

Recovery of Sparse Representations by Polytope Faces Pursuit

Mark D. Plumbley

Department of Electronic Engineering, Queen Mary University of London,
Mile End Road, London E1 4NS, United Kingdom.
Email: `mark.plumbley@elec.qmul.ac.uk`

Abstract. We introduce a new greedy algorithm to find approximate sparse representations \mathbf{s} of $\mathbf{x} = \mathbf{A}\mathbf{s}$ by finding the Basis Pursuit (BP) solution of the linear program $\min\{\|\mathbf{s}\|_1 \mid \mathbf{x} = \mathbf{A}\mathbf{s}\}$. The proposed algorithm is based on the geometry of the polar polytope $P^* = \{\mathbf{c} \mid \tilde{\mathbf{A}}^T \mathbf{c} \leq \mathbf{1}\}$ where $\tilde{\mathbf{A}} = [\mathbf{A}, -\mathbf{A}]$ and searches for the vertex $\mathbf{c}^* \in P^*$ which maximizes $\mathbf{x}^T \mathbf{c}$ using a path following method. The resulting algorithm is in the style of Matching Pursuits (MP), in that it adds new basis vectors one at a time, but it uses a different correlation criterion to determine which basis to add and can switch out basis vectors as necessary. The algorithm complexity is of a similar order to Orthogonal Matching Pursuits (OMP). Experimental results show that this algorithm, which we call *Polytope Faces Pursuit*, produces good results on examples that are known to be hard for MP, and it is faster than the interior point method for BP.

1 Introduction

Suppose are given a sequence of observations $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_T]$, $\mathbf{x}_t \in \mathbb{R}^d$, which we wish to decompose according to the usual independent component analysis (ICA) generative model $\mathbf{X} = \mathbf{A}\mathbf{S}$, where $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_n]$ is a $d \times n$ mixing matrix of real basis vectors \mathbf{a}_i and sequence $\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_T]$ of source coefficient vectors $\mathbf{s}_t \in \mathbb{R}^n$. If we have $n > d$, i.e. the number of basis vectors is larger than the dimensionality of the basis space, then the system is *overcomplete*. This means in particular that if we have identified the mixing matrix \mathbf{A} , using some dictionary learning method (see e.g. [1]), then the equation $\mathbf{x}_t = \mathbf{A}\mathbf{s}_t$ at a particular t still has multiple solutions in general for \mathbf{s}_t given \mathbf{A} and \mathbf{x}_t . In *sparse coding*, we favour the minimum ℓ_0 -norm solution \mathbf{s} to $\mathbf{x} = \mathbf{A}\mathbf{s}$ which has the smallest number of non-zero elements $\|\mathbf{s}\|_0$,

$$\min_{\mathbf{s}} \|\mathbf{s}\|_0 \quad \text{such that} \quad \mathbf{x} = \mathbf{A}\mathbf{s} \quad (1)$$

However, finding the ℓ_0 solution (1) is known to be a hard problem. Instead, the method of *Basis Pursuit* (BP) [2] proposes to find the minimum ℓ_1 norm solution

$$\min_{\mathbf{s}} \|\mathbf{s}\|_1 \quad \text{such that} \quad \mathbf{x} = \mathbf{A}\mathbf{s} \quad (2)$$

which is equivalent to a linear programming (LP) problem and can be solved with e.g. interior point methods. However these methods can still be slow, so in practice greedy algorithms such as *Matching Pursuits* (MP) [3] or *Orthogonal Matching Pursuit* (OMP) [4] have been used as an efficient way to find approximate solutions to (2). Recently there has been much investigation into the conditions under which the solutions to (1) and (2) coincide (ℓ_1/ℓ_0 equivalence) and will be found by MP/OMP (exact recovery condition). For discussions see e.g. [5–8] and references therein.

