APPLICATION OF TREE-BASED SEARCHES TO MATCHING PURSUIT

Shane F. Cotter and Bhaskar D. Rao

Electrical and Computer Engineering Department
University of California, San Diego
La Jolla, California 92093-0407

ABSTRACT

Matching Pursuit (MP) uses a greedy search to construct a subset of vectors, from a larger set, which will best represent a signal of interest. Here, we extend this search for the best subset by keeping the K vectors which maximize the selection criterion at each iteration. This is termed the MP:K algorithm and represents a suboptimal search through the tree of all possible subsets where each node is limited to having K children. As a more suboptimal search, we can use the M-L search to select a subset of dictionary vectors, leading to the MP:M-L algorithm. We compare the computation and storage requirements for three variants of the MP algorithm using these searches. Through simulations, the significantly improved performance obtained using the MP:K and MP:M-L algorithms is demonstrated. We conclude that the Modified Matching Pursuit (MMP) algorithm offers the best compromise between performance and complexity using these search techniques.

1. INTRODUCTION

In many applications, it is desirable to select a small number of vectors from a large dictionary (i.e., over-complete collection of vectors) to represent a signal. For instance, efficient coding of speech and video signals [1] can be achieved using such a representation. This problem has also received attention in areas as diverse as biomagnetic inverse problems and stock market analysis [2]. With this wide spectrum of applications, many different solutions have been proposed for the selection of the representation vectors.

In [3], the subset selection problem has been shown to be NP-hard. If we consider a dictionary of n vectors, $A = \{a_1, a_2, \cdots, a_n\}, a_i \in \mathcal{R}^m$, from which r are to be chosen to represent a signal, $b \in \mathcal{R}^m$, then there are a total of $N = \binom{n}{r}$ possible subsets which must be considered. For small values of n and r, an exhaustive search is possible over all subsets $S_i, i = 1, \cdots, N$ with the optimal set, S^*, minimizing the representation error, i.e., $S^* = \arg\min_{S_i} \|P_{S_i}^{\perp} b\|_2, i = 1, \cdots, N$. In the case where $m \leq n$, we may start from a representation employing all elements of the dictionary and then delete elements sequentially to obtain the optimal subset S^*. This is done by using a branch and bound algorithm [4] which can be used because the representation error can only increase as elements are deleted. This algorithm constructs a tree for the deletion of dictionary elements and efficiently searches through the feasible subsets. A best-first search through the tree results in a sequential elimination algorithm which has been examined in [5]. If $m > n$, then the representation error is 0, in general, until $> (n - m)$ elements have been deleted and so the branch and bound algorithm cannot be used to obtain a solution to this problem. An extension of the backward elimination algorithm to deal with an over-complete dictionary has been considered in [6].

Other approaches have also been suggested such as those based on minimizing diversity measures including functionals whose minimization promotes sparsity like the ℓ_1 norm [7] or the more general $\ell_{(p \leq 1)}$ norm [2, 8]. However, the class of algorithms considered here are Matching Pursuit (MP) algorithms which sequentially add elements to the representation subset. These algorithms offer a low cost suboptimal solution to the problem, frequently performing as well as the algorithms mentioned above [9]. Three variants of MP have been proposed and we have compared the performance and complexity of these algorithms in our previous work [9].

In this paper we consider two extensions of the MP paradigm. As outlined in section 2, each of the three MP algorithms proceeds by greedily selecting vectors to add to the representation subset. In section 3, we extend these MP algorithms by retaining the $K(\geq 1)$ vectors which best match the residual corresponding to each stored subset. These vectors are then used to give K new subsets and the K corresponding residuals are calculated. This algorithm, termed MP:K, essentially works through subtrees of the full tree where each node is limited to having K children. The algorithm terminates when the required depth is reached.

There are many types of tree search [10] but one that has been applied in many engineering applications is the M-L search [11]. The M-L algorithm has been used in speech recognition [12] as well as in obtaining improvements to the Viterbi algorithm in decoding symbols in ISI channels [13]. In section 4, we combine the M-L algorithm with MP in searching for a subset to represent the signal, giving the MP:M-L algorithm. We then present simulations in section 5 which compare the MP:K and MP:M-L algorithms. We draw some conclusions in section 6.

2. MATCHING PURSUIT ALGORITHMS

We review three different algorithms which sequentially form a subset of vectors to represent a signal. They are Basic Matching Pursuit (BMP) [14], Modified Matching Pursuit (MMP), also known as Orthogonal Matching Pursuit [15, 9]

This research was partially supported by the National Science Foundation Grant No. CCR-9902961.

and Order-Recursive Matching Pursuit (ORMP) [3]. In [9], these algorithms are fully described and their performance and computation discussed in detail. We briefly describe the algorithms here.

