FAST NON-NEGATIVE ORTHOGONAL LEAST SQUARES

Mehrdad Yaghoobi and Mike E. Davies

Institute for Digital Communications, the University of Edinburgh, EH9 3JL, UK
{m.yaghoobi-vaighan, d.wu and mike.davies}@ed.ac.uk

ABSTRACT

An important class of sparse signals is the non-negative sparse signals. While numerous greedy techniques have been introduced for low-complexity sparse approximations, there are few non-negative versions. Among such a large class of greedy techniques, one successful method, which is called the Orthogonal Least Squares (OLS) algorithm, is based on the maximum residual energy reduction at each iteration. However, the basic implementation of the OLS is computationally slow. The OLS algorithm has a fast implementation based on the QR matrix factorisation of the dictionary. The extension of such technique to the non-negative domain is possible. In this paper, we present a fast implementation of the non-negative OLS (NNOLS). The computational complexity of the algorithm is compared with the basic implementation, where the new method is faster with two orders of magnitude. We also show that, if the basic implementation of NNOLS is not computationally feasible for moderate size problems, the proposed method is tractable. We also show that the proposed algorithm is even faster than an *approximate* implementation of the non-negative OLS algorithm.

Index Terms— Non-negative sparse approximations, Orthogonal Least Squares, Efficient Implementations, Non-negative Orthogonal Least Squares and QR Matrix Factorization

1. INTRODUCTION

The signal processing with a low-dimensional model, is a modern technique to explore signal structures. We thus need some low-dimensional signal approximation algorithms for this purpose. Such methods have to be fast, if the dimension of data is large/huge. There are many general purpose sparse approximation techniques with fast implementations. In some applications like hyperspectral data decomposition [1] and DNA microarrays analysis [2], we further know that non-zero coefficients of the sparse approximation are positive. In this setting, non-negative sparse approximation techniques [3], often provide better solutions than the universal techniques. Universal greedy techniques can be adapted to such a setting by confining the solutions to the set of non-negative coefficients. However, extension of the fast implementations of the greedy methods to the non-negative setting is often not straightforward. An example of such extensions to a greedy method, with a complicated coefficient update *e.g.* Orthogonal Matching Pursuit (OMP), has already been presented in [4]. We here present a fast implementation of the non-negative OLS [5] technique.

Let $\mathbf{y} \in \mathbb{R}^m$ be a signal with the sparse representation using a normalised linear generative model $\mathbf{\Phi} \in \mathbb{R}^{m\times n}$, *i.e.* dictionary, where $\mathbf{y} = \mathbf{\Phi}\mathbf{x}$ and $\mathbf{x}$ is a sparse vector. At each iteration, OLS selects an element of dictionary, called an atom, which maximally reduces the residual energy of the signal, after adding the atom to the set of previously selected atoms. It then finds the best possible representation of $\mathbf{y}$ using selected atoms, *i.e.* orthogonal projection. In the other words, the algorithm has these steps: i) selecting an atom, *i.e.*

$$i^* = \text{argmin}_i \min_{s_{k+1}:=s_k\cup\{i\}} \|\mathbf{y} - \mathbf{\Phi}_{s_{k+1}}\mathbf{\Phi}^\dagger_{s_{k+1}}\mathbf{y}\|_2, \quad (1)$$

where s_k is the index set of selected atoms up to the iteration k, $\mathbf{\Phi}_{s_{k+1}}$ is the subdictionary selected by the index set s_k and $\dagger$ represents the More-Penrose pseudoinverse and ii) using the Least Squares (LS) representation within the selected atoms. As the optimisation problem (1) is computationally expensive, Chen *et al.* [5] presented a technique, which iteratively updates the QR factorisation of the selected subdictionary, *i.e.* similar to the fast QR factorisation based OMP implementation [6]. It has an extra step in which we orthogonalise the rest of the dictionary with respect to the selected subdictionary, and then normalise the new dictionary. We refer the readers to [5], for a more detailed description of the OLS algorithm.