In an interesting new direction, Donoho [9] has shown that by considering the d -dimensional *polytope* (the d -dimensional generalization of a polygon) whose vertices are the $2n$ signed basis vectors $\pm \mathbf{a}_i$, $\mathbf{a}_i \in \mathbf{A}$, results from the theory of polytopes can give us insight into the question of ℓ_1/ℓ_0 equivalence. Using a similar geometric approach, the present author [10] has explored the geometry of the problem (2), giving a visualization for the conditions discussed by Fuchs [7] and Tropp [6] for a unique optimal solution to (2), and hence ℓ_1/ℓ_0 equivalence.

In this paper we build on this geometrical visualization to propose a new greedy algorithm that performs Basis Pursuit (2). The algorithm adopts a path-following approach through the relative interior of the faces of the polar (dual) polytope associated with the dual LP problem. We refer to this as the *Polytope Faces Pursuit* algorithm.

2 Dual linear programs and polar polytopes

It is sometimes convenient to convert (2) into its standard form [2, 11]

$$\min_{\tilde{\mathbf{s}}} \mathbf{1}^T \tilde{\mathbf{s}} \quad \text{such that} \quad \mathbf{x} = \tilde{\mathbf{A}} \tilde{\mathbf{s}}, \tilde{\mathbf{s}} \geq 0 \quad (3)$$

where $\tilde{\mathbf{A}} = [\mathbf{A}, -\mathbf{A}]$ and $\tilde{\mathbf{s}}$ has $2n$ nonnegative components $\tilde{s}_i = \max(s_i, 0)$ for $1 \leq i \leq n$ and $\tilde{s}_i = \max(-s_{i-n}, 0)$ for $n+1 \leq i \leq 2n$. Clearly $s_i = \tilde{s}_i - \tilde{s}_{i+n}$. The new linear program (3) has a corresponding *dual* linear program [2]

$$\max_{\mathbf{c}} \mathbf{x}^T \mathbf{c} \quad \text{such that} \quad \tilde{\mathbf{A}}^T \mathbf{c} \leq \mathbf{1} \quad (4)$$

which has an optimum \mathbf{c}^0 associated with any optimum $\tilde{\mathbf{s}}^0$ of (3). Thus to perform BP we can search for the optimum \mathbf{c} in (4) and solve the resulting (determined) system for $\tilde{\mathbf{s}}$.

To help us visualize this search space, we introduce some geometric concepts. A d -dimensional *polytope* P is a bounded subset of \mathbb{R}^d defined by a finite set of inequalities, or alternatively as the convex hull of a finite set of extreme points, its *vertices*. If the equality $\mathbf{a}^T \mathbf{x} \leq b$ is *valid* for P , i.e. $\mathbf{a}^T \mathbf{x} \leq b$ for all $\mathbf{x} \in P$, then $F = \{\mathbf{x} \in P \mid \mathbf{a}^T \mathbf{x} \leq b\}$ is a *face* of P . Examples of faces include the *improper* faces \emptyset and P itself, as well as the vertices (0-dimensional faces) and facets ($(d-1)$ -dimensional faces) of P . For more definitions and notation see e.g. [9, 12].

The dotted polygon in Fig. 1(a) shows the *primal polytope* P with vertices given by the signed basis vectors $\pm \mathbf{a}_i$. Of more interest to us is the *polar polytope*

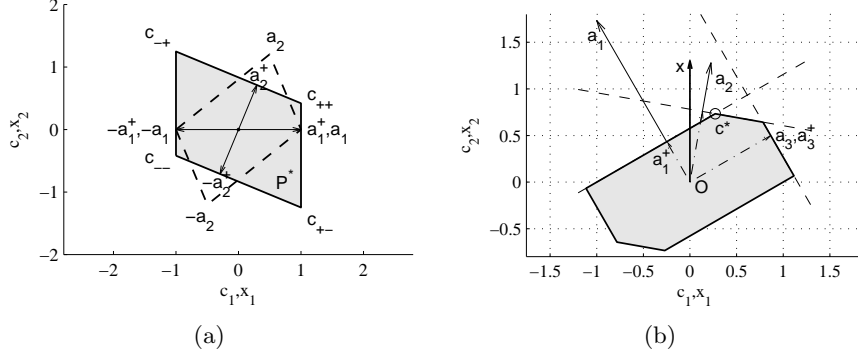


Fig. 1. Examples of polytopes in 2-D: (a) primal (dashed) and polar (solid) polytopes; (b) polar polytope showing ideal basis vertex c^* .

