ALGORITHMS FOR COMPUTING SPARSE SOLUTIONS

Jan Špiřík
Doctoral Degree Programme (1), FEEC BUT
E-mail: jan.spirik@phd.feec.vutbr.cz

Supervised by: Pavel Rajmic
E-mail: rajmic@feec.vutbr.cz

Abstract:

The article deals with algorithms for computing the so-called sparse solutions of linear equations. This is a relatively novel approach to signal representation and it is very attractive, because it brings a wide spectrum of new applications (compressed sensing, fast MRI, radar etc.). However, in typical cases, the number of equations counts hundreds of thousands, and therefore, we need specialized algorithms for computing the sparse solutions. In this paper, such algorithms are divided into classes and their basic principles are described. The algorithms are compared from different viewpoints.

Keywords: sparse, pursuit algorithms, greedy algorithms, relaxation algorithms, hybrid algorithms

1 INTRODUCTION

Signal representation via underdetermined systems of linear equations is nowadays very popular and very efficient type of representation. These methods can be used in many branches of signal processing. The basics of sparse signal representations are introduced in [1]. These methods can be used for example for image compression, image denoising and deblurring, super-resolution imaging and impulsive noise removal [2]. All these methods start from the basic problem (P_0), which is defined as follows:

$$(P_0): \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x}, \tag{1}$$

where $\mathbf{y}$ is the known signal, we want to reconstruct, $\mathbf{D}$ is the dictionary which consists of atoms ("elementary signals") and $\mathbf{x}$ is an unknown sparse solution which represents amounts of each atom in the original signal. We define the norm of a vector:

$$\|\mathbf{x}\|_p := \left(\sum_{i=0}^{N} |\mathbf{x}_i|^p\right)^{1/p}, \quad 0 < p < \infty, \quad \|\mathbf{x}\|_0 := |\{j : \mathbf{x}_j \neq 0\}|. \tag{2}$$

If $0 < p < 1$ then is actually not the norm, it is the quasi-norm. The optimization problem (1) is defined in ℓ_0. When $\|\mathbf{x}\|_0 \ll n$ for $\mathbf{x}_0 \in \mathbb{R}^n$, we could say that $\mathbf{x}$ is sparse. The goal is to find that solution, which involves only a few of the dictionary elements. Vector $\mathbf{x}$ is composed of two parts – the support of the solution and the non-zero values over this support [2]. It is proved that ℓ_1 norm gives the sparse solution and finding the sparse solution without any algorithm in ℓ_0 norm is NP-hard problem. The problem can be redefined to:

$$(P_0): \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{D}\mathbf{x} - \mathbf{y}\|_2 \leq \varepsilon, \tag{3}$$

where ε is the error of solution. In ideal case is $\varepsilon = 0$. There are various algorithms dealing with this problem. We will present Pursuit algorithms. We divide them into 3 main classes: greedy, relaxation and hybrid algorithms.

2 GREEDY ALGORITHMS

Greedy algorithms for sparse approximation are iterative algorithms that include two basic steps and a criterion for suspension [3].

The first phase of the iteration process is called the greedy selection. The algorithm chooses that atom that best matches the signal in the current iteration. The presumption is that the locally optimal choice hopefully leads to globally optimal sparse approximation.

The second phase of the iteration process is called the greedy update. In this phase the algorithm applies the new atom and the current approximation to calculate a new approximation of the given signal.

After each iteration the algorithm decides whether to continue to the next iteration. The stopping rule could be the number of iteration or more commonly the error of the sparse approximation. The algorithm stops when the error $||\mathbf{Dx}-\mathbf{y}||_2$ is below the destination threshold.

2.1 MATCHING PURSUIT

Matching Pursuit (MP) is one of the oldest algorithms for solving sparse problem. It was developed by Mallat and Zhang in 1993. The method can be shown to converge to a solution, though convergence is often slow. Before the two steps are operated, we must initialize the algorithm. We set $\mathbf{x}=\mathbf{0}$, the residual we calculate will be $\mathbf{r}=\mathbf{y}$ and the initial solution support $\mathbf{S}=\mathit{Support}\,[\mathbf{x}]=\{\}$. In all others algorithms we assume the same basic conditions.

In greedy selection the algorithm selects that atom that best correlates with the residual signal. We could formulate as finding the maximum of vector $\mathbf{z}$ which is $\mathbf{z}=|\mathbf{D}^{\mathrm{T}}\mathbf{r}|$. We denote the location of the atom in the dictionary as z_{max}.

