}_i, \mathbf{i}_i]} \right)$$

$$10: \text{dot} = \mathbf{K}[:, \mathbf{i}_i] - \hat{\mathbf{K}}[:, \mathbf{i}_i]$$

$$11: \mathbf{T} = \begin{pmatrix} \mathbf{T} \\ 0, \dots, 0 \end{pmatrix} + \mathbf{t}' \text{dot}'$$

$$12: \hat{\mathbf{K}} = (\hat{\mathbf{K}} \quad , \quad \mathbf{K}[:, \mathbf{i}_i])$$

13: end if

14: end for

$$15: \text{compute } \tilde{\mathbf{K}} = \hat{\mathbf{K}} \mathbf{T}$$

**Output:** output sparse matrix approximation  $\tilde{\mathbf{K}}$

#### ACKNOWLEDGMENT

The authors would like to thank the reviewer and editor for their comments and suggestions which helped improve the readability of the paper. Z. Hussain would like to thank C. Archambeau for his comments on an earlier draft of the paper

#### REFERENCES

- [1] Z. Hussain and J. Shawe-Taylor, "Theory of matching pursuit," in *Adv. Neural Inf. Process. Syst. 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Vancouver, BC, Canada: MIT Press, 2009, pp. 721–728.
- [2] S. Mallat and Z. Zhang, "Matching pursuit with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, 1993.
- [3] Y. Pati, R. Rezaifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. 27th Ann. Asilomar Conf. Signals, Syst., Comput.*, 1993, pp. 40–45.
- [4] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, 1995.
- [5] A. J. Smola and B. Schölkopf, "Sparse greedy matrix approximation for machine learning," in *Proc. 17th Int. Conf. Mach. Learn.*, San Francisco, CA, 2000, pp. 911–918.
- [6] P. Vincent and Y. Bengio, "Kernel matching pursuit," *Mach. Learn.*, vol. 48, pp. 165–187, 2002.
- [7] P. B. Nair, A. Choudhury, and A. J. Keane, "Some greedy learning algorithms for sparse regression and classification with mercer kernels," *J. Mach. Learn. Res.*, vol. 3, pp. 781–801, 2003.
- [8] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Theory of Probabil. Appl.*, vol. 16, no. 2, pp. 264–280, 1971.
- [9] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
- [10] N. Littlestone and M. K. Warmuth, *Relating Data Compression and Learnability*. Santa Cruz, CA: Univ. Calif. Santa Cruz, 1986.
- [11] S. Floyd and M. Warmuth, "Sample compression, learnability, and the Vapnik-Chervonenkis dimension," *Mach. Learn.*, vol. 21, no. 3, pp. 269–304, 1995.
- [12] J. Langford, "Tutorial on practical prediction theory for classification," *J. Mach. Learn. Res.*, vol. 6, pp. 273–306, 2005.
- [13] R. Herbrich, *Learning Kernel Classifiers*. Cambridge, MA: MIT Press, 2002.
- [14] J. Shawe-Taylor, C. K. I. Williams, N. Cristianini, and J. Kandola, "On the eigenspectrum of the gram matrix and the generalization error of kernel-PCA," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2510–2522, 2005.
- [15] G. Blanchard, O. Bousquet, and L. Zwald, "Statistical properties of kernel principal component analysis," *Mach. Learn.*, vol. 66, no. 2–3, pp. 259–294, 2007.
- [16] P. L. Lai and C. Fyfe, "Kernel and nonlinear canonical correlation analysis," in *Proc. Int. Joint Conf. Neural Netw.*, 2000, vol. 4, pp. 4614–4614.
- [17] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *J. Mach. Learn. Res.*, vol. 3, pp. 1–48, 2003.
- [18] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neur. Computat.*, vol. 16, no. 12, pp. 2639–2664, 2004.
- [19] V. Koltchinskii, "Local Rademacher complexities and oracle inequalities in risk minimization," *Ann. Statist.*, vol. 34, no. 6, pp. 2593–2656, 2006.
- [20] P. L. Bartlett, O. Bousquet, and S. Mendelson, "Local rademacher complexities," *Ann. Statist.*, vol. 33, no. 4, pp. 1497–1537, 2005.
- [21] D. A. Torres, D. Turnbull, B. K. Sriperumbudur, L. Barrington, and G. R. G. Lanckriet, "Finding musically meaningful words by sparse CCA," in *Proc. Neural Inf. Process. Syst. (NIPS) Workshop on Music, the Brain and Cognition*, 2007.
- [22] B. K. Sriperumbudur, D. A. Torres, and G. R. G. Lanckriet, "Sparse eigen methods by d.c. programming," in *Proc. 24th Int. Conf. Mach. Learn.*, San Francisco, CA, 2007, pp. 831–838.
- [23] D. R. Hardoon and J. Shawe-Taylor, *Sparse Canonical Correlation Analysis*. London, U.K.: University College London, 2007.
- [24] A. Wiesel, M. Klinger, and A. O. Hero, III, "A Greedy Approach to Sparse Canonical Correlation Analysis 2008 [Online]. Available: <http://www.citebase.org/abstract?id=oai:arXiv.org:0801.2748>
- [25] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
- [26] B. Schölkopf and A. Smola, *Learning With Kernels*. Cambridge, MA: MIT Press, 2002.
- [27] P. Bartlett and J. Shawe-Taylor, "Generalization performance of support vector machines and other pattern classifiers," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 43–54.
- [28] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via  $\ell_1$  minimization," *Proc. Nat. Acad. Sci. USA*, vol. 100, no. 5, pp. 2197–2202, 2003.
- [29] J. Tropp, "Just relax: Convex programming methods for identifying sparse signals in noise," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1030–1051, Mar. 2006.
- [30] D. L. Donoho and M. Elad, "On the stability of the basis pursuit in the presence of noise," *Signal Process.*, vol. 86, no. 3, pp. 511–532, 2006.
- [31] D. L. Donoho, M. Elad, and V. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Trans. Inf. Theory*, vol. 52, pp. 6–18, 2006.
- [32] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Scientific Comput.*, vol. 20, no. 1, pp. 33–61, 1998.
- [33] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.
- [34] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5406–5425, Dec. 2006.
- [35] M. Anthony, "Partitioning points by parallel planes," *Discr. Math.*, vol. 282, pp. 17–21, 2004.
- [36] C. K. I. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2001, vol. 13, pp. 682–688.
- [37] C. B. D. J. Newman, S. Hettich, and C. Merz, UCI Repository of Machine Learning Databases 1998 [Online]. Available: <http://www.ics.uci.edu/~mllearn/MLRepository.html>
- [38] P. Koehn, Europarl: A Parallel Corpus for Statistical Machine Translation 2005 [Online]. Available: <http://www.statmt.org/europarl/>
- [39] T. Diethe, Z. Hussain, D. Hardoon, and J. Shawe-Taylor, "Matching pursuit kernel fisher discriminant analysis," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, 2009, vol. 5, pp. 121–128.
- [40] D. McAllester, "Some PAC-Bayesian theorems," *Mach. Learn.*, vol. 37, pp. 355–363, 1999.