P^* defined by the inequalities $P^* = \{\mathbf{c} \mid \tilde{\mathbf{a}}_i^T \mathbf{c} \leq 1\}$ where $\tilde{\mathbf{a}}_i \in \tilde{\mathbf{A}}$, i.e. $\tilde{\mathbf{a}}_i \in \{+\mathbf{a}_1, +\mathbf{a}_2, -\mathbf{a}_1, -\mathbf{a}_2\}$. The scaled vectors $\mathbf{a}_i^\dagger = \mathbf{a}_i/|\mathbf{a}_i|^2$ satisfy $\mathbf{a}_i^T \mathbf{a}_i^\dagger = 1$ so they touch the faces (extended if necessary) defined by $\mathbf{a}_i^T \mathbf{c} = 1$. Clearly this polar polytope P^* is the feasible region for \mathbf{c} in (4). The vertices \mathbf{c}_{++} etc. of P^* correspond to particular sets of selected vertices. If we let $\mathbf{A}_{+-} = [+ \mathbf{a}_1, -\mathbf{a}_2]$ then \mathbf{c}_{+-} is the vertex such that $\mathbf{A}_{+-}^T \mathbf{c}_{+-} = \mathbf{1}$, i.e. $\mathbf{c}_{+-} = \mathbf{A}_{+-}^\dagger \mathbf{1}$ where \mathbf{A}^\dagger is the Moore-Penrose pseudoinverse of \mathbf{A} (used so that we can define \mathbf{c} for an \mathbf{A} with less than d columns). It is a standard result from linear programming that the optimum of (4) will be achieved at one of the vertices [11]. Therefore it remains for us to identify which is the optimal vertex which maximizes $\mathbf{x}^T \mathbf{c}$, and use that to find the optimal vector $\tilde{\mathbf{s}}$ (and hence \mathbf{s}).

As an example, consider the shaded polytope in Fig. 1(b), which is defined by three basis vector pairs $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$. The scaled pseudo-inverse vectors $+\mathbf{a}_i^\dagger$, $i = 1, 2, 3$ are also shown. The dual vector \mathbf{c} that maximizes the inner product $\mathbf{x}^T \mathbf{c}$ with \mathbf{x} is the one furthest along the direction parallel to \mathbf{x} , and is marked \mathbf{c}^* . In this particular case \mathbf{c}^* corresponds to the optimal basis set $\tilde{\mathbf{A}}_{\text{opt}} = [+ \mathbf{a}_1, +\mathbf{a}_2]$, which has the corresponding primal solution $\tilde{\mathbf{s}} = \tilde{\mathbf{A}}_{\text{opt}}^\dagger \mathbf{x}$, or in original form $\mathbf{s} = \mathbf{A}_{\text{opt}}^\dagger \mathbf{x}$ where $\mathbf{A}_{\text{opt}} = [\mathbf{a}_1, \mathbf{a}_2]$.

