SparkDict: A Fast Dictionary Learning Algorithm

Tobias Schnier, Carsten Bockelmann, Armin Dekorsy
Department of Communications Engineering
University of Bremen, Germany
Email: {schnier,bockelmann,dekorsy}@ant.uni-bremen.de

***Abstract*—For the always increasing amount of data new tools are needed to effectively harvest important information out of them. One of the core fields for data mining is Dictionary Learning, the search for a sparse representation of given data, which is widely used in signal processing and machine learning. In this paper we present a new algorithm in this field that is based on random projections of the data. In particular, we show that our proposition needs a lot less training samples and is a lot faster to achieve the same dictionary accuracy as state of the art algorithms, especially in the medium to high sparsity regions. As the *spark*, the minimum number of linear dependent columns of a matrix, plays an important role in the design of our contribution, we coined our contribution *SparkDict*.**

***Index Terms*—Dictionary Learning, Spark, Sparsity**

I. Introduction

A. Motivation

In any signal processing experiment with a lot of data points, the question of the intrinsic structure of the data emerges. For example, in neuroscience there is an ongoing trend towards wireless implants, which would give patients an overall higher quality of life. Unfortunately, the high data rates of activating neurons prohibit wireless transmission of the raw data without further refining them. To counter this, state of the art transmitters only detect and send spike positions and amplitudes [1]. Even with this limited amount of data, highly functional prostheses have been developed that can be controlled directly by the brain implant ([2],[3]). Today, this is sufficient, but next generation products need to increase the transmitted data rate within the same tight hardware restrictions, so that even delicate movements may be implemented.

This possible advance in neuroscience inspired us to take the signal processing perspective on the problem of finding the sparse structure of the data. In signal processing, the two intertwined frameworks *Compressed Sensing* (CS)[4] and *Dictionary Learning* (DL)[5] directly come to mind. In CS, a fitting dictionary is used to reconstruct sparsely representable vectors out of few measurements. There are several important state of the art algorithms with OMP [6] being the most prominent one. The sparsity requirement here is crucial as the CS framework and algorithms highly depend on it. Normally a reasonable dictionary for the sparse representation (e.g. Wavelet basis for neural data) is assumed, but learning the dictionary out of samples naturally yields better results.

This so called Dictionary Learning framework consists of many algorithms that fulfill the sparsity requirement of CS for given data. An important tool to solve this problem is the *Principal Component Analysis* (PCA) [7] which finds basis vectors for the data in a way that they are as mutually orthogonal as possible. This way the most important variables of the data can be easily extracted. Unfortunately, this representation is hard to interpret. Due to the fact that all PCA coefficient vectors are densly filled, all basis vectors are needed to represent any single sample. If there are only a few sources that get captured by the experiment, there should be other basis vectors that lead to *sparse* linear combinations in a trade off with a higher correlation between the basis vectors. Hence, such a procedure is coined *sparse Principial Component Analysis* (sPCA) [5].

On the dictionary learning side, there exist multiple iterative algorithms, most prominently **K-SVD** ([8],[9]) and **MOD** [10], that alternate between improving the dictionary and the corresponding coefficient matrix. These algorithms work slowly and need a lot of samples to work properly but converge to sufficient results. Furthermore, both need an additional sparse reconstruction algorithm from the CS framework to update the intermediate sparse coefficient matrix. On the sPCA side mostly convex optimization techniques [5] with l_1-norm restricted objective functions like the **LASSO** [11], **ERSpuD** [12] or similar penalty terms [13] are used. Here problems arise from the many tuning parameters that need to be optimized with regard to an unknown dictionary and the very slow computation speed of most optimization algorithms. The main motivation for our contribution is the search for a fast dictionary learning algorithm that overcomes the three main problems of state of the art algorithms: the slow iterative update, the dependency on CS solving algorithms and the need for tuning parameters that are hard to control.

