IN GREEDY PURSUIT OF NEW DIRECTIONS: (NEARLY) ORTHOGONAL MATCHING PURSUIT BY DIRECTIONAL OPTIMISATION

Thomas Blumensath, Mike E. Davies

IDCOM & Joint Research Institute for Signal and Image Processing
The University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh, EH9 3JL, UK

ABSTRACT

Matching Pursuit and orthogonal Matching Pursuit are greedy algorithms used to obtain sparse signal approximations. Orthogonal Matching Pursuit is known to offer better performance, but Matching Pursuit allows more efficient implementations. In this paper we propose novel greedy Pursuit algorithms based on directional updates. Using a conjugate direction, the algorithm becomes a novel implementation of orthogonal Matching Pursuit, with computational requirements similar to current implementations based on QR factorisation. A significant reduction in memory requirements and computational complexity can be achieved by approximating the conjugate direction. Further computational savings can be made by using a steepest descent direction. The two resulting algorithms are then comparable to Matching Pursuit in their computational requirements, their performance is however shown to be closer to that of orthogonal Matching Pursuit with the (slightly slower) approximate conjugate direction based approach outperforming the gradient descent method.

1. INTRODUCTION

In this paper we are interested in modelling a vector $\mathbf{x}$ using the linear model:

$$\mathbf{x} = \Phi\mathbf{y} + \varepsilon, \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^N$ and $\mathbf{x} \in \mathbb{R}^M$ with $N > M$. If we allow for a non-zero error ε we talk about a signal *approximation*, while for zero ε we have an exact signal *representation*.

Given an *overcomplete* full rank matrix Φ (commonly called the dictionary) and a signal $\mathbf{x}$, the problem is to find an optimally sparse expansion $\mathbf{y}$, i.e an expansion with the largest number of zero (or very small) elements. This problem is known to be NP-hard [1] and different sub-optimal strategies have to be used. These are generally based on convex relaxation of the problem, non-convex (often gradient based) local optimisation or greedy search strategies. Greedy methods include the Matching Pursuit [2] and orthogonal Matching Pursuit [3] algorithms.

Matching Pursuit is one of the fastest strategies and very efficient $O(N\log N)$ implementations for each iteration are possible [4], [5] if Φ is the union of dictionaries that allow fast transforms. For general dictionaries the method is $O(NM)$ per iteration. On the other hand, orthogonal Matching Pursuit has superior performance. Different fast implementations are now available, however, these do not achieve the fast $O(N\log N)$ complexity of Matching Pursuit. More importantly these methods also require additional storage, which for large scale problems can become a limiting factor.

In this paper we develop an implementation of orthogonal Matching Pursuit based on conjugate direction updates. This method requires the same computational resources than other fast implementations of orthogonal Matching Pursuit. We therefore propose an approximation of the conjugate direction, which can be computed more efficiently and which requires significantly less storage. A further simplification is then introduced, which uses the gradient direction. This approach offers additional computational savings.

We start the development with a review of Matching Pursuit and orthogonal Matching Pursuit in section 3. State of the art implementations of orthogonal Matching Pursuit are then reviewed in section 4 before we develop the conjugate direction based algorithm in section 5. The approximate implementation of orthogonal Matching Pursuit are developed in 6 while some computational performance studies are presented in 7.

2. NOTATION

In this paper we use the following conventions. The i^{th} column of Φ will be denoted by ϕ_i. The set Γ^n will be a set containing n indices. The matrix Φ_{Γ^n} will be a sub-matrix of Φ containing only those columns of Φ with indices in Γ^n. We use the same convention for vectors, e.g. $\mathbf{y}_{\Gamma^n}$ is a sub-vector of $\mathbf{y}$ containing only those elements of $\mathbf{y}$ with indices in Γ^n. In general, the superscript in the subscript of $\mathbf{y}_{\Gamma^n}$ reminds us that we are in iteration n, on occasion, however, we use additional superscripts, e.g. $\mathbf{y}^{n-1}_{\Gamma^n}$ now refers to an n-dimensional vector as calculated in iteration $n-1$. In general, in iteration $n-1$ we only update elements with indices in Γ^{n-1}, therefore, the element in $\mathbf{y}^{n-1}_{\Gamma^n}$ which is not in $\mathbf{y}^{n-1}_{\Gamma^{n-1}}$ will typically be zero. We also make heavy use of the Gram matrix $\mathbf{G}_{\Gamma^n} = \Phi^T_{\Gamma^n}\Phi_{\Gamma^n}$.