If $\mathbf{y}$ is "non-negative sparse", *i.e.* the representation $\mathbf{x}$ has the property $\|\mathbf{x}\|_0 \leq k,\ \mathbf{x} \in \mathbb{R}^n_+$, we intend to incorporate the extra information and adapt OLS to the new setting. This adaptation can be done if we combine the two steps of the OLS and enforce the non-negativity of the coefficients, *i.e.*

$$i^* = \text{argmin}_i \min_{\substack{s_{k+1} := s_k \cup \{i\} \\ \boldsymbol{\alpha} \in \mathbb{R}^{k+1}_+}} \|\mathbf{y} - \mathbf{\Phi}_{s_{k+1}}\boldsymbol{\alpha}\|_2. \quad (2)$$

We call such an implementation Basic NNOLS (BNNOLS). The extension of the fast implementation of OLS for the non-negative representation is not straightforward, if we want to *exactly* solve NNOLS.

Algorithm 1 Suboptimal Non-Negative OLS

initialisation: $s = \emptyset$, $k = 0$, $\mathbf{x} = \mathbf{0}$, $\boldsymbol{\Omega} = \boldsymbol{\Phi}$ and $\mathbf{r}_0 = \mathbf{y}$
while $k < K$ & $\max(\boldsymbol{\Omega}^T \mathbf{r}_k) > 0$ **do**
  $\iota^* \leftarrow \max(\boldsymbol{\Omega}^T \mathbf{r}_k)$
  $s \leftarrow s \cup \iota^*$
  $\mathbf{x}_s \leftarrow \text{argmin}_{\boldsymbol{\theta}>0} \|\mathbf{y} - \boldsymbol{\Phi}_s \boldsymbol{\theta}\|_2$
  $\mathbf{r}_{k+1} \leftarrow \mathbf{y} - \boldsymbol{\Phi}_s \mathbf{x}_s$
  Updating $\boldsymbol{\Omega} = [\boldsymbol{\omega}_i]$, $\boldsymbol{\omega}_i$, $\forall i \in s^c$, from (3)
  $k \leftarrow k + 1$
end while
$\mathbf{x}|_s \leftarrow \mathbf{x}_s$

We here present a simple technique to extend the fast OLS implementation in section 2, without solving an expensive optimisation problem in the loop. We then show that the new implementation is significantly faster than a basic implementation of NNOLS in section 3.

2. NON-NEGATIVE OLS

Let $\boldsymbol{\Phi}_k \in \mathbb{R}^{m \times k}$ be the k selected atoms of $\boldsymbol{\Phi}$ at iteration k, in which the columns are ordered based on the iteration number, *i.e.* the jth column is the jth selected atom. We can now generate $\bar{\boldsymbol{\Phi}}_k \in \mathbb{R}^{m \times (n-k)}$ as the rest of the dictionary, *i.e.* including all non-selected atoms. In each iteration of OLS, we need to calculate a matrix $\boldsymbol{\Omega} = [\boldsymbol{\omega}_i]_{1 \le i \le n-k}$, which is the column-normalised version of the orthogonal part of $\bar{\boldsymbol{\Phi}}_k$ to the column span of $\boldsymbol{\Phi}_k$. In other words,

$$\boldsymbol{\omega}_i := \frac{\bar{\boldsymbol{\phi}}_i^{\perp}}{\|\bar{\boldsymbol{\phi}}_i^{\perp}\|_2}, \tag{3}$$