However, consider what happens if we apply either MP or OMP, suitably adjusted for non-unit-norm basis vectors, to the situation pictured in Fig. 1(b). The first vector selected is $\arg \max_{\mathbf{a}_i} \mathbf{a}_i^T \mathbf{x} = \mathbf{a}_1$, so after the first step, $\mathbf{A}^1 = [\mathbf{a}_1]$. This produces a residual \mathbf{r}^1 perpendicular to \mathbf{a}_1 , i.e. along \mathbf{a}_3 , so on the second step both MP and OMP will select \mathbf{a}_3 . The basis set $\mathbf{A}^2 = [\mathbf{a}_1, \mathbf{a}_3]$ now spans the space, the residual \mathbf{r}^2 is zero, and both MP and OMP stop after 2 steps. MP and OMP have the same behaviour in this case because \mathbf{a}_1 and \mathbf{a}_3 are orthogonal. Thus both MP and OMP have failed to find the optimal solution to (2). In Natarajan's algorithm [13], sometimes called *Order Recursive Matching Pursuit*, there would be an arbitrary choice between \mathbf{a}_2 and \mathbf{a}_3 in step 2, since both would give zero residual after selection, so this can also fail on this example.

3 Deriving the Faces Pursuit Algorithm

Let us now derive an algorithm to find the true optimal LP solution of (2), but will build up its solution in a similar way to MP/OMP. Our first insight is that if we project from the origin O along $\mathbf{h} = \alpha \mathbf{x}$ for $\alpha \geq 0$ the first polytope face we encounter is at

$$\alpha_1 = \min\{\alpha > 0 \mid \mathbf{a}_i^T(\alpha \mathbf{x}) = 1\} = \min\{\alpha > 0 \mid \mathbf{a}_i^T \mathbf{x} = 1/\alpha\} = 1/\max\{\mathbf{a}_i^T \mathbf{x}\}$$

which is our normal MP/OMP maximum correlation condition. Let us select \mathbf{a}_1 as our first vector, and then continue our path ‘towards’ \mathbf{x} , but with our path now constrained to be within the polytope face $F^1 = \{\mathbf{h} \in P^* \mid \mathbf{a}_1^T \mathbf{h} = 1\}$. We can achieve this by projecting \mathbf{x} into the subspace parallel to F^1 , to give $\mathbf{r}^1 = (\mathbf{I} - \mathbf{Q}_1)\mathbf{x}$ where $\mathbf{Q}_1 = \mathbf{a}_1 \mathbf{a}_1^{\dagger T} = \mathbf{a}_1 \mathbf{a}_1^T / |\mathbf{a}_1|^2$. Since $\tilde{\mathbf{s}}^1 = \mathbf{a}_1^{\dagger T} \mathbf{x}$ and $\hat{\mathbf{x}}^1 = \mathbf{a}_1 \tilde{\mathbf{s}}^1$ we have $\mathbf{r}^1 = \mathbf{x} - \mathbf{a}_1 \tilde{\mathbf{s}}^1 = \mathbf{x} - \hat{\mathbf{x}}^1$ so \mathbf{r}^1 is therefore the residual from the approximation $\hat{\mathbf{x}}^1$ to \mathbf{x} obtained after Step 1 (Fig. 2)