3. FROM MATCHING PURSUIT TO ORTHOGONAL MATCHING PURSUIT

Matching Pursuit [2] is a greedy algorithm that iteratively selects elements from Φ that have the largest (in absolute magnitude) inner product with the current approximation error, i.e the algorithm selects that atom that on its own leads to the largest reduction in the approximation error. In the first step, the approximation error $\mathbf{r}^0$ is the signal $\mathbf{x}$ itself. After an element is selected, the coefficients $\mathbf{y}$ are updated by adding

This research was supported by EPSRC grant D000246/1. MED acknowledges support of his position from the Scottish Funding Council and their support of the Joint Research Institute with the Heriot-Watt University as a component part of the Edinburgh Research Partnership. Some of the fundamental ideas underlying the work presented here were inspired by a discussion the second author had with Rémi Gribonval, whom we like to thank for this stimulating input.

the value of the inner product to the appropriate coefficient y_i. The error is then reduced accordingly. More formally we write this algorithm as:

1. Initialise $\mathbf{r}^0 = \mathbf{x}, \mathbf{y}^0 = 0$
2. for $n = 1; n := n+1$ till stopping criterion is met
   (a) $\nabla^n = \mathbf{\Phi}^T \mathbf{r}^n$
   (b) $i^n = \arg_i \max |\nabla_i|$
   (c) $y^n_{i^n} = y^{n-1}_{i^n} + \nabla_{i^n}$
   (d) $\mathbf{r}^n = \mathbf{r}^{n-1} - \phi_{i^n} \nabla_{i^n}$
3. Output: $\mathbf{r}^n, \mathbf{y}^n$

The computational bottle neck of this algorithm is the required matrix multiplication $\mathbf{\Phi}^T \mathbf{r}^n$, however, if the error $\mathbf{r}^n$ only changes locally, more efficient updates are possible [2]. Furthermore, if the dictionary $\mathbf{\Phi}$ is the union of orthogonal dictionaries for which fast transforms are available, then this multiplication can be implemented efficiently [4]. In this case, the search for the maximum correlation in step 2.b) is often more costly and efficient search strategies should be used [4].

Orthogonal Matching Pursuit [3], [6] differs from Matching Pursuit in that in each iteration $\mathbf{y}$ is calculated by projecting the signal $\mathbf{x}$ orthogonally onto all selected atoms. Orthogonal Matching Pursuit therefore calculates the best signal approximation possible with these atoms. This benefit comes at the cost of the orthogonal projection. The orthogonal Matching Pursuit algorithm is:

1. Initialise $\mathbf{r}^0 = \mathbf{x}, \mathbf{y}^0 = 0, \Gamma^0 = \oslash$
2. for $n = 1; n := n+1$ till stopping criterion is met
   (a) $\nabla^n = \mathbf{\Phi}^T \mathbf{r}^n$
   (b) $i^n = \arg_i \max |\nabla_i|$
   (c) $\Gamma^n = \Gamma^{n-1} \cup i^n$
   (d) $\mathbf{y}^n = \mathbf{\Phi}^\dagger_{\Gamma^n} \mathbf{x}$
   (e) $\mathbf{r}^n = \mathbf{x} - \mathbf{\Phi} \mathbf{y}^n$
3. Output: $\mathbf{r}^n, \mathbf{y}^n$

Here the dagger † indicates the Moore-Penrose pseudo-inverse.

4. TRADITIONAL IMPLEMENTATIONS OF ORTHOGONAL MATCHING PURSUIT

Calculating the pseudo-inverse in each iteration of orthogonal Matching Pursuit is costly and faster strategies are available. Here we mention two approaches, one based on QR factorisation and one, based on iterative updates of the required matrix inverse.

4.1 Orthogonal Matching Pursuit by QR Factorisation

Currently, the most efficient implementation of orthogonal Matching Pursuit is based on QR factorisation. In this implementation the sub-dictionary $\mathbf{\Phi}_{\Gamma^n}$ is factorised as follows:

$$\mathbf{\Phi}_{\Gamma^n} = \mathbf{Q}_{\Gamma^n} \mathbf{R}_{\Gamma^n}, \tag{2}$$

where $\mathbf{Q}_{\Gamma^n} \in \mathbb{R}^{M \times n}$ is a unitary matrix, i.e. $\mathbf{Q}^T_{\Gamma^n} \mathbf{Q}_{\Gamma^n}$ is the identity matrix and $\mathbf{R}_{\Gamma^n}$ is upper-triangular. With this factorisation we can write the n-term approximations $\hat{\mathbf{x}}_{\Gamma^n}$ as:

$$\hat{\mathbf{x}}_{\Gamma^n} = \mathbf{\Phi}_{\Gamma^n} \mathbf{y}_{\Gamma^n} = \mathbf{Q}_{\Gamma^n} \mathbf{R}_{\Gamma^n} \mathbf{y}_{\Gamma^n}. \tag{3}$$

If we let $\mathbf{z}_{\Gamma^n} = \mathbf{R}_{\Gamma^n} \mathbf{y}_{\Gamma^n}$, then a very efficient implementation of orthogonal Matching Pursuit does not need to calculate $\mathbf{y}_{\Gamma^n}$ in each iteration, i.e. the algorithm becomes:

1. Initialise $\mathbf{r}^0 = \mathbf{x}, \mathbf{z}^0 = 0, \Gamma^0 = \oslash$
2. for $n = 1; n := n+1$ till stopping criterion is met
   (a) $\nabla^n = \mathbf{\Phi}^T \mathbf{r}^n$
   (b) $i^n = \arg_i \max |\nabla_i|$
   (c) $\Gamma^n = \Gamma^{n-1} \cup i^n$
   (d) Update $\mathbf{Q}_{\Gamma^n}$ and $\mathbf{R}_{\Gamma^n}$ such that $\mathbf{Q}_{\Gamma^n} \mathbf{R}_{\Gamma^n} = \mathbf{\Phi}_{\Gamma^n}$, $\mathbf{Q}^T_{\Gamma^n} \mathbf{Q}_{\Gamma^n} = \mathbf{I}$ and $\mathbf{R}_{\Gamma^n}$ is upper triangular.
   (e) $\mathbf{z}_{\Gamma^n} = [\mathbf{z}_{\Gamma^{n-1}}; z^n]$
   (f) $\mathbf{r}^n = \mathbf{r}^{n-1} - z^n \mathbf{q}$
3. $\mathbf{y}_{\Gamma^n} = \mathbf{R}^{-1}_{\Gamma^n} \mathbf{z}_{\Gamma^n}$
4. Output: $\mathbf{r}^n, \mathbf{y}^n$

where $z^n = \mathbf{q}^T \mathbf{x}$. $\mathbf{q}$ is defined below. There are several important computational tricks to be used when implementing the above algorithm. The QR factorisation in step 2.d) can be solved iteratively using a variety of methods and normally a form of the Modified Gram-Schmidt procedure is used[1]. In iteration n both $\mathbf{Q}_{\Gamma^n}$ and $\mathbf{R}_{\Gamma^n}$ are updated using:

$$\mathbf{R}_{\Gamma^n} = \begin{bmatrix} \mathbf{R}_{\Gamma^{n-1}} & \mathbf{Q}^T_{\Gamma^n - 1} \phi_{i^n} \\ \mathbf{0} & \|\mathbf{q}\|_2 \end{bmatrix}; \tag{4}$$

and

$$\mathbf{Q}_{\Gamma^n} = [\mathbf{Q}_{\Gamma^n - 1}; \mathbf{q}], \tag{5}$$

where $\mathbf{q}$ is such that: $\mathbf{Q}^T_{\Gamma^n - 1} \mathbf{q} = 0, span\{\mathbf{Q}^T_{\Gamma^{n-1}}, \mathbf{q}\} = span\{\Phi_{\Gamma^n}\}$ and $\|\mathbf{q}\|_2 = 1$. Finally, the calculation of $\mathbf{y}_{\Gamma^n}$ in 3) can be done efficiently using back-substitution.

4.2 Orthogonal Matching Pursuit using Directional Updates

Orthogonal Matching Pursuit can also be implemented using a form of directional update. In each iteration, we calculate a direction $\mathbf{d}^n_{\Gamma^n}$ and a step-size a^n and update :

$$\mathbf{y}^n_{\Gamma^n} = \mathbf{y}^{n-1}_{\Gamma^n} + a^n \mathbf{d}^n_{\Gamma^n}. \tag{6}$$

It was shown by Pati et al. in [3] that we can solve the orthogonal Matching Pursuit problem by using the direction:

$$\mathbf{d}^n_{\Gamma^n} = \begin{bmatrix} -\mathbf{G}^{-1}_{\Gamma^{n-1}} \mathbf{\Phi}_{\Gamma^{n-1}} \phi_{i^n} \\ 1 \end{bmatrix}; \tag{7}$$

where $\mathbf{G}^{n-1}_\Gamma = \mathbf{\Phi}^T_{\Gamma^{n-1}} \mathbf{\Phi}_{\Gamma^{n-1}}$. The step size is then $a^n = \nabla_{i^n} / (1 - \phi^T_{i^n} \mathbf{\Phi}^T_{\Gamma^{n-1}} \mathbf{G}^{-1}_{\Gamma^{n-1}} \mathbf{\Phi}_{\Gamma^{n-1}} \phi_{i^n})$. Note that the matrix inverse can be updated efficiently in each iteration using the block matrix inversion formula [3].