B. Main Contribution

Our main contribution is the *SparkDict* algorithm that is inspired by random sub sampling. With enough samples of the form $\mathbf{Y} = \mathbf{\Psi C}$ with sparse $\mathbf{C}$, the algorithm finds the underlying dictionary $\mathbf{\Psi}$ by consecutively buildings rows of $\mathbf{\Psi}^{-1}$. If the optimal dictionary consists of d atoms and $\mathbf{C}$ has k nonzero elements per column, the algorithm needs much less samples (in the order of $d(d-k)$ samples) than state of the art algorithms. In contrast to most existing algorithms, the coefficient matrix $\mathbf{C}$ of our contribution is analytically sparse and not only tends to zero with high iterations. In fact, due to the non iterative fashion, the computational complexity is very low. Furthermore, in the noiseless case there is no need to tune parameters or other sparse approximation algorithms when using SparkDict, which makes it very versatile and easy to use.

ISBN 978-0-9928626-7-1

II. Theoretical Background

A. Introduction to dictionary learning

Formally spoken, the dictionary learning problem is the extraction of an optimal dictionary $\mathbf{\Psi} \in \mathbb{R}^{N\times d}$ and the corresponding coefficient matrix $\mathbf{C} \in \mathbb{R}^{d\times m}$ out of m given noisy samples $\mathbf{Y} \in \mathbb{R}^{N\times m}$, that are known to be representable in a sparse way. First, To get a non-degenerate problem, the condition

$$d \leq N \tag{1}$$

is required, because allowing $d > N$ would lead to an over-complete problem. Then, there would be a trade off between the number of dictionary atoms d and the minimum sparsity of the corresponding $\mathbf{C}$ with a guaranteed minimal sparsity of $\mathbf{C} = \mathbf{I}$ for $d = m$. As our goal is to find the most compact representation for $\mathbf{Y}$, we concentrate on the better posed (under-)complete case $d \leq N$.
Second, the "coupon collection phenomenon" [12] shows, that

$$m \geq d\log(d) \tag{2}$$

has to be fulfilled to reduce the probability p that any dictionary element does not occur at all to $p < \frac{1}{d}$. If the sparsity k increases, this becomes easier to accomplish, because more dictionary elements are present in each sample. At the same time, for every dictionary element there should be at least one sample where that element did *not* occur to be able to find inherit structure inside the dictionary. As this condition gets easier for lower sparsity, for each dictionary size d there exists a sweet spot sparsity $k(d)$ that leads to a well conditioned problem. A more formal discussion is presented in the numerical simulations section IV, in which simulations under different conditions are presented that better clarify these borders.
To solve the dictionary learning problem, two conditions need to be fulfilled. First, the consistency condition

$$||\mathbf{Y} - \mathbf{\Psi}\mathbf{C}||_F < \epsilon \tag{3}$$

with ϵ as the noise variance needs to be met to ensure the validity of the computation. Second, the coefficient matrix $\mathbf{C}$ should be *as sparse as possible* (minimal number of nonzero elements) to highlight the underlying structure of the signal.
Most state of the art algorithms solve the dictionary learning task by alternating between improving $\mathbf{\Psi}$ and $\mathbf{C}$ in an iterative fashion. This approach leads to very high computational complexity and difficult convergence proofs. Also it requires an additional independent algorithm to find a sparse solution for $\mathbf{C}$ with intermediate non-optimal $\mathbf{\Psi}$.
In this paper we present a faster alternative. Instead of learning the columns of $\mathbf{\Psi}$, an equivalent problem is to learn orthogonal complements of each hyperspace spanned by all columns of $\mathbf{\Psi}$ but exactly one missing. If enough data about the hyperspaces is collected, $\mathbf{\Psi}$ can be easily computed and $\mathbf{C}$ emerges with the standard solution of

$$\min_{\mathbf{C}} ||\mathbf{Y} - \mathbf{\Psi}\mathbf{C}||_F \tag{4}$$

that is sparse by design of the dictionary. As mentioned before, the *spark* plays an important role as part of our contribution, so it will be introduced in the next section more formally.