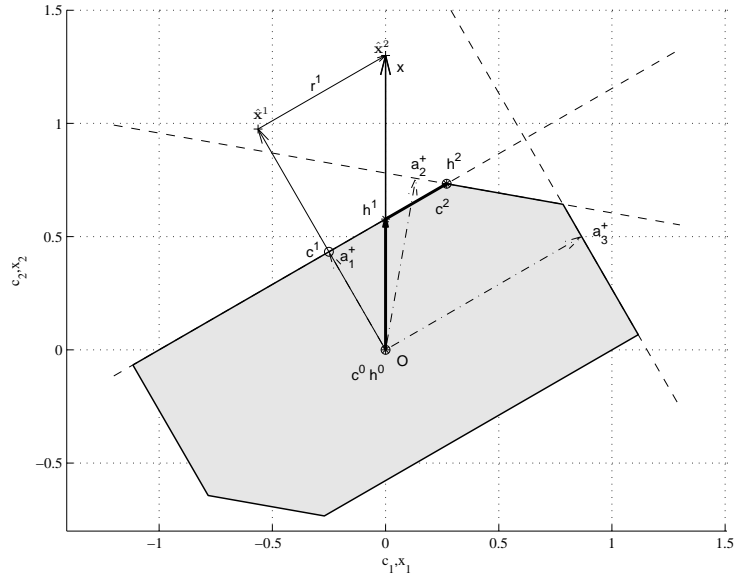


Fig. 2. Path of the Faces Pursuit Algorithm

The second step is where the difference from MP/OMP arises. These would find $\max_i \mathbf{a}_i^T \mathbf{r}^1$, but this is the first face encountered projecting along the residual \mathbf{r}^1 from the origin, not within face F^1 . Instead, to correctly determine the next face we encounter along the face F^1 we project along the residual starting at $\mathbf{h}_1 = \alpha_1 \mathbf{x}$. A little manipulation will confirm that the next face is encountered at at $\min\{\alpha > \alpha^1 \mid \mathbf{a}_i^T(\mathbf{c}^1 + \alpha \mathbf{r}^1) = 1\} = 1/\max\{(\mathbf{a}_i^T \mathbf{r}^1)/(1 - \mathbf{a}_i^T \mathbf{c}^1) \mid \mathbf{a}_i^T \mathbf{r}^1 > 0\}$

where $\mathbf{c}^1 = \mathbf{a}_1^\dagger \cdot \mathbf{1}$ and we exclude faces already encountered. Further consideration along these lines shows that this generalizes so at each step k we do not simply want the maximum correlation $\mathbf{a}_i^T \mathbf{r}^{k-1}$, but the maximum scaled correlation

$$\mathbf{a}^k = \arg \max_{\mathbf{a}_i} \frac{\mathbf{a}_i^T \mathbf{r}^{k-1}}{1 - \mathbf{a}_i^T \mathbf{c}^{k-1}} \quad (5)$$

where we consider only vectors \mathbf{a}_i for which $\mathbf{a}_i^T \mathbf{r}^{k-1} > 0$. Note that if we have a fast method to compute $\mathbf{a}_i^T \mathbf{r}^{k-1}$, such as a fast Wavelet transform if \mathbf{A} is a wavelet basis, we can use the same method to compute $\mathbf{a}_i^T \mathbf{c}^{k-1}$.

In the complete algorithm (Algorithm 1) we also need to be able to optionally switch out certain constraints once we have encountered a new face (consider ‘climbing up’ the face corresponding to \mathbf{a}_3 instead of \mathbf{a}_1 in Fig. 2: we leave that constraint once we encounter the face corresponding to \mathbf{a}_2). A little manipulation will show that constraints should be switched out after step k if $\tilde{\mathbf{s}}^k$ contains any negative entries: the negative entries corresponding to the constraints to be removed. We also have to take practical steps to ensure that $\mathbf{a}_i^T \mathbf{c}^k \approx 1$ does not give divide-by-zero errors, and to ensure the same bases are not considered candidates again.