1. Initialise $\mathbf{r}^0 = \mathbf{x}, \mathbf{y}^0 = 0, \Gamma^0 = \oslash$
2. for $n = 1; n := n+1$ till stopping criterion is met
   (a) $\nabla^n = \mathbf{\Phi}^T \mathbf{r}^n$
   (b) $i^n = \arg_i \max |\nabla_i|$
   (c) $\Gamma^n = \Gamma^{n-1} \cup i^n$
   (d) update $\mathbf{G}^{-1}_{\Gamma^n}$
   (e) calculate $\mathbf{d}_{\Gamma^n}$
   (f) calculate a^n
   (g) $\mathbf{y}^n_{\Gamma^n} = \mathbf{y}^{n-1}_{\Gamma^n} + a^n \mathbf{d}^n_{\Gamma^n}$
   (h) $\mathbf{r}^n = \mathbf{r}^{n-1} - a^n \mathbf{\Phi} \mathbf{d}^n_{\Gamma^n}$
3. Output: $\mathbf{r}^n, \mathbf{y}^n$

[1] The normal Modified Gram-Schmidt method orthogonalises all selected elements and at the same time ensures that a copy of the not selected elements is also orthogonalised to all selected elements. This can be costly. A faster implementation orthogonalises new elements only once they are selected.

5. CONJUGATE GRADIENT IMPLEMENTATION OF ORTHOGONAL MATCHING PURSUIT

Instead of calculating the update direction as in equation (7), we here derive a different method based on the idea of conjugate directions.

The conjugate direction method [7, Section 10.2], [8, Section 16.4] is similar to the gradient descent algorithm, however, it is guaranteed to solve quadratic optimisation problems in N steps, where N is the dimension of the problem. In general, let us assume we want to minimise a quadratic function, which can be written as $\frac{1}{2}\mathbf{y}^T\mathbf{G}\mathbf{y} - \mathbf{b}^T\mathbf{y}$. This is equivalently to solving $\mathbf{G}\mathbf{y} = \mathbf{b}$ over $\mathbf{y}$. The conjugate direction method calculates a new update direction such that the new direction is $\mathbf{G}$-conjugate to the previously chosen directions. This means that in the n^{th} iteration the conjugate direction $\mathbf{d}^n$ satisfies:

$$\mathbf{d}^{n^T}\mathbf{G}\mathbf{d}^k = 0 \tag{8}$$

for all $k \neq n$. See [7, Section 10.2] and [8, Section 16.4] for further details.

In iteration n of the orthogonal Matching Pursuit algorithm we need to minimise the quadratic cost-function in n unknowns:

$$\|\mathbf{x} - \mathbf{\Phi}_{\Gamma^n}\mathbf{y}_{\Gamma^n}\|_2^2. \tag{9}$$

The dimension n of the cost function in equation (9) changes in each iteration. Let us use the notation $\mathbf{d}^k_{\Gamma^n}$ to denote the k^{th} conjugate direction, where the subscript reminds us that this vector is n dimensional. In the Γ^n notation, $\mathbf{y}_{\Gamma^n}$ is an n-dimensional sub-vector of $\mathbf{y}$, which is updated in direction $\mathbf{d}^n_{\Gamma^n}$. This is equivalent to updating $\mathbf{y}$ using a directional vector $\mathbf{d}^n$ of dimension N, with all elements zero apart from the element indexed by Γ^n. Similarly, the previous update directions $\mathbf{d}^k_{\Gamma^k}$ can be thought of as higher dimensional update directions $\mathbf{d}^k_{\Gamma^n}$ that are only non-zero at the indexes associated with Γ^k.

With this notation we can state the conjugate direction Theorem [8, Theorem 16.2] as:

Theorem 1 *[8, Theorem 16.2] Let $[\mathbf{d}^1_{\Gamma^n}, \mathbf{d}^2_{\Gamma^n}, \ldots, \mathbf{d}^n_{\Gamma^n}]$ be any set of non-zero $\mathbf{G}_{\Gamma^n}$-conjugate vectors, then the solution to the problem $\mathbf{G}_{\Gamma^n}\mathbf{y}_{\Gamma^n} = \mathbf{\Phi}^T_{\Gamma^n}\mathbf{x}$ is*

$$\mathbf{y}^n_{\Gamma^n} = \sum_{k=1}^{n} a^k \mathbf{d}^k_{\Gamma^n}, \tag{10}$$

with a step-size of:

$$a^k = \frac{\mathbf{r}^{k^T}\mathbf{\Phi}_{\Gamma^k}\mathbf{d}^k_{\Gamma^n}}{\mathbf{d}^{k^T}_{\Gamma^n}\mathbf{G}_{\Gamma^k}\mathbf{d}^k_{\Gamma^n}}, \tag{11}$$

where $\mathbf{r}^k = \mathbf{x} - \mathbf{\Phi}_{\Gamma^{k-1}}\left(\sum_{i=1}^{k-1} a^i \mathbf{d}^i_{\Gamma^n}\right)$.