B. The spark of a matrix

The *spark* [14] (a neologism from *sparse* and *rank*) of a matrix $\mathbf{A}$ defines a similar attribute as its rank. While the rank is the **maximum** number of linear **independent** columns, the spark is the **minimum** number of linear **dependent** columns. Formally that means

$$\text{spark}(\mathbf{A}) = \min_{\mathbf{x}\neq 0} ||\mathbf{x}||_0 \text{ s.t. } \mathbf{A}\mathbf{x} = 0. \tag{5}$$

In the Compressed Sensing theory [15] the spark is an important measure to determine reconstruction guarantees, because the spark describes the cardinality of the sparsest vector in the null space. If one wants to reconstruct every k-sparse $\mathbf{x}$ out of $\mathbf{y} = \mathbf{A}\mathbf{x}$, this can only work if $\text{spark}(\mathbf{A}) > 2k$. Otherwise there exists a $2k$-sparse $\mathbf{x}_{\text{null}} \in \text{null}(\mathbf{A})$ and a fitting k-sparse vector $\mathbf{x}$ s.t.

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \mathbf{A}(\mathbf{x} + \mathbf{x}_{\text{null}}) = \mathbf{A}\hat{\mathbf{x}} \tag{6}$$

with $\hat{\mathbf{x}}$ being k-sparse, thus preventing the recovery. This directly leads to a requirement towards the minimum number of measurements $m \geq 2k$. If there is more than one minimal set of columns that build the spark, several columns in $\mathbf{\Psi}$ will have the same l_0-norm and the nonzero elements correspond to the used columns of $\mathbf{A}$. A simple reformulation can be used to extract all spark-columns. It has been shown that computing the spark of arbitrary given matrix is NP-hard [16]. However, there is one interesting case where the computation can be done in an efficient way. If $\mathbf{A} \in \mathbb{R}^{N-1\times N}$ is a full rank matrix, its null space is spanned by one vector $\mathbf{v}$. It follows with $\mathbf{v} = \text{null}(\mathbf{A})$ that

$$\text{spark}(\mathbf{A}) = \sum_{i=1}^{N} (\mathbf{v}_i \neq 0) = ||\mathbf{v}||_0. \tag{7}$$

We show through our contribution, that the dictionary learning problem can make good use of the spark inside the computation. Another crucial concept of our contribution is the computation of *orthogonal* complements. The next section will introduce the concept more formally.

C. Orthogonal complements

Let V, W be two vector spaces with $V \subset W$. Then the *orthogonal complement* of V in W is formally defined as

$$V^{\perp} := \{\mathbf{w} \in W | \mathbf{w}^{\perp}\mathbf{v} = 0 \quad \forall \quad \mathbf{v} \in V\} \tag{8}$$

In other words the orthogonal complement $V^{\perp}$ is the vector space that is perpendicular to all vectors in V. It is easy to see that

$$\dim(V^{\perp}) = \dim(W) - \dim(V) \tag{9}$$

It directly follows that the orthogonal complement $H^{\perp}$ of any hyperspace H in $\mathbb{R}^N$ has dimension 1. That is $H^{\perp}$ is spanned by the normal vector on H.

ISBN 978-0-9928626-7-1 © EURASIP 2017

If $\mathbf{v}_1, \cdots, \mathbf{v}_N$ are linearly independent vectors in $\mathbb{R}^N$ one can span N different N-1 dimensional hyper spaces by

$$H_i = \text{span}\{\mathbf{v}_j | j \in \{1, \ldots, N\}, j \neq i\} \tag{10}$$