The dictionary of vectors is $A = [a_1, a_2, \cdots, a_n]$ and, for convenience, each vector is assumed to be of unit norm. At the pth iteration, the chosen subset of vectors is S_p with the indices stored in I_p. The residual vector is denoted by b_p, with $b_0 = b$.

In the BMP, at the pth iteration, the vector from A most closely aligned with the residual b_{p-1} is chosen, where the alignment is measured as the 2-norm (denoted by $\|\cdot\|$) of the projection of the residual onto the vector, i.e.,

$$\begin{aligned} k_p &= \arg\max_l \|P_{a_l} b_{p-1}\| \qquad (1) \\ &= \arg\max_l |a_l^H b_{p-1}|^2,\ l = 1, \cdots, n,\ l \neq k_{p-1}. \end{aligned}$$

The new residual vector is then computed as

$$b_p = b_{p-1} - P_{a_{k_p}} b_{p-1} = b_{p-1} - (a_{k_p}^H b_{p-1}) a_{k_p}. \qquad (2)$$

The algorithm terminates when the desired number, r, of vectors have been selected.

There is an inherent suboptimality in the calculation of the residual in the BMP algorithm, as given in (2), as seen by noting that

$$b_p^{BMP} = P_{a_{k_p}}^{\perp} b_{p-1} = \Pi_{l=1}^{p} P_{a_{k_l}}^{\perp} b \neq P_{S_p^{BMP}}^{\perp} b,\ a_{k_l} \in S_p^{BMP}.$$

At the pth iteration of the MMP and ORMP algorithms, the residual is obtained by projecting b onto the orthogonal complement of the range space of S_{p-1}, i.e., $b_{p-1} = P_{S_{p-1}}^{\perp} b$. The selection steps of MMP and ORMP use this residual but the selection criteria are different [9]. Due to space limitations, we omit these details here.

3. MP:K ALGORITHMS

In each of the MP algorithms, a greedy approach is taken to choosing which vector to add to the subset, S_{p-1}, based on the current value of the residual, b_{p-1}. Even though we calculate the alignment of b_{p-1} with all of the dictionary vectors in order to make this choice, we discard all this information. Instead, we propose in the MP:K algorithms to utilize this information to search other subsets which may be able to represent b more efficiently. Therefore, we keep the K best matching vectors and the K corresponding residuals obtained from the MP algorithm. Given the residual at the $(p-1)$th stage, we select the K best matching vectors $\{k_p^{(1)}, \cdots, k_p^{(K)}\}$ as follows:

$$k_p^{(i)} = \arg\max_l F(a_l, b_{p-1}),\ l \neq \{k_p^{(1)}, .., k_p^{(i-1)}\}, i = 1, .., K, \qquad (3)$$

where we have used the notation $F(a_l, b_{p-1})$ to denote the general selection criterion, i.e., that of either BMP, MMP or ORMP. We then form $\{b_p^{(1)}, b_p^{(2)}, \cdots, b_p^{(K)}\}$.

If we consider this algorithm in terms of a tree, we use the MP algorithms to rank which vectors should be added to the set S_{p-1} to best represent the signal b. However, because of the infeasibility of considering all such subsets, we only consider the K best matching vectors. We, therefore, limit each node in the tree to having K children. Even though we exclude certain vectors at a given level, these may enter the subset further down the tree.

We note that setting $K = 1$ just gives the "regular" MP solution and that this will be among the subsets generated for all values of K. Indeed, by ordering the search so that we descend along nodes corresponding to $k_p^{(1)}$, the "regular" MP solution is generated first. This corresponds to a best-first search [10]. The advantage of this is that if the answer is determined to be sufficiently good, e.g., the representation error is small, then there is no need for further search of the tree. However, searching more of the tree can often significantly improve the performance of the algorithms as will be seen in the simulations of section 5. Once the required depth, r, is reached, we are left with a set of subsets, $\{S_r^{(1)}, \cdots, S_r^{(L)}\}$ where there are $L = K^{(r-1)}$ leaf nodes and the residuals obtained when the signal is represented by each of these subsets. The subset yielding the smallest residual norm is chosen.