where $\bar{\boldsymbol{\phi}}_i^{\perp} := (\mathbf{I} - \boldsymbol{\Phi}_k \boldsymbol{\Phi}_k^{\dagger}) \bar{\boldsymbol{\phi}}_i$. We recall that $\bar{\boldsymbol{\phi}}_i$'s with the property $\bar{\boldsymbol{\phi}}_i^{\perp} = \mathbf{0}$, will be eliminated, as they lie in the span of $\boldsymbol{\Phi}_k$ and there is no need to consider them in the following iterations. Let $\mathbf{r}_k = \mathbf{y} - \boldsymbol{\Phi}_k \mathbf{x}_k$ be the residual error of the representation at iteration k, and $\mathbf{x}_k$ is the coefficient vector at this iteration. As $\boldsymbol{\Omega}$ is orthogonal to the span of $\boldsymbol{\Phi}_k$ and normalised, we can find the *direction with the maximum residual energy reduction* by simply finding the maximum of $|\boldsymbol{\Omega}^T \mathbf{r}_k|$, where $|\cdot|$ is the element by element absolute value operator. The optimal direction corresponding to $\boldsymbol{\omega}_{i^*}$ provides us the index of an atom in this iteration. If we now consider the non-negative representation, one *suboptimal* approach is to: i) select an atom with positive contribution, *i.e* $\max_i(\boldsymbol{\omega}_i^T \mathbf{r}_k) > 0$, ii) updating the coefficients using NNLS. A pseudo code for this algorithm, which is called Suboptimal NNOLS (SNNOLS) is presented in Algorithm 1. s^c is the complement set of s. SNNOLS method does not find the solution of iterating (2). The reason is related to the fact that the selection criteria does not consider the fact that representation based on the selected atoms in $\boldsymbol{\Phi}_k$ have to be positive. In the case which we enforce an already selected atom to have a zero coefficient value, we may have some suboptimality. We also need to consider the computational cost of SNNOLS, as it iteratively needs to solve NNLS. The solution which we propose here is to combine the selection and coefficient update steps to avoid such suboptimality and reduce the computational cost by not directly solving NNLS.

Algorithm 2 Fast Non-Negative Orthogonal Least Squares

initialisation: $s = z_0 = \emptyset$, $k = 0$, $\boldsymbol{\Omega} = \boldsymbol{\Phi}$ and $\mathbf{r}_0 = \mathbf{y}$
while $k < K$ & $\max(\boldsymbol{\Omega}^T \mathbf{r}_k) > 0$ **do**
  $(\boldsymbol{\zeta}, \boldsymbol{\iota}) \leftarrow \text{sort}_{\downarrow}(\boldsymbol{\Omega}^T \mathbf{r}_k)$
  $p \leftarrow 1$
  $p^c \leftarrow 1$
  $z^c \leftarrow 0$
  while $\sim$ Terminate & $p < N$ **do**
    z^t from (6)
    $z \leftarrow \boldsymbol{\psi}_{\boldsymbol{\iota}(p)}^T \mathbf{r}_k : \boldsymbol{\psi}_{\boldsymbol{\iota}(p)} = \bar{\boldsymbol{\Phi}}_k^{\perp} / \|\bar{\boldsymbol{\Phi}}_k^{\perp}\|_2,$
    $\bar{\boldsymbol{\Phi}}_k^{\perp} = (\mathbf{I} - \boldsymbol{\Psi}_k \boldsymbol{\Psi}_k^T) \bar{\boldsymbol{\phi}}_{\boldsymbol{\iota}(p)}$
    Update based on Table 1
  end while
  $s \leftarrow s \cup \boldsymbol{\iota}(p)$
  Update $\boldsymbol{\Psi}$, $\bar{\boldsymbol{\Phi}}$, $\boldsymbol{\Omega}$ and $\mathbf{R}^{-1}$
  $\mathbf{z}_{k+1} \leftarrow [\mathbf{z}_k; z_{k+1}]$
  $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - z_{k+1} \boldsymbol{\psi}_{k+1}$
  $k \leftarrow k + 1$
end while
output: $\mathbf{x}|_s \leftarrow \mathbf{R}^{-1} \mathbf{z}_K$

If	Then
$0 < z \le z^t,\ z > z^c$	$z_{k+1} \leftarrow z$, Terminate
$0 < z \le z^t,\ z \le z^c$	$z_{k+1} \leftarrow z^c, p \leftarrow p^c$, Terminate
$z > z^c \ge z^t$	$p \leftarrow p + 1$
$z^c \ge z > z^t$	$z_{k+1} \leftarrow z^c, p \leftarrow p^c$, Terminate
$z > z^t > z^c$	$z^c \leftarrow z^t, p^c \leftarrow p,\ p \leftarrow p + 1$
$z < 0$	Terminate

Table 1. Decision rules to guarantee positivity of the coefficients.