With $\mathbf{v}_1^\perp, \cdots, \mathbf{v}_N^\perp$ as the corresponding normal vectors (that is spanning the orthogonal complement) of $H_1, \cdots, H_N$, the identity

$$\left(\mathbf{v}_1^\perp, \quad \cdots \quad, \mathbf{v}_N^\perp\right)^{-1} = \left(\mathbf{v}_1, \quad \cdots, \quad \mathbf{v}_N\right)^\top \tag{11}$$

holds up to a scaling constant. The orthogonal complement in $\mathbb{R}^N$ of such hyperspaces $\mathbf{V} := (\mathbf{v}_1, \cdots, \mathbf{v}_{N-1})$ of size $N-1$ is given by its *left null space* or equivalently the *right null space of its transpose* $\mathbf{V}^\top$

$$\left(\mathbf{v}_1, \quad \cdots \quad, \mathbf{v}_{N-1}\right)^\perp = \text{null}\left(\mathbf{v}_1, \quad \cdots \quad, \mathbf{v}_{N-1}\right)^\top \tag{12}$$

The next section will now clarify in more details, how the *SparkDict* dictionary learning algorithm exploits orthogonal complements to build hyperspaces of $N-1$ dictionary elements. To present the main reasoning more clearly, in we first examine the noiseless case and then later adapt the algorithm to cope with noise.

III. Design of SparkDict

Most dictionary learning approaches try to find few dictionary elements that can well represent a high amount of the data. But to get a good understanding of the data, the represented part is as important as the *missing* pieces. Imagine choosing a set of columns $\mathbf{Y}_I$ taken out of the samples $\mathbf{Y}$, corresponding to an index set I in a way that **all** N dictionary elements **but one** are present. Then this set can be used to compute the orthogonal complement of $\mathbf{Y}_I$. If for every dictionary element $\mathbf{\Psi}_i$ a set of columns $\mathbf{Y}_I$ can be found where only this i-th element is missing, the dictionary can be reconstructed. Consequently, the main problem of this section is how to find a suitable set of samples out of $\mathbf{Y}$.

A. The search for suitable index sets

For a given set I, it is possible to find the amount of directions spanned by $\mathbf{Y}_I$ by computing its rank. If $\text{rank}(\mathbf{\Psi}) = d$ and thus $\text{rank}(\mathbf{Y}) = d$ as well and for one subset $\mathbf{Y}_I$ the condition $\text{rank}(\mathbf{Y}_I) < d$ holds, it is very likely (but not guaranteed) that one dictionary element is missing in $\mathbf{Y}_I$. Without any further knowledge about the samples the best solution is to pick random subsets I of size d and test $\mathbf{Y}_I$ for its rank. If the rank is $d-1$ two possible *example* outcomes can occur ($*$ stands for nonzero entries):

$$\mathbf{Y}_I = \mathbf{\Psi}\begin{pmatrix} * & 0 & * & 0 & * \\ 0 & * & * & 0 & 0 \\ 0 & * & 0 & * & * \\ * & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \text{ or } \mathbf{Y}_I = \mathbf{\Psi}\begin{pmatrix} * & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & * \\ 0 & * & * & * & * \\ 0 & 0 & 0 & * & 0 \\ * & 0 & 0 & 0 & 0 \end{pmatrix}$$

Both systems have rank $d-1$, but only the left system is subject of missing one component of $\mathbf{\Psi}$ (the last column in this example). In contrast to that, the right system has a linear dependency between the second, third and fifth column of $\mathbf{Y}_I$ with only two present columns of the dictionary inside them, so again a rank deficiency occurs. To differentiate between these two systems without knowledge over $\mathbf{C}$, we use the **spark** of the subset. Due to $\mathbf{Y} = \mathbf{\Psi}\mathbf{C}$ with no kernel in $\mathbf{\Psi}$,

$$\text{null}(\mathbf{Y}_I) = \text{null}(\mathbf{C}_I) \quad \forall I \tag{13}$$

holds and consequently

$$\text{spark}(\mathbf{Y}_I) = \text{spark}(\mathbf{C}_I) \quad \forall I \tag{14}$$

follows. If $\text{spark}(\mathbf{Y}_I) < d$, it follows that a linear dependency happened, thus the subset needs to be dismissed. However, if

$$\text{rank}(\mathbf{Y}_I) = d-1 \text{ and } \text{spark}(\mathbf{Y}_I) = d, \tag{15}$$