In greedy update is $\mathbf{x}$ updated in position of z_{max} as $\mathbf{x}(z_{max})=\mathbf{x}(z_{max})+\mathbf{D}^{\mathrm{T}}_{z_{max}}\mathbf{r}$, where $\mathbf{D}_{z_{max}}$ denotes the column z_{max} in the dictionary $\mathbf{D}$. We add z_{max} into $\mathbf{S}$ as well. Then the residual $\mathbf{r}$ is updated as $\mathbf{r}=\mathbf{r}-\mathbf{D}_{z_{max}}\mathbf{D}^{\mathrm{T}}_{z_{max}}\mathbf{r}$. After these procedures the algorithm decides if to stop or go to the next iteration.

This algorithm has some limits. The more the dictionary is not orthogonal, the more the probability of exact reconstruction decreases. If the algorithm chooses bad atom from some reasons in greedy selection step, the algorithm is not able to eliminate this fault. In addition, there are some dictionaries and signals, where MP is not optimal for the reconstruction of the sparse solution.

2.2 ORTHOGONAL MATCHING PURSUIT

Orthogonal Matching Pursuit (OMP) is a variant of MP. OMP adds a least-squares minimization at each step. In other words, OMP adds orthogonalization to MP. The greedy selection is the same as MP. The difference is in the greedy update.

In greedy update we compute $\mathbf{x}$ as the minimizer of $||\mathbf{Dx}-\mathbf{y}||_2$ subject to $\mathbf{S}$. Then we compute the residual $\mathbf{r}=\mathbf{y}-\mathbf{Dx}$. After this step we check the stopping rule.

When we implement this algorithm it is better to compute the greedy update in different way. First we sort the elements in $\mathbf{S}$. Then we compute the residual $\mathbf{r}=\mathbf{y}-\mathbf{D}_{z_{max}}\mathbf{D}^{+}_{z_{max}}\mathbf{y}$, where the operator $^{+}$ denotes the Moore-Penrose pseudoinverse. Then we check the stopping rule. When the stopping rule is reached, we calculate the sparse solution $\mathbf{x}$ as $\mathbf{x}_{SS}=\mathbf{D}^{+}_{SS}\mathbf{y}$. $\mathbf{x}_{SS}$ denotes that elements in vector $\mathbf{x}$ with support accordance with $\mathbf{S}$. $\mathbf{D}_{SS}$ denotes the columns in the dictionary accordance with $\mathbf{S}$.

OMP is nowadays probably the most used algorithm for finding the sparse solution for its speed of convergence and computational efficiency.

2.3 LEAST-SQUARES-OMP

A more complex and slightly better behaving variant of OMP is Least-Squares-OMP (LS-OMP). There are each of the tests is done as full Least-Squares.

At first we evaluate the error of reconstruction according to the support. We will express that as $\min_{\mathbf{x}} ||\mathbf{D}\mathbf{x} - \mathbf{y}||$ subject to $\mathbf{S}$. Then we choose the atoms as above and update the support $\mathbf{S}$. At last we update the residual $\mathbf{r} = \mathbf{y} - \mathbf{D}\mathbf{x}$. These are the main differences compared to OMP.

When we implement this algorithm we must write the first step in the loop. This could be the problem in some programming languages. In the loop we compute residual $\mathbf{r}$ k-times, where k is the number of atoms in the dictionary. Residual $\mathbf{r}$ is computing in each loop as $\mathbf{r} = \mathbf{y} - \mathbf{D}_{act}\mathbf{D}_{act}^{+}\mathbf{y}$, where $\mathbf{D}_{act}$ is the atom in the dictionary accordance with $\mathbf{S}$. After these procedures we find the minimal residual from the loop. Then we insert the position of residual to $\mathbf{S}$. Finally we check the stopping rule. The sparse solution $\mathbf{x}$ is calculated after all iteration in the same way as OMP.

2.4 WEAK MATCHING PURSUIT

The Weak Matching Pursuit (Weak-MP) is simplification of the MP. It allows a suboptimal choice of the next element to be added to the support. We can choose any index that is factor t (in the range $[0,1]$) away from the optimal choice. The greedy update step is the same as MP. So I describe only the greedy selection step.

In the greedy selection step we choose the factor t. When we define $\mathbf{z}$ as $\mathbf{z} = |\mathbf{D}^{\mathrm{T}}\mathbf{r}|$, then we choose only the elements which are larger than $t\sqrt{\mathbf{r}^{\mathrm{T}}\mathbf{r}}$. The we choose the maximum value and mark the location as z_{max}. This way, the chosen index clearly points to a column that is at the most factor t away from the possible maximum.

If we use this algorithm for searching the sparse solution could become much faster. But we must understand that the rate of decay of the residual has been somewhat compromised [2].