3.1. Computation

As has been mentioned, the tree search is performed by first searching along the best possible path. From this point, we need to backtrack to the previous level and then descend along any of the alternative paths from this level to the leaves of the tree. Then, we go up another level and descend along any paths from here - this process continues until all elements in the tree have been searched. Therefore, the main storage and computational requirements are introduced by the backtracking step.

In the MP:K algorithm as described above, the number of nodes at each level grows exponentially. It is easily seen that the total number of nodes will be

$$\#\ \text{Nodes} = \frac{K^r - 1}{K - 1},\ \forall K \geq 2$$

where K is the number of children at each node and r is the number of representation vectors sought. Of course, once the number of nodes becomes too large, it is possible to form a new root node and recommence the MP:K search from this node.

For the BMP, the residual and residual norm must be available for backtracking once the best-first path has been examined. Therefore, the number of storage locations (each location holds a floating point number) required for the residual norm is $K(r-1)$ and for the actual residual is $mK(r-1)$. In the case of MMP, $\{q_p^{(1)}, q_p^{(2)}, \cdots, q_p^{(K)}\}$ must be formed to obtain the subsets $\{S_p^{(1)}, \cdots, S_p^{(K)}\}$, i.e., at each node, we have S_{p-1} and form $S_p^{(i)} = [S_{p-1}, q_p^{(i)}], i = 1, .., K$. This requires K additional Gram-Schmidt computations [9] at each node.

The selection step in ORMP can be written as [9]

$$k_p = \arg\max_l \frac{|a_l^H b_{p-1}|}{\|a_l^{(p-1)}\|},\ l \notin I_{p-1}, \qquad (4)$$

where $a_l^{(p-1)} = P_{S_{p-1}}^{\perp} a_l$. The norms $\|a_l^{(p-1)}\|, l \notin I_{p-1}$, increase the computation and storage over that required

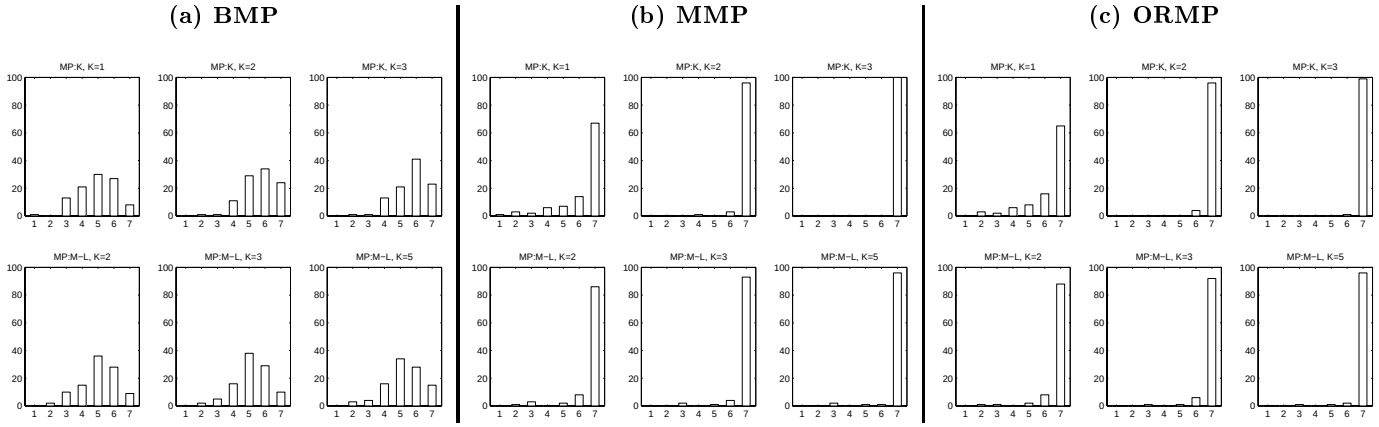

Fig. 1. For each of the three methods, in the first row the MP:K results are shown with $K = 1, 2, 3$ (recall that $K = 1$ gives the "regular" MP results). In the second row, the MP:M-L results are given for $K = 2, 3, 5$. Each histogram gives the percentage of trials in which 1-7 components are correctly identified.

in MMP. In order to backtrack we must have the norms, $\|a_l^{(p)}\|^2, l = 1, \cdots, n$, available at each level of the tree and this requires $n(r-1)$ storage locations. It is possible to obtain these quantities recursively as given in [9]

$$\|a_l^{(p)}\|^2 = \|a_l^{(p-1)}\|^2 - (|a_l^H q_p|)^2. \quad (5)$$

Moving down the tree, we need to calculate the final term in this expression at each node for $\{q_p^{(1)}, q_p^{(2)}, \cdots, q_p^{(K)}\}$. This means that there is a significant increase in computation of $\approx nK$ inner-products at each node.