2.1. Fast Non-Negative OLS

If $\boldsymbol{\Psi}_k \in \mathbb{R}^{m \times k}$ is iteratively generated using the selected atoms of $\boldsymbol{\Omega}$, up to the kth iteration, we can QR factorise $\boldsymbol{\Phi}_k$ by $\boldsymbol{\Psi}_k \mathbf{R}_k$, where $\boldsymbol{\Psi}_k$ is orthonormal and $\mathbf{R}_k \in R^{k \times k}$ is an upper-diagonal matrix. The reason for such a structure for $\mathbf{R}$ is the fact that $\boldsymbol{\Phi}_j$ is orthogonal to $\boldsymbol{\psi}_p$'s, for $\forall p : \ j < p$. This QR factorisation is updating throughout the iterations as follows: i) adding $\boldsymbol{\omega}_{i^*}$ as the new column of $\boldsymbol{\Psi}$, *i.e.*

$$\boldsymbol{\Psi}_{k+1} = [\boldsymbol{\Psi}_k \ \boldsymbol{\omega}_{i^*}] \tag{4}$$

and ii) updating $\mathbf{R}_k$ using the following equation,

$$\mathbf{R}_{k+1} = \begin{bmatrix} \mathbf{R}_k & \nu \\ \mathbf{0} & \mu \end{bmatrix}, \tag{5}$$

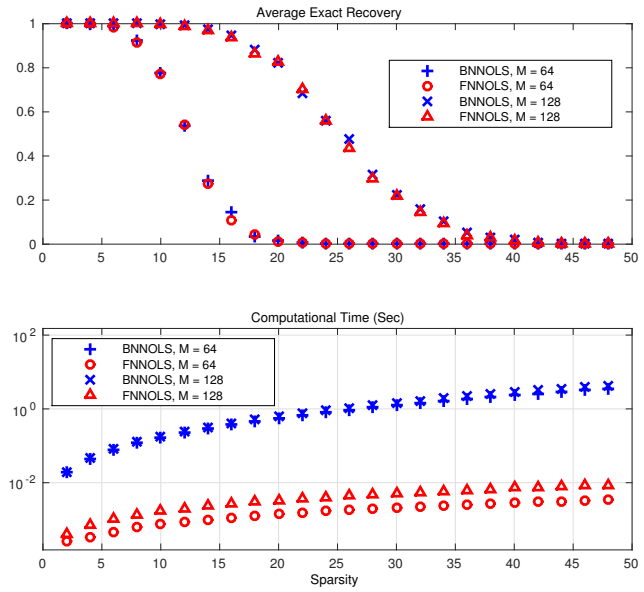

Fig. 1. Exact recovery and computation time. $N = 256$, M is 64 or 128 and sparsity K is changing. The proposed technique FNNOLS is roughly 100 times faster than BNNOLS!

where $\nu = \mathbf{\Psi}_k^T \bar{\boldsymbol{\phi}}_{i^*}$, $\mu = \|\bar{\boldsymbol{\phi}}_{i^*}^{\perp}\|_2$ and $\bar{\boldsymbol{\phi}}_{i^*}^{\perp}$ can be easily calculated using $\bar{\boldsymbol{\phi}}_{i^*}^{\perp} = (\mathbf{I} - \mathbf{\Psi}_k \mathbf{\Psi}_k^T)\bar{\boldsymbol{\phi}}_{i^*}$, *i.e.* no need to calculate the pseudoinverse of $\mathbf{\Phi}_k$, comparing to the definition after (3). This process can be interpreted as a type of Graham-Schmidt (GS) orthogonalisation process, which is a well-known QR factorisation technique [7]. With the new setting, we can rewrite the OLS algorithm as follows: In the first iteration, we do not need any orthogonalisation and we have $\boldsymbol{\phi}_1 = \boldsymbol{\psi}_1$ and $\mathbf{R}_1 = [1]$. In the kth iteration, $k \geq 1$, let the best approximation of $\mathbf{y}$, using $\mathbf{\Phi}_k$, be $\sum_{i=1}^{k} x_i \boldsymbol{\phi}_i = \sum_{i=1}^{k} z_i \boldsymbol{\psi}_i$. In the $k+1$ iteration, we have,

$$\begin{aligned}\sum_{i=1}^{k+1} z_i \boldsymbol{\psi}_i &= \sum_{i=1}^{k} z_i \boldsymbol{\psi}_i + z_{k+1}\boldsymbol{\psi}_{k+1} \\ &= \sum_{i=1}^{k} x_i \boldsymbol{\phi}_i + z_{k+1}\boldsymbol{\psi}_{k+1}.\end{aligned}$$