2.5 THRESHOLDING ALGORITHM

Thresholding Algorithm is another method than the MP-family. It is much more simpler than the previous ones. The main idea is to choose the n largest inner products as the expected support.

First we compute vector $\mathbf{a} = \mathbf{D}^{\mathrm{T}}\mathbf{y}$ and sort this vector by absolute descending order. Then in each iteration we add the next element from $\mathbf{a}$ to the support $\mathbf{S}$. In each iteration we compute the sparse solution $\mathbf{x} = D_{act}^{+}\mathbf{y}$ and residual $\mathbf{r} = \mathbf{y} - D_{act}\mathbf{x}_{act}$. This iteration stops when the stopping rule is reached.

3 RELAXATION ALGORITHMS

The main principle of relaxation algorithms is to relax the ℓ_0 norm. There could be more approaches to the relaxation algorithms. One of this approaches could be to smooth the ℓ_0 norm and replace it with a continuous function that can be handled using classic optimization. One of the most succesful approach is to substitute the ℓ_0 norm for different norm ℓ_p with $p \in (0,1]$.

Except the introduced methods exists many more interesting algorithms. They could solve a different optimization problem to achieve the sparse solution, for example the Danzig Selector algorithm that optimize $\min_{\mathbf{x}} ||\mathbf{x}||_1$ subject to $||\mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{x} - \mathbf{y})||_\infty \leq T$.

3.1 Basis Pursuit

Basis Pursuit (BP) is a method by Chen, Donoho and Saunders, which was published in 1995. The main principle is to solve (P_1) problem instead of (P_0) [4]. The main optimization should be rewrite as $\min_{\mathbf{x}} \|\mathbf{x}\|_1$ subject to $\|\mathbf{Dx}-\mathbf{y}\|_2 \leq \varepsilon$. Now it is the convex optimization problem. For the case where $\varepsilon = 0$, this becomes a linear programming problem. We could simply rewrite the (P_1) to linear programming (LP_1) as follows:

$$(LP_1): \quad \min_{\widetilde{\mathbf{x}}} \mathbf{1}^{\mathrm{T}}\widetilde{\mathbf{x}} \quad \text{subject to} \quad \widetilde{\mathbf{D}}\widetilde{\mathbf{x}} = \mathbf{y}, \quad \widetilde{\mathbf{x}} \geq 0, \tag{4}$$

where $\widetilde{\mathbf{D}} = [\mathbf{D} \ -\mathbf{D}]$, $\widetilde{\mathbf{x}} = [\mathbf{x}_+^{\mathrm{T}} \ \mathbf{x}_-^{\mathrm{T}}]$. There are known many algorithms to solve this problem (simplex method, Bland method, Dantzig method) [4].

3.2 Iterative Reweighted Least-Squares

The next example of relaxation algorithms is Iterative Reweighted Least-Squares (IRLS) algorithm by Gorodnitsky and Rao. This method is also called FOCUSS (FOcal Underdetermined System Solver). This method uses representation of ℓ_p norm (for some fixed $p \in (0,1]$) as a weighted ℓ_2 norm. It is an iterative algorithm as well. This algorithm has more definition. At first we initiate an approximation $\mathbf{x} = \mathbf{1}$ and the initial weight matrix $\mathbf{X} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. In the main iteration we firstly solve the linear system $\mathbf{x} = \mathbf{X}^2\mathbf{D}^{\mathrm{T}}\left(\mathbf{D}\mathbf{X}^2\mathbf{D}^{\mathrm{T}}\right)^+\mathbf{y}$ either directly or iteratively. The second and the last step in the iteration is to update the diagonal weight of the matrix $\mathbf{X}$ using $\mathbf{x} : \mathbf{X}(j,j) = |\mathbf{x}(j)|^{1-p/2}$. We stop the iteration, when the stopping rule is reached. The rule is most frequently the difference between the vector $\mathbf{x}$ in actual and previous iteration in ℓ_2 norm.

4 HYBRID ALGORITHMS

The hybrid algorithms are these ones that connect the relaxation a greedy methods. There are many representatives of this category. We mention for example three of these. **Subspace-Pursuit** (SP) is constructed on the principal of finding L-sparse approximate solution to the problem $\min_{\mathbf{x}} \|\mathbf{Dx}-\mathbf{y}\|_2$ subject to $\|\mathbf{x}\|_0 = L$. **The CoSaMP** is similar to SP, but differs in two ways. It works with $3L$ support and has only one LS step. The third one is **Iterative Hard Thresholding** (IHT) that may look similar to SP and CoSaMP, but it is closer to different types of algorithms called iterative shrinkage. All these algorithms are relatively new (SP – 2009, CoSaMP – 2009) and are still waiting for next improvements.