exactly one column of $\mathbf{\Psi}$ is missing. The orthogonal complement of $\mathbf{Y}_I$ can then be used to determine the corresponding orthogonal complement of all columns of $\mathbf{\Psi}$ that were present in the underlying source structure. Thus, *one row of* $\mathbf{\Psi}^{-1}$ *can be computed*. To test if both conditions for the randomly drawn set are fulfilled, the results from section II can be utilized. The orthogonal complement can be easily computed with equation (12) and the spark computation follows equation (7).
In other words, the left and right null space of $\mathbf{Y}_I$ need to be computed, which can be done by the standard SVD. This step is repeated d times for different missing columns of $\mathbf{\Psi}$, until all orthogonal complements $\mathbf{\Psi}_i^\perp$ are present. With this information, we can compute $\mathbf{\Psi}$ as the inverse of $\mathbf{\Psi}^\perp$. The resulting sparse coefficient matrix $\mathbf{C}$ is then computed by

$$\mathbf{C} = \mathbf{\Psi}^\perp \mathbf{Y}. \tag{16}$$

In the real world, measurements are corrupted by noise, so rank and null space computations are meaningless. However, this can easily be altered by using the aforementioned SVD and testing the lowest singular value and its corresponding singular vector. The complete algorithm with regard to noise is depicted in algorithm 1. Finding an appropriate threshold tol (set to 10^-3 in the algorithm) for the noise is not an easy task and can only be chosen via a sufficiently good model assumption.

Algorithm 1 SparkDict

Require: $\mathbf{Y}$, tol
  $\mathbf{D}$=svd($\mathbf{Y}$), d =sum($\mathbf{D}$ >tol), founditems= 0, $\mathbf{\Psi} = [\,]$
  while founditems< d **do**
    Get random subset I of size d
    $[\mathbf{U}, \mathbf{D}, \mathbf{V}]$ =SVD($\mathbf{Y}_I$)
    if sum($\mathbf{D}$ >tol) == $d-1$ and
    sum(abs($\mathbf{V}$(:, end) >tol) == d **then**
      if $\mathbf{U}$(:, end) is not in $\mathbf{\Psi}^\perp$ **then**
        $\mathbf{\Psi}^\perp$(:, founditems) = $\mathbf{U}$(:, end), founditems++
      end if
    end if
  end while
  Compute $\mathbf{\Psi} = (\mathbf{\Psi}^\perp)^{-1}$
  Compute $\mathbf{C} = \mathbf{\Psi}^\perp \mathbf{Y}$
  return $\mathbf{\Psi}$, $\mathbf{C}$

ISBN 978-0-9928626-7-1 © EURASIP 2017

B. Numerical improvements

For high d, the SVD computation becomes computationally prohibitive. Luckily, this problem can be avoided. We discard the higher singular values without using them, so a direct computation of the lowest singular values would be enough. Due to the dominance of the highest singular values this is not directly possible, but the following idea can be used to avoid this problem: Let $\lambda_1, \cdots, \lambda_N$ be the singular values of a matrix $\mathbf{A}$. It directly follows, that the singular values of $\mathbf{A}_2 := \lambda_1^2\mathbf{I} - \mathbf{A}^\top\mathbf{A}$ are $0, \lambda_1^2 - \lambda_2^2, \cdots, \lambda_1^2 - \lambda_N^2$. As λ_1 is the biggest singular value, the highest difference now is $\lambda_1^2 - \lambda_N^2$. Additionally, the right singular vectors of $\mathbf{A}$ and $\mathbf{A}_2$ are identical, because the matrix is only shifted by a constant diagonal term. Due to this fact, the **dominant** singular value of $\mathbf{A}_2$ corresponds to the **lowest** singular value of $\mathbf{A}$. In other words, only the dominant singular value of $\mathbf{A}$ and $\mathbf{A}_2$ needs to be computed (for example via power iteration) to identify the null space. If both values are identical, $\lambda_1^2 - \lambda_N^2 = \lambda_1^2 \Rightarrow \lambda_N = 0$ follows, so that the corresponding singular vector is the *right* null space vector. If not, the subset can be discarded as there is no null space present. Repeating this process for $\mathbf{A}_3 := \lambda_1^2\mathbf{I} - \mathbf{A}\mathbf{A}^\top$ gives rise to the corresponding *left* null space vector. This procedure leads to a vast decrease in computation time, especially in high dimensions since the computation complexity of power iteration is only $\mathcal{O}(d^2)$[17], a substantial difference to the $\mathcal{O}(d^3)$[17] complexity of the standard SVD.