We recall that $\boldsymbol{\psi}_i$'s are orthogonal and choosing a new elementary function $\boldsymbol{\psi}_{k+1}$ does not change the optimal value of the best representation z_i, $\forall i : i < k+1$, calculated up to the previous iteration k. This is obviously not true for x_i's as $\mathbf{\Phi}_{k+1}$ is not orthogonal.

As $\boldsymbol{\psi}_{k+1}$ lives in the span of the non-redundant set $\{\boldsymbol{\phi}_j\}_{j\in[1,k+1]}$, $\boldsymbol{\psi}_{k+1} = \sum_{j=1}^{k+1} \gamma_j \boldsymbol{\phi}_j$ for some unique γ_j's.

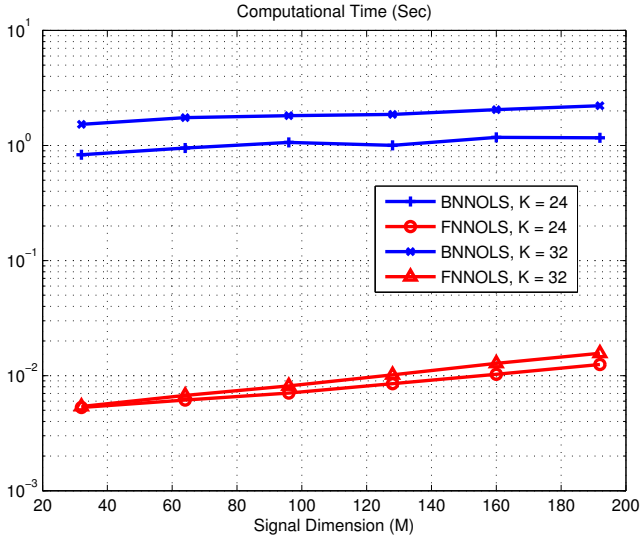

Fig. 2. Computation time for the fixed $N = 256$ and $K = 24$ and 32.

We can then have,

$$\begin{aligned}\sum_{i=1}^{k+1} z_i \boldsymbol{\psi}_i &= \sum_{i=1}^{k} x_i \boldsymbol{\phi}_i + \sum_{j=1}^{k+1} z_{k+1}\gamma_j \boldsymbol{\phi}_j \\ &= \sum_{i=1}^{k} (x_i + z_{k+1}\gamma_i)\boldsymbol{\phi}_i + z_{k+1}\gamma_{k+1}\boldsymbol{\phi}_{k+1}.\end{aligned}$$

As $z_{k+1}\gamma_{k+1}$ is always positive, we only need to assure that $x_i + z_{k+1}\gamma_i \geq 0$ or

$$z_{k+1} \leq z^t := \begin{cases} \min_{i,\gamma_i<0} \left|\frac{x_i}{\gamma_i}\right| & \exists i, \gamma_i < 0 \\ +\infty & \text{Otherwise}. \end{cases} \quad (6)$$

In OLS, we only need to keep z_i's in the memory and calculate x_i's at the end of algorithm [8]. To assure that x_i's are all non-negative, z_i's should comply the condition of (6). We then choose the atom that the corresponding z_{k+1}, or shrunk by the upper-bound of (6), has the largest value. We therefore need to track the record of the best possible value for z, if $z = \boldsymbol{\omega}_{i^*}^T \mathbf{r}_k$ does not comply with (6). If $(\boldsymbol{\zeta}, \boldsymbol{\iota}) = \text{sort}_{\downarrow}(\mathbf{\Omega}^T \mathbf{r}_k)$, where $\text{sort}_{\downarrow}$ is the sorting operator in a descent order, $\boldsymbol{\zeta}$ is the sorted coefficients with corresponding indices $\boldsymbol{\iota}$. A pseudocode of the fast non-negative orthogonal least square is presented in Algorithm 2. In this algorithm, z is the current candidate, starting with $z = \boldsymbol{\zeta}(p)$, $p = 1$, in an internal loop in the kth iteration, we make the decision based on the rules of Table 1, to guarantee that the coefficients x_i's will remain non-negative throughout iterations. We also need to calculate $\mathbf{R}^{-1}$ at the last iteration, which would be expensive. We thus can iteratively generate $\mathbf{R}^{-1}$ using the following update rule,

