MORE EFFICIENT SPARSITY-INDUCING ALGORITHMS USING INEXACT GRADIENT

Alain Rakotomamonjy[*,†], *Sokol Koço*[*], *Liva Ralaivola*[†]

[*] Université de Rouen, LITIS EA4108
[†] Aix-Marseille Université, CNRS UMR 7279 LIF

ABSTRACT

In this paper, we tackle the problem of adapting a set of classic sparsity-inducing methods to cases when the gradient of the objective function is either difficult or very expensive to compute. Our contributions are two-fold: first, we propose methodologies for computing fair estimations of inexact gradients, second we propose novel stopping criteria for computing these gradients. For each contribution we provide theoretical backgrounds and justifications. In the experimental part, we study the impact of the proposed methods for two well-known algorithms, Frank-Wolfe and Orthogonal Matching Pursuit. Results on toy datasets show that inexact gradients can be as useful as exact ones provided the appropriate stopping criterion is used.

Index Terms— sparse learning, greedy algorithms, inexact gradient, randomization

1. INTRODUCTION

Over the last few years, there has been an increasing interest in inference problems featuring data of very high-dimension and few number of observations. Such problems occur in a wide variety of application domains ranging from computational biology and text mining to information retrieval and finance. In order to extract knowledge from these datasets, a large amount of research from the machine learning, statistics and signal processing communities has been devoted to developing statistical models featuring some sparsity properties, such as models that use only few of the data's dimensions. To obtain these models, one typically needs to solve:

$$\min_{\mathbf{w}} L(\mathbf{w}) \text{ subject to } \|\mathbf{w}\|_0 \leq K \tag{1}$$

where L is a function that measures the goodness-of-fit of the model, $\|\mathbf{w}\|_0$ is the ℓ_0 pseudo-norm of the vector $\mathbf{w}$ and K is a parameter that controls the sparsity level.

A common approach to solve (1) is to make use of greedy methods that provide a (possibly approximate) solution. A flurry of algorithms have been proposed to this end, the most popular ones being *Matching Pursuit* (MP), *Orthogonal Matching Pursuit* (OMP) [1, 2] and their descendants, most of which are in the family of the so-called *Gradient Pursuit* algorithms [3]. Another way for escaping the combinatorial nature of the problem induced by the ℓ_0 pseudo-norm is to have it replaced by ℓ_1-norm, a convex and continuous surrogate of ℓ_0: this transforms (1) into the well-known Lasso problem. There also exists a wide variety of algorithms for its resolution ranging from homotopy methods [4] to the Frank-Wolfe (FW) algorithm [5].

A common point of the aforementioned approaches is that they are iterative methods requiring, at each iteration, the computation of the gradient of the objective function L. For large-scale and high-dimensional settings, computing the gradient at each iteration may be very time-consuming.

In this work, we address the problem of accelerating some sparsity-inducing algorithms such as gradient pursuit methods and the Frank-Wolfe algorithm with an ℓ_1 ball constraint, by exploiting inexact gradient information. Our contributions come in the form of strategies that make use of inexact gradients. These approaches exploit a specific property of sparsity-inducing algorithms: at each iteration, they seek the component of the gradient with the largest (absolute) value.

2. SPARSE LEARNING ALGORITHM WITH EXTREME GRADIENT COMPONENT

2.1. Framework

Let us consider a problem where we want to estimate a relation between a set of n samples gathered in a vector $\mathbf{y} \in \mathbb{R}^n$ and the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. In a sparse signal approximation problem, the columns of $\mathbf{X}$ are the elements of a dictionary and $\mathbf{y}$ the target signal, while in a machine learning problem, the i-th row $\mathbf{x}_i$ contains the features of the i-th example and y_i is its label. Our goal is to learn the relation between $\mathbf{y}$ and $\mathbf{X}$ through a linear model of the data denoted as $\mathbf{X}\mathbf{w}$ by looking for the vector $\mathbf{w}$ that solves problem (1) when:

$$L(\mathbf{w}) = \sum_{i=1}^{n} \ell(y_i, g(\mathbf{w}^\top \mathbf{x}_i)). \tag{2}$$

For instance, L might be the least-square error

$$L(\mathbf{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2,$$

This work is supported by Agence Nationale de la Recherche, project GRETA 12-BS02-004-01.