C. Further application: Spark computation of arbitrary matrices

The spark of a matrix $\mathbf{A}$ is defined as the cardinality of the sparsest element in the kernel of $\mathbf{A}$ (cf. (5)). In other words if we find a sparse basis for the null space $\mathbf{Y} = \text{null}(\mathbf{A})$, we can compute the number of nonzero elements in each basis vector and use the best fit $\mathbf{Y}_{\text{min}}$. If the computation of the sparse basis was optimal, it follows that

$$\|\mathbf{Y}_{\text{min}}\|_0 = \text{spark}(\mathbf{A}) \tag{17}$$

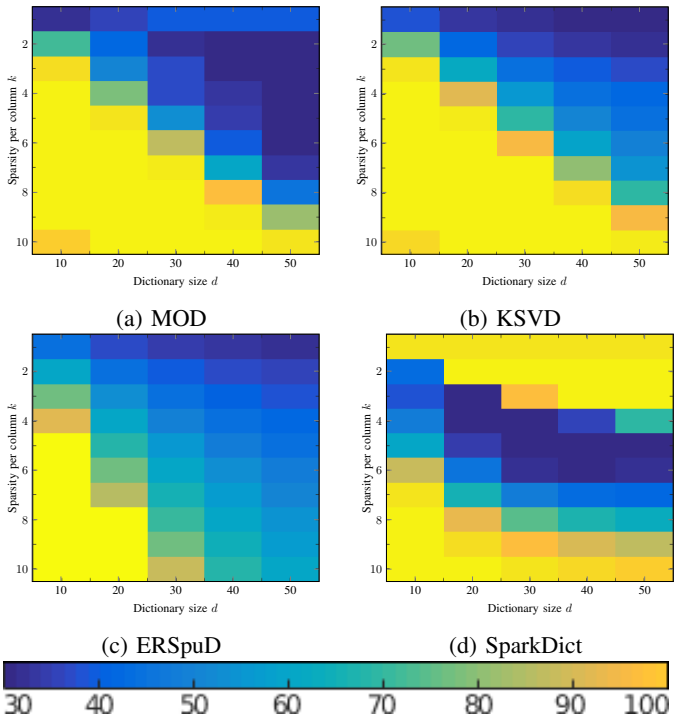
Normally the dictionary learning algorithm described above tries to find a sparse $\mathbf{C}$, so we need to reformulate it by transposing the problem. Due to

$$\mathbf{Y} = \mathbf{\Psi}\mathbf{C} \text{ with } \mathbf{C} \text{ sparse } \Rightarrow \mathbf{Y}^\top = \mathbf{C}^\top\mathbf{\Psi}^\top \tag{18}$$

we can use any dictionary learning approach on $\mathbf{Y}^\top$ to get the *sparse coefficient matrix* $\mathbf{C}$, so that the *sparse basis* $\mathbf{Y} = \mathbf{C}^\top$ for the kernel of $\mathbf{A}$ emerges.
This idea can be combined with every state of the art dictionary learning algorithm, but as most of them only tend to zero and SparkDict analytically enforces zeros the computed spark for SparkDict will be much more precise.