4. MP:M-L ALGORITHMS

The problem with the previous search is that the number of subsets increases exponentially with the depth of the tree. In the M-L search [11] there are M paths at each level and the final level is L. In keeping with the notation above, we will let $M = K$ and so at each level we retain only the K nodes which have the lowest residual norm and discard all others. For each of the K nodes at a given level, we use the MP algorithm, as in (3), to find the best residuals, $\{b_p^{(1)}, \cdots, b_p^{(K)}\}$, at the next level. There are K^2 residuals to choose from. We expand only the nodes with the K lowest cost residuals and prune the other nodes. We move to the next level and continue until we reach the required depth.

This is also a tree search but is more suboptimal than the search outlined in the previous section since we reduce the number of nodes at each level to K. In the previous case, we were ensured of having the MP solution ($K = 1$) as one of the solutions which would be considered. This is not the case here as, with $K > 1$, we may be taken away from this solution at some level in the tree. It is possible also that different paths give the same residual so these two nodes are merged and the next highest cost node is used to give an alternative path through the tree.

An advantage of this method is that there are only $K(r-1)$ MP computations required. In addition there is no backtracking so only the current level must be stored which reduces the storage. The ORMP will still require storage of the norms, $\|a_l^{(p-1)}\|, l \notin I_{p-1}$, along each surviving path.

5. SIMULATIONS

We consider two simulations in keeping with our previous work on MP algorithms [9]. In the first simulation, there is no correlation between the dictionary elements while in the second simulation the dictionary is highly correlated.

5.1. Experiment 1

In this experiment, the dictionary is created as a random $m \times n$ matrix, A, whose entries are Gaussian random variables with mean 0 and variance 1. A sparse solution, x_s, with a specified number of nonzero entries, r, is then created; the indices of these r entries are random, and their amplitudes are random. The vector b is then computed as $b = Ax_s$. Noise is added to b to give a set value of SNR. The algorithms are then run with the value of r known.

We treat the problem as a component detection problem and determine how successfully we can find the vectors which were used in generating the observed signal, b. Once all the solutions have been produced by MP:K or MP:M-L for a given value of K, the subset corresponding to the smallest residual norm is chosen as the suboptimal solution. The vectors in this subset are then compared to the generating vectors. The experiment was run with $m = 20, n = 30$ and $r = 7$ with SNR set to 60dB.

Results

In the first row of figure 1, the results are plotted for BMP, MMP and ORMP where MP:K is used with $K = 1, 2, 3$. The second row shows the results for the M-L algorithm with $K = 2, 3, 5$ for each algorithm. In figure 1(a), increasing K improves the performance of BMP but due to the inherent suboptimality of the BMP [9], our success is relatively poor in identifying all r of the generating vectors with this small tree depth. This problem is overcome by using the MMP or ORMP algorithms and the usefulness of the larger value of K becomes clear.

In figure 1(b), we note that for MMP there is a large improvement in successfully finding all seven components which were used in generating the observed vectors. For MMP:K, the percentage of trials where all seven compo-

nents are detected is increased from 66% for $K = 1$ to 96% for $K = 2$ and we have 100% success with $K = 3$. There is little gained in increasing K from 2 to 3 which is important since there is a huge increase in complexity incurred by doing this, i.e., the number of nodes increases from 128 to 1093. In the second row of figure 1(b), for MMP:M-L, the percentage success increases from 66% to 86% to 93% as K goes from 1 to 3 to 5 and so this is slightly inferior to MMP:K, though the complexity of this search is much less than MMP:K.

The results obtained for ORMP in figure 1(c) are very similar to those of the MMP, showing only a very marginal improvement in both the case of the ORMP:K and ORMP:M-L algorithms.

5.2. Experiment 2:

This simulation is based on work in [7] and the analyzing dictionary is generated by filters for a class of wavelets called Symmlets so that the dictionary vectors are correlated. The input signal is termed "Carbon" and consists of a linear combination of elements from this dictionary: a Dirac, a sinusoid and 4 mutually orthogonal wavelet packet atoms.