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

Algorithm 1 Frank-Wolfe Algorithm

set k = 0, initialize $\mathbf{w}_0$
for k=0,1, $\cdots$ **do**
  $\mathbf{s}_k = \arg\min_{\mathbf{s}\in C} \mathbf{s}^\top \nabla L(\mathbf{w}_k)$
  $\mathbf{d}_k = \mathbf{s}_k - \mathbf{w}_k$
  set, linesearch or optimize $\gamma_k \in [0; 1]$
  $\mathbf{w}_{k+1} = \mathbf{w}_k + \gamma_k \mathbf{d}_k$
end for

or the logistic loss

$$L_{log}(\mathbf{w}) = \sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{x}_i^\top \mathbf{w}}).$$

2.2. Algorithms for Solving Problem (1)

We briefly recall the core of the methods we consider as best candidates to take advantage of our acceleration procedure.

Sparse learning with the Frank-Wolfe Algorithm (FW). The FW algorithm (Algorithm 1), is a simple procedure that provides a solution for:

$$\min_{\mathbf{w}\in C} L(\mathbf{w}), \tag{3}$$

where C is usually a compact subset of $\mathbb{R}^d$ and L is a convex and differentiable function, by iteratively looking for a search direction and updating the current iterate. The search direction $\mathbf{s}_k$ is obtained from the following convex optimizaton problem:

$$\mathbf{s}_k = \arg\min_{\mathbf{s}\in C} \mathbf{s}^\top \nabla L(\mathbf{w}_k), \tag{4}$$

which may (or not) be efficiently solved, depending on the constraint set C — for achieving sparsity, we typically choose C as a ℓ_1-norm ball, which turns eq. (4) into a linear program. While conceptually simple, this algorithm has been shown to be linearly convergent [5, 6]. Interestingly, it can also be shown that, if an inexact gradient information $\hat{\nabla} L(\mathbf{w}_k)$ is available, convergence is still guaranteed under the condition:

$$\mathbf{s}_k^\top \hat{\nabla} L(\mathbf{w}_k) \leq \min_{\mathbf{s}\in C} \mathbf{s}^\top \nabla L(\mathbf{w}_k) + \epsilon',$$

and some closeness condition between $\hat{\nabla} L(\mathbf{w}_k)$ and $\nabla L(\mathbf{w}_k)$. This condition comes in handy when the minimization problem (4) is expensive to solve. For instance, if C is an unit-norm ball associated with some norm $\|\cdot\|$ and $\mathbf{s}_k$ a minimizer of $\min_{\mathbf{s}\in C} \mathbf{s}^\top \hat{\nabla} L(\mathbf{w}_k)$, then it suffices that:

$$\|\hat{\nabla} L(\mathbf{x}_k) - \nabla L(\mathbf{x}_k)\|_\star \leq \epsilon', \tag{5}$$

where $\|\cdot\|_\star$ is the conjugate norm associated with $\|\cdot\|$, to ensure convergence [5]. However, this condition is not useful in practice, since it depends on the exact gradient.

Gradient Pursuit. This set of methods, originally introduced in [3], is constituted of greedy strategies exhibiting the effectiveness of OMP with the small computational requirements of MP, and both MP and OMP might be seen as members of this family. These methods are specifically designed to provide a solution for problem (1), when L is the least square error L_{LS} — note however that they can be extended to carry over to more general losses such as the one stated in (2). The essence of these pursuit methods is to iteratively refine the parameter vector $\mathbf{w}$ by performing updates of the form

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \mathbf{d},$$

where $\mathbf{d}$ is some *descent direction* and α an appropriate step size. At each iteration, the computation of $\mathbf{d}$ hinges on the descent directions computed so far and on the gradient of L; more precisely, it requires the index of the gradient component with the largest absolute value.

2.3. Extreme Gradient Component

As stated above, the gradient pursuit algorithm needs to find at each iteration, the largest absolute component of the gradient:

$$i^\star = \arg\max_i |\nabla L(\mathbf{w}_k)|_i.$$

Similar situations arise in the FW algorithm, for instance, when the set C is the ℓ_1-norm ball or the simplex constraint, and in addition the components of $\mathbf{w}$ need to be positive. The search direction is then given by:

$$\arg\min_{\mathbf{s}\in C} \mathbf{s}^\top \nabla L(\mathbf{w}_k) = \arg\min_i [\nabla L(\mathbf{w}_k)|]_i$$

Hence, in both cases (as well as in MP and OMP), at each iteration, we are neither interested in the gradient, nor in the extreme values, but only in the coordinate of the component with the smallest or the largest absolute value.