IV. Comparison to other algorithms

This section compares our contribution with three other state of the art algorithms, namely the K-SVD [9] and MOD [10] algorithm with their intrinsic OMP solver [6] and the convex optimization algorithm ERSpuD [12]. Figure 1 shows



(e) Relative mean sparsity in percent

Fig. 1: Relative mean sparsity for different dictionary sizes (x-axis) and sparsities (y-axis) with $m = 5d \cdot \log(d)$ after 1000 trials.

the relative mean sparsity in $\mathbf{C}$ that was achieved through the computation, depicted versus the sample size. In the simulated setting, several synthetic Gaussian ($\mu = 0, \sigma = 1$) dictionaries with varying sizes $\mathbf{\Psi} \in \mathbb{R}^{d\times d}$ for $10 \leq d \leq 50$ were created and multiplied with a coefficient matrix $\mathbf{C} \in \mathbb{R}^{d\times 5d\cdot\log(d)}$ that had $1 \leq k \leq 10$ non-zero Gaussian ($\mu = 0, \sigma = 1$) random elements in each of the columns. After 1000 trials the relative mean sparsity per column of $\mathbf{C} = \mathbf{\Psi}\backslash\mathbf{Y}$ of all results was taken and depicted in a 2D plot (see figure 1) against the dictionary size d and the sparsity k. Here, blue (dark) areas describe low sparsity regions while yellow (light) areas show that the algorithm failed (see the attached color map). Due to our construction of $\mathbf{\Psi}$ we can guarantee that in the noiseless case the small entries in $\mathbf{C}$ are in the order of 10^{-10} in contrast to K-SVD and MOD which only tend to zero. To achieve a fair comparison figure we used the threshold $t_{\text{thresh}} = 10^{-2}$ as border to zero.

A. Numerical results

In comparison, the four algorithms show completely different behaviors. K-SVD and MOD both perform best for low sparsities as their approach of finding the dominant eigenvector works best in that case. Up to around a sparsity k of 10 percent of the dictionary size, they nearly get a perfect result. K-SVD shows a steep ascend around that border until it breaks consistently for higher sparsities while outperforming MOD in the very low sparsity region. On the other side, MOD shows