Note that the step-size a^k only depends on the first $k-1$ step-sizes and the first $k-1$ conjugate-directions so that $\mathbf{y}^n_{\Gamma^n}$ can be approximated iteratively by calculating a new direction and step-size in each iteration.

In the first iteration of orthogonal Matching Pursuit we are dealing with a one dimensional problem and we have a single trivial direction and step-size. In the n^{th} iteration, we select a further atom from the dictionary and increase the dimension by one. If the $n-1$ previous update directions are $\mathbf{G}_{\Gamma^n}$ conjugate, then, by the above theorem, we only require one more conjugate direction and step-size to exactly solve the n dimensional problem.

To show that all the previous $\mathbf{G}_{\Gamma^{n-1}}$-conjugate directions are also $\mathbf{G}_{\Gamma^n}$-conjugate (note the different subscripts), we define the matrix $\mathbf{D}^{n-1}_{\Gamma^{n-1}}$ as the matrix containing all conjugate update directions from iteration $n-1$ and the matrix $\mathbf{D}^{n-1}_{\Gamma^n}$ to be the same matrix but with an additional row of zeros at the bottom. Then in the definition of $\mathbf{G}_{\Gamma^n}$-conjugacy, i.e. $\mathbf{D}^{n-1^T}_{\Gamma^n}\mathbf{G}_{\Gamma^n}\mathbf{D}^{n-1}_{\Gamma^n}$ the last row and column of $\mathbf{G}_{\Gamma^n}$ are multiplied by zeros, which implies that $\mathbf{D}^{n-1^T}_{\Gamma^{n-1}}\mathbf{G}_{\Gamma^{n-1}}\mathbf{D}^{n-1}_{\Gamma^{n-1}} = \mathbf{D}^{n-1^T}_{\Gamma^n}\mathbf{G}_{\Gamma^n}\mathbf{D}^{n-1}_{\Gamma^n}$.

It therefore remains to calculate a single conjugate direction in each iteration. Due to $\mathbf{G}_{\Gamma^n}$-conjugacy in the n^{th} iteration, the new direction has to satisfy the following constraint:

$$\mathbf{D}^{n-1^T}_{\Gamma^n}\mathbf{G}_{\Gamma^n}\mathbf{d}^n_{\Gamma^n} = \mathbf{0} \tag{12}$$

In step $n-1$, assume $\mathbf{y}^n$ to be the projection onto the elements of $\Phi_{\Gamma^{n-1}}$, then each new direction can be expressed as a combination of all previously chosen directions and the current gradient ∇_{Γ^n} [7, Section 10.2], i.e. we express $\mathbf{d}^n_{\Gamma^n}$ as:

$$\mathbf{d}^n_{\Gamma^n} = b_0\nabla_{\Gamma^n} + \mathbf{D}^{n-1}_{\Gamma^n}\mathbf{b}. \tag{13}$$

Without loss of generality we can set $b_0 = 1$. Pre-multiplying by $\mathbf{D}^{n-1}_{\Gamma^n}\mathbf{G}_{\Gamma^n}$ and using the $\mathbf{G}_{\Gamma^n}$-conjugacy then leads to the $n-1$ constraints:

$$\mathbf{D}^{n-1^T}_{\Gamma^n}\mathbf{G}_{\Gamma^n}(\nabla_{\Gamma^n} + \mathbf{D}^{n-1}_{\Gamma^n}\mathbf{b}) = \mathbf{0} \tag{14}$$

from which we can write

$$\mathbf{b} = -(\mathbf{D}^{n-1^T}_{\Gamma^n}\mathbf{G}_{\Gamma^n}\mathbf{D}^{n-1}_{\Gamma^n})^{-1}(\mathbf{D}^{n-1^T}_{\Gamma^n}\mathbf{G}_{\Gamma^n}\nabla_{\Gamma^k}). \tag{15}$$

Again using $\mathbf{G}_{\Gamma^n}$-conjugacy we find that $\mathbf{D}^{n-1}_{\Gamma^n}\mathbf{G}_{\Gamma^n}\mathbf{D}^{n-1}_{\Gamma^n}$ is diagonal so that the conjugate direction can be calculated without matrix inversion.

Note that in the standard conjugate gradient algorithm [7, Section 10.2], [8, Section 16.4] each new update direction can be calculated as a combination of the current gradient and the previous update direction alone, i.e. in the standard conjugate gradient algorithm, all but the last conjugate update direction turn out to be $\mathbf{G}$ conjugate to the current gradient such that $\mathbf{b}$ in equation (15) has only a single non-zero element. Unfortunately, in the context of orthogonal Matching Pursuit, the changing dimensionality destroys this property so that we have to take account of all previous update directions in each step.