Based on these observations, our goal is to propose methods that efficiently compute the extreme component of the gradient. In particular, in situations where the gradient is expensive to compute, we aim at providing algorithms that produce cheaper approximated gradient for which the extreme component of interest is the same as the exact gradient one.

3. LOOKING FOR THE EXTREME GRADIENT COMPONENT

3.1. Statement of the Problem

The gradient of the objective function (2) is of the form:

$$\nabla L(\mathbf{w}) = \sum_i \ell'(y_i, g(\mathbf{w}^\top \mathbf{x}_i)) g'(\mathbf{w}^\top \mathbf{x}_i) \mathbf{x}_i = \mathbf{X}^\top \mathbf{r},$$

where $\mathbf{r} = (\ell'(y_i, g(\mathbf{w}^\top \mathbf{x}_i))) g'(\mathbf{w}^\top \mathbf{x}_i))_{i\in[n]} \in \mathbb{R}^n$. This particular form implies that the gradient can be computed iteratively, obtaining at each iteration an approximate gradient

$\hat{\nabla} L$. This means that for a given iteration t, we have:

$$\hat{\nabla} L_t = \hat{\nabla} L_{t-1} + U_t \tag{6}$$

where U_t is typically $r_j \mathbf{x}_j^\top$ for a given j.

Building on this observation, our goal is to devise efficient ways of approximating the gradient so that the top entry of this approximation is also the top entry of the exact gradient.

3.2. A Deterministic Approach

Let $\mathcal{I}_t$ denote the index set of the examples already chosen in the first t iterations for computing $\hat{\nabla} L_t$. At $t+1$, our goal is to find the example $j^\star$ such that:

$$j^\star = \underset{j \in \{1,\dots,n\} \setminus \mathcal{I}_t}{\operatorname{argmin}} \|\nabla L - \hat{\nabla} L_t - \mathbf{x}_j r_j\| \tag{7}$$

where by definition $\hat{\nabla} L_{t+1} = \hat{\nabla} L_t + \mathbf{x}_{j^\star} r_{j^\star}$. However, since ∇L is not accessible, the solution of this problem cannot be computed. Hence, we have to resort to an approximation:

$$j^\star = \underset{j \in \{1,\dots,n\} \setminus \mathcal{I}_t}{\operatorname{argmax}} \|\nabla L - \hat{\nabla} L_t\| - \|\nabla L - \hat{\nabla} L_{t+1}\| \tag{8}$$

By upper-bounding the objective value in (8), one can derive the best choice of $r_j \mathbf{x}_j$ that leads to the largest variation of the gradient estimation norm residual. Indeed, we have:

$$\begin{aligned} \|\nabla L - \hat{\nabla} L_t\| - \|\nabla L - \hat{\nabla} L_{t+1}\| &\leq \|\nabla L - \hat{\nabla} L_t - \nabla L + \hat{\nabla} L_{t+1}\| \\ &= \|\mathbf{x}_j^\top r_j\| = \|\mathbf{x}_j\| |r_j| \end{aligned} \tag{9}$$

This suggests that the selected index j should be the one with the largest absolute residual $\|\mathbf{x}_j\| |r_j|$. In the first iteration, the algorithm chooses the index that leads to the largest value of eq. (9), in the second one, it selects the second best, and so forth: the selection boils down to considering the examples in a decreasing order of absolute residuals.

Note that for this method, the exact gradient is recovered when $t = n$. We are assured to retrieve the correct extreme entry of the gradient, alas at the expense of extra computations needed for evaluating (useless) stopping criteria. If the procedure stops at $t < n$, we gain in efficiency at the risk of missing the correct extreme entry.

3.3. A Randomized Approach

The problem of finding the gradient's extreme component can also be addressed from the point of view of randomization.