5 COMPARISION OF DIFFERENT ALGORITHMS

The greedy and relaxation algorithms were compared in different ways. All the simulations were running in Mathworks Matlab R2010a on the Intel(R) Core(TM) i5 CPU 680 @ 3.60 GHz with 4 GB RAM. The stopping rule was set to the error of $\varepsilon = 1 \cdot \exp^{-4}$ and there were used 1000 examples of each cardinality.

In Figure 1 is shown the performance based on the relative ℓ_2 recovery error. Best algorithms in this point of view are the relaxation ones. From the greedy is the best method LS-OMP, followed by OMP. In Figure 2 is shown the performance based on the success rate in detecting the true support. The result is very similar as in the first figure. Almost all algorithms give good results for low cardinalities and relaxation algorithms are better in higher cardinalities. In Figure 3 we could see dependency on computation time. The results could be a little inaccurate because Matlab is not optimized for inner loops. The greedy algorithms are much more quicker than relaxation algorithms. The relaxation algorithms are many times slower.

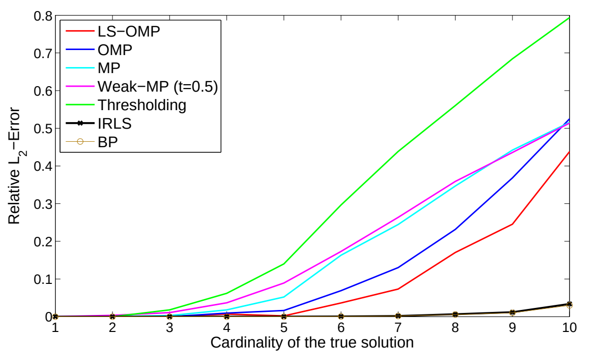

Figure 1: Performance based on the relative ℓ_2 recovery error

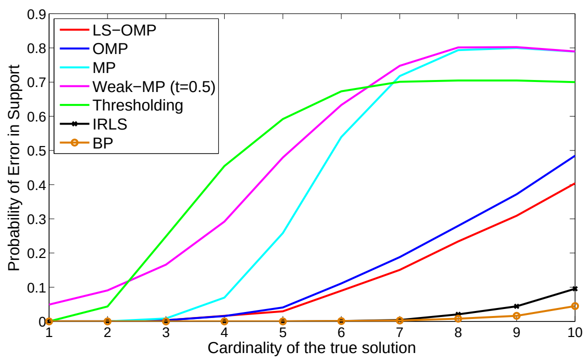

Figure 2: Performance based on the success rate in detecting the true support

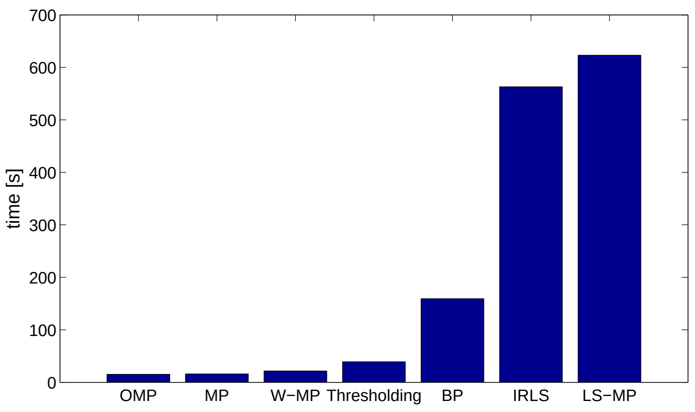

Figure 3: Performance based on computation time

6 CONCLUSION

There were introduced three basic classes of pursuit algorithms for computing sparse solutions of linear equations. In each class were mentioned algorithms and its basic principles. They were compared in different ways. From the figures we could see that the OMP is the compromise between the computation time, speed of convergence and probability of true solution. OMP is very often used in applications based on underdetermined systems nowadays. If we request the true solution with higher probability, we should use the BP or IRLS. But these algorithms take more time to achieve the sparse solution.

REFERENCES

[1] O. Christensen, *Frames and Bases, An Introductory Course*. Boston: Birkhäuser, 2008.

[2] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010.

[3] J. A. Tropp, "Average-case analysis of greedy pursuit," *Proc. SPIE Wavelets XI, pp. 590401.01-11, San Diego*, 2005.

[4] Y. Tsaig, "Sparse solution of underdetermined linear systems: algorithms and applications," Master's thesis, Stanford university, 2007.