We call this algorithm Conjugate Gradient Pursuit (CGP), which can be summarised as:

1. Initialise: $\mathbf{r}^0 = \mathbf{x}, \mathbf{y}^0 = 0, \Gamma^0 = \oslash$
2. for $n = 1; n := n+1$ till stopping criterion is met
   (a) $\nabla^n = \mathbf{\Phi}^T\mathbf{r}^n$
   (b) $i^n = \arg_i \max|\nabla_i|$
   (c) $\Gamma^n = \Gamma^{n-1} \cup i^n$
   (d) if $n = 1$
   - $\mathbf{d}^1_{\Gamma^1} = 1$
   (e) if $n \neq 1$

- $\mathbf{b} = -(\mathbf{D}_{\Gamma^n}^{n-1^T}\mathbf{G}_{\Gamma^n}\mathbf{D}_{\Gamma^n}^{n-1})^{-1}(\mathbf{D}_{\Gamma^n}^{n-1^T}\mathbf{G}_{\Gamma^n}\nabla_{\Gamma^k})$
- $\mathbf{d}_{\Gamma^n}^n = \nabla_{\Gamma^n} - \mathbf{D}\mathbf{b}$

(f) $\mathbf{D}_{\Gamma^n}^n = [\mathbf{D}_{\Gamma^n}^{n-1}; \mathbf{d}_{\Gamma^n}^n]$
(g) $\mathbf{c}^n = \Phi_{\Gamma^n}\mathbf{d}_{\Gamma^n}^n$
(h) $a^n = \mathbf{r}^{n^T}\mathbf{c}^n/(\mathbf{c}^{n^T}\mathbf{c}^n)$
(i) $\mathbf{y}_{\Gamma^n}^n = \mathbf{y}_{\Gamma^n}^{n-1} + a^n\mathbf{d}_{\Gamma^n}^n$
(j) $\mathbf{r}^n = \mathbf{r}^{n-1} - a^n\mathbf{c}^n$

3. Output: $\mathbf{r}^n, \mathbf{y}^n$

Note that in the calculation of $\mathbf{b}$ the product $\mathbf{D}_{\Gamma^n}^{n-1}\mathbf{G}_{\Gamma^n}$ can be updated recursively by adding a single new row and column in each iteration. Note also that $(\mathbf{D}_{\Gamma^{n-1}}^{n-2}\mathbf{G}_{\Gamma^{n-1}}\mathbf{D}_{\Gamma^{n-1}}^{n-2})^{-1}$ and $(\mathbf{D}_{\Gamma^n}^{n-1}\mathbf{G}_{\Gamma^n}\mathbf{D}_{\Gamma^n}^{n-1})^{-1}$ are equal apart from a single additional value added in each iteration. This value is $(\mathbf{c}^{n-1^T}\mathbf{c}^{n-1})$ used in step 2.h) of iteration $n-1$ and does not have to be recalculated.

6. APPROXIMATING ORTHOGONAL MATCHING PURSUIT USING DIRECTIONAL UPDATES

We have remarked that all previous implementations of orthogonal Matching Pursuit as well as the proposed conjugate direction method require the storage of different matrices. For several problems, this storage requirement is undesirable. In this section, we therefore propose two directional update methods, similar to the conjugate direction method, which do not require the storage of any matrices. The price we pay for the reduced memory requirement is that we do not solve orthogonal Matching Pursuit exactly. The two directional optimisation strategies proposed below can however easily be used several times in each Matching Pursuit iteration to get better approximations to orthogonal Matching Pursuit. This naturally comes at the additional cost of multiple calculations of the required update directions and step-sizes.

6.1 Using an Approximated Conjugate Direction

As stressed above, the changing dimensionality of the problem makes it necessary to take account of all previous update directions in the conjugate direction algorithm. If we cannot store all previous update directions we can nevertheless calculate an update direction that is $\mathbf{G}_{\Gamma^n}$-conjugate to only a limited number of previous update directions. We will call this approach Approximate Conjugate Gradient Pursuit (ACGP). Let us here derive this approximate conjugate direction method by only considering a single previous update direction.

In iteration n of the algorithm we require the new update direction to be a linear combination of the previous update direction and the current gradient direction, i.e. we calculate the new direction as:

$$\mathbf{d}_{\Gamma^n}^n = b_0\nabla_{\Gamma^n} + \mathbf{d}_{\Gamma^n}^{n-1}b_1. \tag{16}$$

We again set $b_0 = 1$. To calculate b_1 we then enforce $\mathbf{G}_{\Gamma^n}$-conjugacy with the previous update direction, which leads to the constraint:

$$\mathbf{d}_{\Gamma^n}^{n-1^T}\mathbf{G}_{\Gamma^n}(\nabla_{\Gamma^n} + \mathbf{d}_{\Gamma^n}^{n-1}b_1) = 0 \tag{17}$$

from which we get

$$b_1 = -(\mathbf{d}_{\Gamma^n}^{n-1^T}\mathbf{G}_{\Gamma^n}\mathbf{d}_{\Gamma^n}^{n-1})^{-1}(\mathbf{d}_{\Gamma^n}^{n-1^T}\mathbf{G}_{\Gamma^n}\nabla_{\Gamma^k}). \tag{18}$$

The optimal step-size can also be easily calculated as [7, pp. 521]:

$$a^n = r^{n^T}\mathbf{c}^n/(\mathbf{c}^{n^T}\mathbf{c}^n), \tag{19}$$

where $\mathbf{c}^n$ is the vector $\mathbf{c}^n = \Phi_{\Gamma^n}\mathbf{d}_{\Gamma^n}^n$.