Algorithm 1 Polytope Faces Pursuit

- 1: Input: $\tilde{\mathbf{A}} = [\tilde{\mathbf{a}}_i]$, \mathbf{x} {If required, set $\tilde{\mathbf{A}} \leftarrow [\mathbf{A}, -\mathbf{A}]$ }
 - 2: Set stopping conditions k_{\max} and θ_{\min}
 - 3: Initialize: $k \leftarrow 0$, $\mathcal{I}^k \leftarrow \emptyset$, $\tilde{\mathbf{A}}^k \leftarrow \emptyset$, $\mathbf{c}^k \leftarrow \mathbf{0}$, $\tilde{\mathbf{s}}^k \leftarrow \emptyset$, $\hat{\mathbf{x}}^k \leftarrow \mathbf{0}$, $\mathbf{r}^k \leftarrow \mathbf{x}$
 - 4: **while** $k < k_{\max}$ and $\max_i \tilde{\mathbf{a}}_i^T \mathbf{r}^{k-1} > \theta_{\min}$ **do** {Find next face}
 - 5: $k \leftarrow k + 1$
 - 6: Find face: $i^k \leftarrow \arg \max_{i \notin \mathcal{I}^{k-1}} \{(\tilde{\mathbf{a}}_i^T \mathbf{r}^{k-1}) / (1 - \tilde{\mathbf{a}}_i^T \mathbf{c}^{k-1}) \mid \tilde{\mathbf{a}}_i^T \mathbf{r}^{k-1} > 0\}$
 - 7: Optionally: $\alpha^k \leftarrow (1 - \tilde{\mathbf{a}}_{i^k}^T \mathbf{c}^{k-1}) / (\tilde{\mathbf{a}}_{i^k}^T \mathbf{r}^{k-1})$, $\mathbf{h}^k \leftarrow \mathbf{c}^{k-1} + \alpha^k \mathbf{r}^{k-1}$
 - 8: Add constraint: $\tilde{\mathbf{A}}^k \leftarrow [\tilde{\mathbf{A}}^{k-1}, \mathbf{a}_{i^k}]$, $\mathcal{I}^k \leftarrow \mathcal{I}^{k-1} \cup \{i^k\}$, $\mathbf{B}^k \leftarrow (\tilde{\mathbf{A}}^k)^\dagger$, $\tilde{\mathbf{s}}^k \leftarrow \mathbf{B}^k \mathbf{x}$
 - 9: **while** $\tilde{\mathbf{s}}^k \not\geq \mathbf{0}$ **do** {Release retarding constraints}
 - 10: Select some $j \in \mathcal{I}^k$ such that $\tilde{s}_j^k < 0$; remove column \mathbf{a}_j from $\tilde{\mathbf{A}}^k$
 - 11: Update: $\mathcal{I}^k \leftarrow \mathcal{I}^{k-1} \setminus \{j\}$, $\mathbf{B}^k \leftarrow (\tilde{\mathbf{A}}^k)^\dagger$, $\tilde{\mathbf{s}}^k \leftarrow \mathbf{B}^k \mathbf{x}$
 - 12: **end while**
 - 13: $\mathbf{c}^k \leftarrow (\mathbf{B}^k)^T \mathbf{1}$, $\hat{\mathbf{x}}^k \leftarrow \tilde{\mathbf{A}}^k \tilde{\mathbf{s}}^k$, $\mathbf{r}^k \leftarrow \mathbf{x} - \hat{\mathbf{x}}^k$
 - 14: **end while**
 - 15: Output: $\mathbf{c}^* = \mathbf{c}^k$, $\tilde{\mathbf{s}}^* \leftarrow \mathbf{0}$ + corresponding entries from $\tilde{\mathbf{s}}^k$
 {If required, get $s_i^* \leftarrow (\tilde{s}_i^* - \tilde{s}_{i+n}^*)$, $1 \leq i \leq n$ }
-

The most expensive operations in the algorithm are the dictionary analysis calculations $\tilde{\mathbf{a}}_i^T \mathbf{r}^{k-1}$ and $\tilde{\mathbf{a}}_i^T \mathbf{c}^{k-1}$ (two per step k instead of one per step for MP/OMP) and the pseudoinverse calculation $(\tilde{\mathbf{A}}^k)^\dagger$ (one per step k as for OMP, plus one each retarding constraint release in the inner loop). Thus the complexity is similar to OMP but with an additional dictionary probe each loop. Constraint releases appear to be relatively infrequent: on the Gong example in Section 4

constraint releases occur on average every 5–6 steps. Note that the Algorithm 1 does not require \mathbf{A} to be a unit norm dictionary.