The approach consists in considering the computation of $\mathbf{X}^\top \mathbf{r}$ as an the expectation of a given random variable. Recall that $\mathbf{X}$ is composed of the vectors $\{\mathbf{x}_i^\top\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$. Hence, the matrix-vector product $\mathbf{X}^\top \mathbf{r}$ can be rewritten: $\mathbf{X}^\top \mathbf{r} = \sum_{j=1}^n r_j \mathbf{x}_j$. Let Δ_n^* denote the interior of the probabilistic simplex of size n. For any element $\mathbf{p} = (p_j)_{j \in [n]} \in \Delta_n^*$, we introduce a random vector C that takes value in the finite set

$$\mathcal{C} \doteq \{c_j \doteq r_j \mathbf{x}_j / p_j : j = 1, \dots, n\}, \tag{10}$$

so that $\mathbb{P}(C = c_j) = p_j$. This way,

$$\mathbb{E}C = \sum_{j=1}^n p_j c_j = \sum_{j=1}^n \frac{p_j r_j \mathbf{x}_j}{p_j} = \sum_{j=1}^n r_j \mathbf{x}_j = \mathbf{X}^\top \mathbf{r}. \tag{11}$$

If $C_1, \cdots, C_s$ are independent copies of C and $\hat{C}^s$ is their mean then

$$\mathbb{E}\hat{C}^s = \mathbf{X}^\top \mathbf{r}, \tag{12}$$

thus $\hat{C}^s$ is an estimator of the matrix-vector product that we are interested in *i.e* the gradient of our objective function.

Hence, according to the above, a nice approach for estimating the extreme component of the gradient is to randomly draw s copies of C, to compute their average and then to look for the extreme component of this estimated gradient. Interestingly, this approach based on randomized matrix multiplication can be related to our deterministic approach. Indeed, a result given by [7] (Lemma 4), says that the element $\mathbf{p} \in \Delta_n^*$ that minimizes the variance of $\hat{C}^s$ is such that

$$p_j \propto |r_j| \|\mathbf{x}_j\|. \tag{13}$$

It thus implies that vectors of large $|r_j| \|\mathbf{x}_j\|$ have higher probability to be sampled.

3.4. Stopping Criteria

In the following we present two strategies for determining the number of elements needed in the previous methods: one that holds for any cases and the second one that holds only for the Frank-Wolfe algorithm in the deterministic sampling case.

Stability condition Let $j^\star$ denote the coordinate *s.t.* $s_j^\star = \arg\min_j \nabla L(\mathbf{x}_k)|_j$ and T the maximum number of iterations. Our objective is to estimate $j^\star$ with the fewest number t of iterations as possible, under the following hypothesis:

$$\exists t_1 \leq T, \forall t : t_1 \leq t \leq n \quad j^\star = \arg\min_j \hat{\nabla} L_t(\mathbf{x}_k)|_j.$$

Formally, this condition means that $\forall t \geq t_1$, we have:

$$\left[\hat{\nabla} L_t + \sum_{i=0}^{T-t} U_{t+i}\right]_{j^\star} \leq \left[\hat{\nabla} L_t + \sum_{i=0}^{T-t} U_{t+i}\right]_j \quad \forall j \in [1, \cdots, d]$$

However, checking the above condition is as expensive as computing the full gradient, thus we propose an estimation of $j^\star$ based on an approximation of this inequality, by truncating the sum to few iterations on each side. This consists in checking whether the index $j^\star$ has changed over the last N_s iterations. We refer to this as the *stability criterion*.

Error bound criterion In Section 2.2, equation (5) ties the convergence of the inexact FW algorithm to the ability of upper-bounding the difference between the approximate gradient and the exact one, i.e. the method can be stopped as soon

as the difference is smaller than the quantity ϵ'. In practice, this criterion cannot be computed as it depends on the computation of the exact gradient but it can be upper-bounded by norm equivalence and simple algebras:

$$\|\hat{\nabla} L_t - \nabla L\|_\infty \leq \|\hat{\nabla} L_t - \nabla L\|_1 = \sum_i \Big| \sum_{j \notin \mathcal{I}_t} \mathbf{x}_j r_j \Big|_i$$
$$\leq \sum_i \sum_{j \notin \mathcal{I}_t} |\mathbf{x}_j|_i |r_j| = \sum_{j \notin \mathcal{I}_t} \|\mathbf{x}_j\|_1 |r_j|.$$

This leads to the following criterion $\sum_{j \notin \mathcal{I}_t} \|\mathbf{x}_j\|_1 |r_j| \leq \epsilon'$. Note that this criterion can be easily checked at each gradient update, provided the norms $\|\mathbf{x}_j\|_1$ are computed beforehand.