The update direction is not guaranteed to be $\mathbf{G}_{\Gamma^n}$-conjugate to all previous update directions, however, we now only require a single vector to be kept in memory between iterations. Furthermore:

$$(\mathbf{d}_{\Gamma^n}^{n-1^T}\mathbf{G}_{\Gamma^n}\mathbf{d}_{\Gamma^n}^{n-1}) = (\Phi_{\Gamma^n}\mathbf{d}_{\Gamma^n}^{n-1})^T(\Phi_{\Gamma^n}\mathbf{d}_{\Gamma^n}^{n-1}) \tag{20}$$

where $\Phi_{\Gamma^n}\mathbf{d}_{\Gamma^n}^{n-1} = \mathbf{c}^{n-1}$ has been evaluated in the previous iteration to determine the step size a^n and the same is true for the denominator, which is nothing else than the product $\mathbf{c}^{n-1^T}\mathbf{c}^{n-1}$ calculated in the previous iteration. Therefore, the algorithm uses the same computations as Matching Pursuit, with the addition of two more matrix vector multiplication, one to evaluate b_1 and one to evaluate the step-size $\mathbf{c}^n$.

6.2 Using the Gradient Direction

One further approximation and simplification could be made by setting $b_1 = 0$ so that only a single additional matrix multiplication is required compared to Matching Pursuit. This method is then a gradient descent procedure we call Gradient Pursuit (GP). The update direction is now simply chosen to be the current gradient ∇_{Γ^n}, i.e

$$\mathbf{d}_{\Gamma^n}^n = \nabla_{\Gamma^n}. \tag{21}$$

The step size can also be calculated as in equation (19).

6.3 Approximate orthogonal Matching Pursuit Algorithm

The approximate orthogonal Matching Pursuit algorithms are then:

1. Initialise: $\mathbf{r}^0 = \mathbf{x}, \mathbf{y}^0 = 0, \Gamma^0 = \oslash$
2. for $n = 1; n := n+1$ till stopping criterion is met
   (a) $\nabla^n = \Phi^T\mathbf{r}^n$
   (b) $i^n = \arg_i\max|\nabla_i|$
   (c) $\Gamma^n = \Gamma^{n-1} \cup i^n$
   (d) if $n = 1$
   - $\mathbf{d}_{\Gamma^1}^1 = 1$

   (e) if $n \neq 1$
   - for the approximate conjugate direction method: $b_1 = -\mathbf{c}^{n-1^T}\Phi_{\Gamma^n}\nabla_{\Gamma^k}/(\mathbf{c}^{n-1^T}\mathbf{c}^{n-1})$ for the gradient descent method $b_1 = 0$
   - $\mathbf{d}_{\Gamma^n}^n = \nabla_{\Gamma^n} - b_1\mathbf{d}_{\Gamma^n}^{n-1}$

   (f) $\mathbf{c}^n = \Phi_{\Gamma^n}\mathbf{d}_{\Gamma^n}^n$
   (g) $a^n = \mathbf{r}^{n^T}\mathbf{c}^n/(\mathbf{c}^{n^T}\mathbf{c}^n)$
   (h) $\mathbf{y}_{\Gamma^n}^n = \mathbf{y}_{\Gamma^n}^{n-1} + a^n\mathbf{d}_{\Gamma^n}^n$
   (i) $\mathbf{r}^n = \mathbf{r}^{n-1} - a^n\mathbf{c}^n$
3. Output: $\mathbf{r}^n, \mathbf{y}^n$

7. EXPERIMENTAL EVALUATION

To give an indication of the stability of the different implementations of exact orthogonal Matching Pursuit we used ill conditioned dictionaries generated by perturbing a single

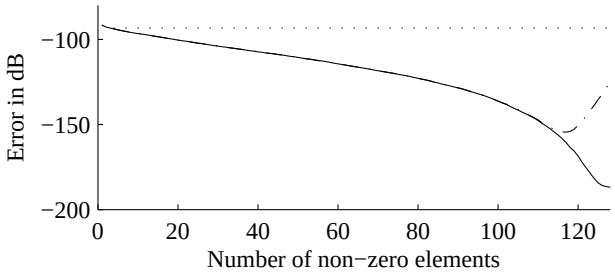

Figure 1: Comparison between matrix inverse based (dashed), QR based (dash-dotted) and conjugate direction based (solid) implementation of orthogonal Matching Pursuit for a dictionary with condition number of $3 * 10^6$.