ISBN 978-0-9928626-7-1 © EURASIP 2017

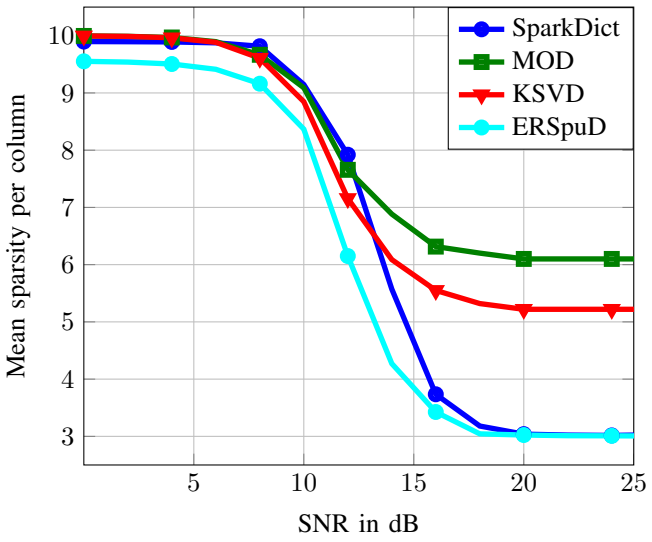


Fig. 2: Mean sparsity in the presence of noise

an overall gradual ascend with slightly better overall results. ERSpuD achieves the best results at the cost of computational complexity. Again, lower sparsity regions are better suited with a much higher breaking point at around 25 percent (off the chart for $d \geq 40$). Unfortunately, the computation time increases dramatically for higher d as one complex optimization problem has to be solved for every single data point. If speed is not an issue, ERSpuD is a versatile and powerful algorithm. Our contribution, *SparkDict*, shows a completely different behavior. As mentioned in section II, a very high and a very low sparsity lead to problems as they both make it hard to find a subset of columns of the data with exactly one missing column of the dictionary. Because of this, there is a sweet spot in the middle region of the sparsity, in which SparkDict vastly outperforms the other algorithms. Because the working regions of K-SVD and SparkDict are nearly non overlapping, a combination of both algorithms will lead to a powerful and versatile algorithm with much less computation complexity than ERSpuD. Especially, it is possible to do an a posteriori evaluation regarding the number of nonzero elements in the coefficient matrix and then choose the corresponding better dictionary. Figure 2 depicts the four algorithms in the presence of noise. With $m = 120$ samples and $d = 10$ dictionary elements, the mean sparsity per column is plotted against the SNR in dB. Here SparkDict shows a unique behavior. In the low SNR region, the algorithm completely fails to find the fitting hyperplanes and thus can not increase the sparsity in $\mathbf{C}$. At around 20 dB, the plot shows a steep descent as the missing elements can be detected correctly. At the same time, the speed increases, because the samples get more reliable, so correct orthogonal complements can be detected more easily. In contrast to that, K-SVD and MOD show a more fluent increase of the overall performance in the low SNR region, but then saturate at around 25 dB into the (in comparison worse) convergence point. Their computational complexity is far worse in comparison to SparkDict, with MOD still outperforming K-SVD. ERSpuD achieves the best performance by getting to the optimum even in lower SNR regions, but again at the price of the highest computational complexity by far.

V. Conclusion

We presented the new dictionary learning algorithm *SparkDict* which uses orthogonal complements and computations of the spark. This algorithm increases the utility of dictionary learning as it works best in regions where state of the art algorithms like K-SVD and MOD break and SparkDict fails in already utilized regions. Additionally, the algorithm can run without the need for additional third party algorithms and does not need to be tuned towards the concrete problem.

VI. Acknowledgment

This work was funded by the German Research Foundation (DFG) under grant DE 759/5-1.

References

[1] T. Jochum, T. Denison, and P. Wolf, "Integrated circuit amplifiers for multi-electrode intracortical recording," *Journal of neural engineering*, vol. 6, no. 1, p. 012001, 2009.
[2] M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz, "Cortical control of a prosthetic arm for self-feeding," *Nature*, vol. 453, no. 7198, pp. 1098–1101, 2008.
[3] A. B. Schwartz, X. T. Cui, D. J. Weber, and D. W. Moran, "Brain-controlled interfaces: movement restoration with neural prosthetics," *Neuron*, vol. 52, no. 1, pp. 205–220, 2006.
[4] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18 914–18 919, 2009.
[5] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of computational and graphical statistics*, vol. 15, no. 2, pp. 265–286, 2006.
[6] T. T. Cai and Lie Wang, "Orthogonal matching pursuit for sparse signal recovery with noise," *Information Theory, IEEE Transactions on*, vol. 57, no. 7, pp. 4680–4688, 2011.
[7] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.
[8] M. Aharon, M. Elad, and A. Bruckstein, "k-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.
[9] Mairal, Julien et al., "Online dictionary learning for sparse coding," 2009.
[10] K. Engan, S. O. Aase, and J. H. Husoy, Eds., *Method of optimal directions for frame design*, 1999.
[11] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.
[12] D. A. Spielman, H. Wang, and J. Wright, "Exact recovery of sparsely-used dictionaries," *arXiv preprint arXiv:1206.5882*, 2012.
[13] H. Shen and J. Z. Huang, "Sparse principal component analysis via regularized low rank matrix approximation," *Journal of multivariate analysis*, vol. 99, no. 6, pp. 1015–1034, 2008.
[14] R. Gribonval and M. Nielsen, "Sparse representations in unions of bases," *Information Theory, IEEE Transactions on*, vol. 49, no. 12, pp. 3320–3325, 2003.
[15] D. L. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.
[16] A. M. Tillmann and M. E. Pfetsch, "The computational complexity of the restricted isometry property, the nullspace property, and related concepts in compressed sensing," *Information Theory, IEEE Transactions on*, vol. 60, no. 2, pp. 1248–1259, 2014.
[17] G. H. Golub and Van Loan, Charles F, *Matrix computations*. JHU Press, 2012.