Results

The ORMP:M-L algorithm was run for $K = 1, 3, 5$ and the best subset was obtained based on the smallest residual error. In figure 2(a), we plot the fall off in residual norm for this subset against the iteration number for each of the solutions obtained using the ORMP:M-L algorithm with $K = 1, 3, 5$. The best subsets for $K = 3$ and $K = 5$ coincide and the residual norm obtained after 9 elements are selected is negligible; it takes 11 elements to achieve this with $K = 1$. The ORMP:K algorithm with $K = 2$ also generates the same subset as ORMP:M-L with $K = 3$.

In the case of MMP:M-L, as shown in figure 2(b), there is a large reduction in the residual after 5 elements have been selected with $K = 3, 5$ while the residual is reduced to .0063 after 12 elements have been selected. In contrast, with $K = 1$, after 15 elements have been chosen, the residual still has a magnitude of .0421. Although, the scale makes it difficult to see, there is an improvement from .0019 to .0008 in the error obtained after 15 selections when $K = 5$ is used rather than $K = 3$. The performance is slightly inferior to that obtained using MMP:K with $K = 2$.

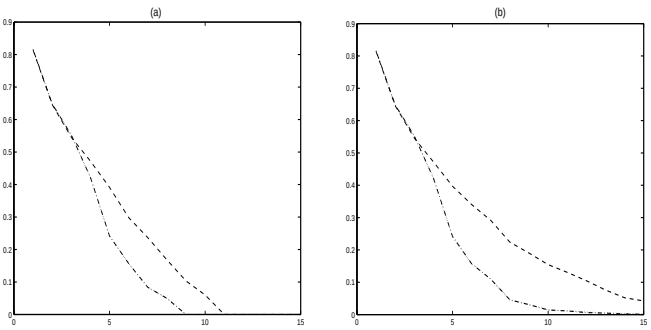

Fig. 2. Plot of residual norm with iteration number: (a) ORMP:M-L (b) MMP:M-L with $K = 1(--)$, $K = 3(\cdots)$ and $K = 5(-.-)$ ($K = 3$ and $K = 5$ coincide).

6. CONCLUSIONS

We have explored the use of more complex tree search techniques using MP and experimented with the MP:K and MP:M-L algorithms. Both these algorithms significantly improve upon the performance obtained using the regular MP algorithm. With the MP:K algorithm, we saw that $K = 2$ gives a large increase in performance with low cost compared to larger values of K. The performance of the MP:M-L algorithm almost matches that of MP:K and has a lower search complexity. In both algorithms, the increased complexity and storage required by using the ORMP makes the use of the MMP more attractive for this type of search.

7. REFERENCES

[1] F. Bergeaud and S. Mallat,"Matching pursuit: adaptive representations of images and sounds", *Computational and Applied Mathematics*, pp. 97-109, Jan. 1996.

[2] B.D. Rao, "Signal processing with the sparseness constraint", Proc. ICASSP '98, pp. 1861-4, May 1998.

[3] B. K. Natarajan,"Sparse approximate solutions to linear systems", *SIAM Journal on Computing*, pp. 227-234, April 1995.

[4] P.M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection", *IEEE Trans. on Computers*, pp. 917-926, Sept. 1977.

[5] C. Couvreur and Y. Bresler, "On the optimality of the backward greedy algorithm for the subset selection problem", *SIAM Journal on Matrix Analysis and Its Applications*, pp. 797-808, May, 1998.

[6] S.F. Cotter et al., "Backward sequential elimination for sparse vector subset selection", *Submitted to Signal Processing*, 2000.

[7] S.S. Chen et al., "Atomic decomposition by basis pursuit", *SIAM Journal on Scientific Computing*, pp. 33-61, Jan. 1998.

[8] B.D. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection", *IEEE Trans. on SP*, pp. 187-200, Jan. 1999.

[9] S. Cotter et al., "Forward sequential algorithms for best basis selection", *IEE Proc.*,pp. 235-244,Oct. 1999.

[10] J. Pearl, *Heuristics - Intelligent search strategies for computer problem solving*, Addison-Wesley, 1984.

[11] J.B. Anderson and S. Mohan, "Sequential coding algorithms: a survey and cost analysis", *IEEE Trans. Commun.*, pp. 1689-76, Feb. 1984.

[12] L.R. Rabiner and B-H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.

[13] N. Seshadri and J.B. Anderson, "Decoding of severely filtered modulation codes using the (M,L) algorithm", *IEEE JSAC*, pp. 1006-16, Aug. 1989.

[14] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries", *IEEE Trans. ASSP* , pp. 3397-3415, Dec. 1993.

[15] G. Davis, S. Mallat and Z. Zhang, "Adaptive time-frequency decompositions", *Optical Engineering*, pp. 2183-91, July 1994.