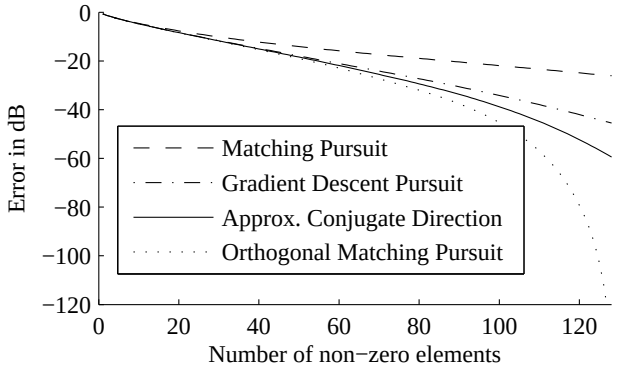

Figure 2: Comparison between MP (dashed), CGP (dotted), the GP (dash dotted) and ACGP (solid). Data averaged over 1 000 randomly generated signals (see text).

randomly generated atom to create all other dictionary elements. Average results for dictionaries of size 128×256 with condition number of around $3 * 10^6$ are shown in figure 1.

To evaluate the Gradient Pursuit and the Approximate Conjugate Gradient Pursuit algorithms we used a toy example and compared the performance to standard Matching Pursuit and orthogonal Matching Pursuit. We generated 1 000 dictionaries of size 128×256, drawing elements ϕ_i uniformly from the unit sphere. From each dictionary we selected 64 elements at random and multiplied these with unit variance zero mean Gaussian coefficients to generate 1 000 test signals.

The averaged results are shown in figure 2 where we plot the number of non-zero coefficients against the approximation error in dB. From these results it can be seen that the approximation to the conjugate direction leads to better results than the use of the simpler gradient based approximation to orthogonal Matching Pursuit. Both methods are shown to outperform Matching Pursuit and for a wide range of sparsity levels perform nearly as good as orthogonal Matching Pursuit.

8. CONCLUSIONS

In this paper we have introduced a conjugate direction implementation of orthogonal Matching Pursuit which has a similar computational complexity as currently used QR factorisation base approaches. The memory requirements of orthogonal Matching Pursuit implementations such as those based on QR factorisation as well as our new conjugate direction based method are often too high. We therefore have introduced two new directional update schemes that approximate the conjugate update direction. These do not require additional storage. Furthermore they can be implemented very efficiently.

Table 1: Comparison of the methods in terms of computational requirements in iteration n.

Algo.	Multiplications of vector by Φ	Storage
MP	1	0
GP	2	0
ACGP	3	0
CGP a)	$n+3(+1$ mult. by $\mathbf{D}\Phi^T)$	$\mathbf{D}$
CGP b)	$3(+1$ by $\mathbf{D}$ and 1 by $\mathbf{D}^T\mathbf{G})$	$\mathbf{D}$, $\mathbf{D}^T\mathbf{G}$

The gradient based method only requires the additional evaluation of the step size, while the approximation to the conjugate direction involves one further matrix vector multiplication. Typically used over-complete dictionaries are unions of transforms for which fast algorithms are available, so that multiplications involving the dictionary can be evaluated much faster than general matrix multiplications. The computational cost in terms of multiplications of vectors with the dictionary and additional storage is summarised in table 1 for the Matching Pursuit (MP), the Gradient Pursuit (GP), the Approximate Conjugate Gradient Pursuit (ACGP) and two implementations of Conjugate Gradient Pursuit (CGP), one, which only stores $\mathbf{D}$ and one that also iteratively updates and stores $\mathbf{D}^T\mathbf{G}$. Note that the last two algorithms also require general vector matrix multiplications as given in the parenthesis.

REFERENCES

[1] G. Davis, *Adaptive Nonlinear Approximations*. PhD thesis, New York University, New York, USA, 1994.

[2] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[3] Y. C. Pati, R. Rezaifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in 27^{th} *Asilomar Conf. on Signals, Systems and Comput.*, Nov. 1993.

[4] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[5] S. Krustulovic and R. Gribonval, "MPTK: Matching pursuit made tractable," in *Proc. Int. Conf. on Acoustic Speech and Signal Processing*, (Toulouse, France), May 2006.

[6] S. Mallat, G. Davis, and Z. Zhang, "Adaptive time-frequency decompositions," *SPIE Journal of Optical Engineering*, vol. 33, pp. 2183–2191, July 1994.

[7] G. H. Golub and F. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 3rd ed., 1996.

[8] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Addison Wesley, September 1999.