4 Experiments

To confirm the operation of the Polytope Faces Pursuit algorithm and compare it with MP and the interior point method for BP, we applied the algorithm to some examples where MP and BP have already been compared [2]. Fig. 3 shows MP, OMP, BP and the proposed Polytope Faces Pursuit algorithm (FP) applied to the signal ‘TwinSine’, a superposition of two sinusoids separated by less than the Rayleigh Distance $2\pi/n$. The analysis is performed in a 4-fold overcomplete

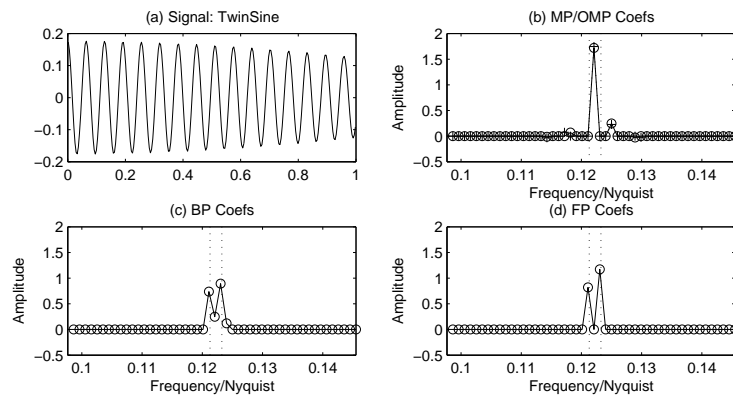


Fig. 3. The ‘TwinSine’ signal (see [2]) with decompositions by (b) MP (‘o’) and OMP (‘+’), (c) BP using the interior point method, and (d) Polytope Faces Pursuit.

discrete cosine dictionary [2]. The stopping condition θ_{\min} was set to 10^{-2} times the signal norm for MP and Polytope Faces Pursuit.

As is already known, MP and OMP fail to resolve this signal, in that they both initially select the middle frequency atom (atom 126) and subsequently do not remove it, while BP (using the interior point method) does resolve the signal correctly. Faces Pursuit quickly produces a clean sparse decomposition: atom 126 is initially selected, the same as for MP/OMP as we would expect, but following the addition of atoms 125 and 127 atom 126 it is removed at step 3 as a retarding constraint. Thus Faces Pursuit stops in 3 steps, yielding a sparse representation consisting of only 2 basis atoms (Fig. 3(d)).

Fig. 4 shows MP, BP and Polytope Faces Pursuit applied to the signal ‘Gong’, which is zero until t_0 and then follows a decaying sinusoid. The analysis is performed using a cosine packet dictionary [2]. As already known, BP using the interior point method produces a ‘cleaner’ decomposition than MP, although it is approximately an order of magnitude slower than MP on this example. We

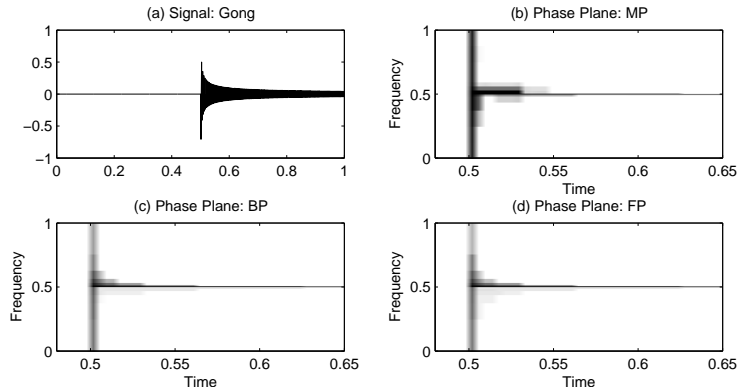


Fig. 4. The ‘Gong’ signal (see [2]) with decompositions by (b) MP, (c) BP using the interior point method, and (d) Polytope Faces Pursuit.

can confirm that Polytope Faces Pursuit produces a similar result to BP, taking a time between MP and BP for the same accuracy. A typical run on the three algorithms using Matlab 7.0 (R14) under Windows XP on a 1.5GHz Intel Pentium M laptop takes $t_{MP} = 28s$, $t_{BP} = 224s$, $t_{FP} = 74s$. We expect to improve the speed of the Faces Pursuit algorithm in future through the use of matrix and vector updating and downdating formulae in place of the expensive pseudo-inverse calculation each step.