**Zakria Hussain** received the B.Sc. (Hons.) degree in mathematics and computer science, and the M.Sc. degree in machine learning with Distinction, both from Royal Holloway, University of London, U.K., in 2002 and 2003, respectively, and the Ph.D. degree in machine learning from the University College London in 2008.

He is currently with University College London as a Research Fellow, with research interests ranging from computational learning theory, sparsity, and kernel methods.

**John Shawe-Taylor** (M'98) received the Ph.D. degree in mathematics from Royal Holloway, University of London, U.K., in 1986. He received the M.Sc. degree from the Foundations of Advanced Information Technology, Imperial College, London, in 1987.

He was promoted to Professor of Computing Science in 1996. He has published more than 150 research papers. He moved to the University of Southampton in 2003 to lead the ISIS research group. He was appointed Director of the Centre for Computational Statistics and Machine Learning at University College, London (UCL), from July 2006—September 2010. He has coordinated a number of European-wide projects investigating the theory and practice of machine learning, including the NeuroCOLT projects. He is currently the scientific coordinator of a Network of Excellence in Pattern Analysis, Statistical Modelling and Computational Learning (PASCAL 2) and also the head of the Computer Science Department, UCL. His main research area is statistical learning theory, but his contributions range from neural networks, to machine learning, to graph theory.

**David R. Hardoon** received the B.Sc. (Hons.) degree in computer science and artificial intelligence with first class honors at Royal Holloway, University of London, U.K., in 2002, and the Ph.D. degree in machine learning from the University of Southampton, U.K., in 2006.

He is with Principal Analytics, SAS Singapore, where he is engaged in positioning SAS advanced analytic capabilities and solutions to customers across different business sectors. Previous to his current engagement, he had established expertise in developing and applying computational analytical models for business knowledge discovery and analysis through his involvement in a number of research projects in the domains of taxonomy, neuroscience, aerospace, and finance. His research interests are in machine learning. He is currently an Adjunct Assistant Professor with the School of Computing, National University of Singapore.

Dr. Hardoon is an Honorary Senior Research Associate with the Centre for Computational Statistics and Machine Learning, University College London and a visiting Research Fellow with the Institute of Psychiatry, King's College London.

**Charanpal Dhanjal** received the Ph.D. degree from the Image, Speech and Intelligent Systems group, School of Electronics and Computer Science, University of Southampton, in 2009.

He is currently a Postdoctoral Researcher with the Department of Image and Signal Processing, Telecom ParisTech.