$$\mathbf{R}_{k+1}^{-1} = \begin{bmatrix} \mathbf{R}_k^{-1} & -\frac{\mathbf{R}_k^{-1}\nu}{\mu} \\ \mathbf{0} & \frac{1}{\mu} \end{bmatrix}. \quad (7)$$

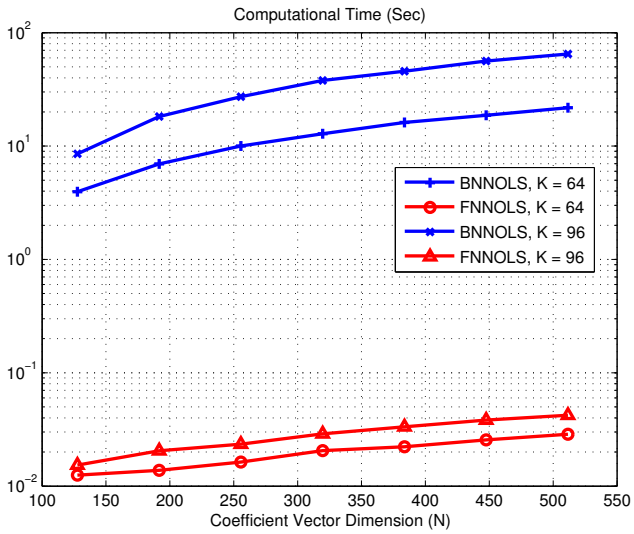

Fig. 3. Computation time for the fixed $M = 128$ and $K = 64$ and 96.

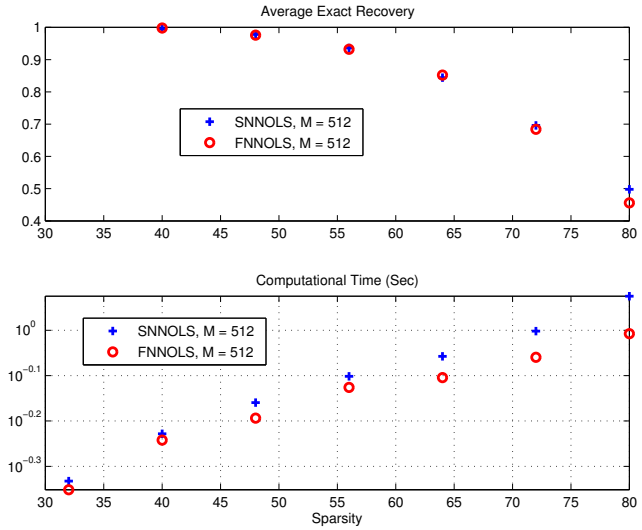

Fig. 4. Exact recovery and computation time. Comparison between SNNOLS and FNNOLS. $N = 1024$, $M = 512$ sparsity K is changing.

where ν and μ were defined after (5). For the fast implementation of this algorithm, we have to also efficiently calculate $\boldsymbol{\gamma}$. It is not difficult to show that $\boldsymbol{\gamma}$ is the last column of $\mathbf{R}_{k+1}^{-1}$, *i.e.*, $\boldsymbol{\gamma} = [-\frac{\mathbf{R}_k^{-1}\nu}{\mu}; \frac{1}{\mu}]$, which is kept in the memory.