5 Conclusions

We have introduced a new greedy algorithm to find approximate sparse representations \mathbf{s} of $\mathbf{x} = \mathbf{A}\mathbf{s}$ given \mathbf{A} and \mathbf{x} . The algorithm is based on the geometry of the polar polytope $P^* = \{\mathbf{c} \mid \tilde{\mathbf{A}}^T \mathbf{c} \leq \mathbf{1}\}$ where $\tilde{\mathbf{A}} = [\mathbf{A}, -\mathbf{A}]$ which defines the feasible region of the dual linear program $\max\{\mathbf{x}^T \mathbf{c} \mid \tilde{\mathbf{A}}^T \mathbf{c} \leq \mathbf{1}\}$. The algorithm searches for the vertex $\mathbf{c}^* \in P^*$ which maximizes $\mathbf{x}^T \mathbf{c}$ using a path following method through the relative interior of faces of P . We call this method *Polytope Faces Pursuit*.

The resulting algorithm is in the style of Matching Pursuits (MP), in that it adds new basis vectors one at a time based in a correlation criterion, but has two major differences: (1) the correlation criterion depends on the current vertex \mathbf{c}^k at step k as well as the residual \mathbf{r}^k ; and (2) basis vectors are switched out if necessary. The algorithm complexity is of the same order as OMP, although it has one additional dictionary probe per step, and has costs associated with switching out of basis vectors.

Experimental results confirm that the Polytope Faces Pursuit algorithm produces good results on examples that are known to be challenging for MP, and that it is faster than the interior point method for Basis Pursuit (BP).

6 Acknowledgements

This work was partially supported by EPSRC grants GR/S82213/01, GR/S75802/01, EP/C005554/1 and EP/D000246/1. Some of the figures were generated using the Multi-Parametric Toolbox (MPT) for Matlab [14], WaveLab802 and Atomizer802 (<http://www-stat.stanford.edu/~wavelab/> and <http://www-stat.stanford.edu/~atomizer/>).

References

1. Kreutz-Delgado, K., Murray, J.F., Rao, B.D., Engan, K., Lee, T.W., Sejnowski, T.J.: Dictionary learning algorithms for sparse representation. *Neural Computation* **15** (2003) 349–396
2. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing* **20** (1998) 33–61
3. Mallat, S., Zhang, Z.: Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing* **41** (1993) 3397–3415
4. Pati, Y.C., Rezaiifar, R., Krishnaprasad, P.S.: Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In: *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA. (1993) 40–44
5. Gribonval, R., Nielsen, M.: Approximation with highly redundant dictionaries. In: *Wavelets: Applications in Signal and Image Processing, Proc. SPIE'03*, San Diego, USA. (2003) 216–227
6. Tropp, J.A.: Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory* **50** (2004) 2231–2242
7. Fuchs, J.J.: On sparse representations in arbitrary redundant bases. *IEEE Transactions on Information Theory* **50** (2004) 1341–1344
8. Donoho, D.L., Elad, M.: Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization. *Proc. Nat. Aca. Sci.* **100** (2003) 2197–2202
9. Donoho, D.L.: Neighborly polytopes and sparse solutions of underdetermined linear equations. Technical report, Statistics Department, Stanford University (2004)
10. Plumbley, M.D.: Polar polytopes and recovery of sparse representations (2005) Submitted for publication.
11. Schrijver, A.: *Theory of Linear and Integer Programming*. John Wiley & Sons Ltd, Chichester, UK (1998)
12. Grünbaum, B.: *Convex Polytopes*. Second edn. Graduate Texts in Mathematics 221. Springer-Verlag, New York (2003)
13. Natarajan, B.K.: Sparse approximate solutions to linear systems. *SIAM J. Computing* **25** (1995) 227–234
14. Kvasnica, M., Grieder, P., Baotić, M.: *Multi-Parametric Toolbox (MPT)* (2004)