3. SIMULATIONS

We investigate the runtime of the algorithms on a single core of an Intel core 2.6 GHz processor. In the first experiment, we randomly generated $\mathbf{\Phi}$ with 256 atoms and 64 or 128 rows using *i.i.d.* Gaussian random variables, followed by column normalisation. $\mathbf{y}$ was generated using a Gaussian-Bernoulli model, *i.e.* uniformly random support and Normal distribution for the non-zero coefficients. We repeated the simulations 100 times, using proposed FNNOLS and BNNOLS and different sparsity levels. The correct recovery of the support and computational time are shown in Figure 1. While the exact recoveries are very similar, proposed method is significantly faster. The difference is more obvious for large k's, where we can get a computation gain of 100.

In the second experiment, we used the same method to generate the dictionary, but we fixed $N = 256$, to investigate the computation time as a function of M. Here, $K = 24$ or 32 and M is between 32 and 196. The computational time is plotted in Figure 2. The computational time is slowly increasing with M. We can observe that the difference in computations is not significant for different K's using the fast technique, while it is noticeable using the basic implementation.

We then fixed $M = 128$ and changed N from 128 to 512 and similarly repeated each simulation 100 times. The sparsity was set to be 64 or 96, and the results are shown in Figure 3. The computational cost is slowly increasing *w.r.t.* the parameter N in FNNOLS, while it is increasing faster for the BNNOLS. However, the difference between the computational time of the fast and basic implementations are considerable, where we yield a gain of 1000 times acceleration, using the FNNOLS algorithm!

We also presented an approximate NNOLS technique, called the SNNOLS, which can be seen as an intuitive extension of the fast OLS implementation to the non-negative domain. As we can see, the computational time of SNNOLS is also low, which means that we can compare it with FNNOLS in a larger dimension setting. We then chose $M = 512$, $N = 1024$ and changed the sparsity level K from 32 to 80 and averaged the simulations, with the random settings which explained before, over 500 trials. We have plotted the exact support recovery (top panel) and the algorithms run-time (bottom panel) in Figure 4. While the exact recoveries are very similar, the computational times are slightly different. For the example, the computational time of SNNOLS is 25% higher than FNNOLS for $K = 80$. The similarity between the exact recovery of two algorithms is an interesting fact, which needs more investigation.

4. CONCLUSION

A significantly faster implementation for the non-negative orthogonal least square algorithm was proposed here. The new implementation is based on the recursive generation of a QR factorisation of the dictionary and it allows us to use FNNOLS in a broader application. We also presented an algorithm which approximately solve the OLS problem in a non-negative setting, which is called SNNOLS. This algorithm performs similarly in the exact recovery of the sparsity support, when the dictionaries are randomly generated, but it is slightly slower than FNNOLS. The performance of the proposed techniques, *i.e.* FNNOLS and SNNOLS, have to

be investigated, using other types of dictionaries. The computational complexity analysis of the FNNOLS is also an interesting subject, which we left for the future work.

Acknowledgment

The authors acknowledge the support of EPSRC and the MOD University Defence Research Collaboration in Signal Processing, through the grant number EP/K014277/1.

REFERENCES

[1] Y Qian, S Jia, J Zhou, and A Robles-Kelly, "Hyperspectral unmixing via sparsity-constrained nonnegative matrix factorization," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 49, no. 11, pp. 4282–4297, 2011.

[2] F. Parvaresh, H. Vikalo, S. Misra, and B. Hassibi, "Recovering sparse signals using sparse measurement matrices in compressed DNA microarrays," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 3, pp. 275–285, 2008.

[3] PO Hoyer, "Non-negative sparse coding," in *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*. 2002, pp. 557–565, IEEE.

[4] M. Yaghoobi, D. Wu, and M.E. Davies, "Fast non-negative orthogonal matching pursuit," *IEEE Signal Processing Letters*, vol. 22, no. 9, 2015.

[5] S. Chen, S.A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *International Journal of Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[6] B.K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM Journal of Comput*, vol. 24, no. 2, pp. 227–234, 1995.

[7] G.H. Golub and C.F. Van Loan, *Matrix computations*, Johns Hopkins University Press Baltimore, 1996.

[8] T Blumensath and ME Davies, "On the difference between orthogonal matching pursuit and orthogonal least squares," Tech. Rep., 2007.