4. EXPERIMENTS

4.1. Experimental setting

In order to illustrate the benefit of using inexact gradient for sparse learning, we have designed several experiments for which a sparse signal has to be recovered either by means of FW or OMP. The target sparse signals have been built as follows. For a given size of the dictionary d and a number k of active elements in the dictionary, the k non- zero positions of the true coefficient vector $\mathbf{w}^\star$ are chosen randomly and their values are drawn from zero-mean unit variance Gaussian distribution, from which we added ± 0.1 according to the sign of the values. The columns of the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ are drawn uniformly from the surface of a unit hypersphere of dimension n. Finally, the target vector is obtained as:

$$\mathbf{y} = \mathbf{X}\mathbf{w}^\star + \mathbf{e}$$

where $\mathbf{e}$ is a noise vector drawn from Gaussian distribution with zero-mean and variance σ_e^2 determined from a given Signal-To-Noise ratio as

$$\sigma_e^2 = \frac{1}{n} \|\mathbf{X}\mathbf{w}^\star\|^2 \cdot 10^{-SNR/10}.$$

Unless specified, the SNR ratio has been set to 3. For each setting, the results are averaged over 20 trials ($\mathbf{X}$, $\mathbf{w}^\star$ and $\mathbf{e}$ are resampled at each trial).

The criteria used for evaluating and comparing the proposed approaches are the running time of the algorithms and their ability to recover the true non-zero elements of $\mathbf{w}^\star$. The latter is estimated through the F-measure between the support of $\mathbf{w}^\star$ and its estimation $\hat{\mathbf{w}}$:

$$F - meas = 2 \frac{|\text{supp}(\mathbf{w}^\star) \cup \text{supp}(\hat{\mathbf{w}})|}{|\text{supp}(\mathbf{w}^\star)| + |\text{supp}(\hat{\mathbf{w}})|},$$

where, $\text{supp}(\mathbf{w}) = \{j : w_j > \gamma\}$ is the support of vector $\mathbf{w}$ and γ is a threshold (set to 0.001) used to neglect some non-zero coefficients that may be obliterate by the noise.

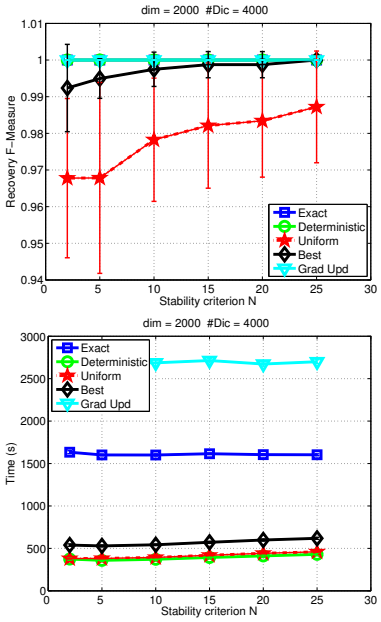

Fig. 1. Performances are compared with increasing precision on the inexact gradient computation.

4.2. Sparse learning using a Frank-Wolfe algorithm

For this experiment, the constraint set C is the ℓ_1 unit-ball and the loss function is $L(\mathbf{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$. Our objectives are:

- analyze the capability of inexact gradient approaches to recover the true support and compare them to the exact gradient of FW,
- compare the two stopping criteria of Section 3.4.

The exact gradient is computed using the accumulation strategy of eq. (6) so as to make all running times comparable. The maximum number of iteration for FW is set to 5000.

Figure 1 shows the results obtained for $n = 2000$, $d = 4000$ and $k = 50$. We present the running time and recovery abilities of the FW algorithm with an exact gradient (**exact**), a deterministic gradient sampling with a stability stopping criterion (**deterministic**), same with an error bound stopping criterion (**grad upb**), a uniform random sampling (**uniform**) and a best probability sampling (**best**). The figures depict the performances with respect to stopping condition parameter N_s of the stability criterion.

The deterministic approach used with any stopping criterion perfectly recovers the exact support of $\mathbf{w}^\star$, while both randomized approaches achieve an average F-measure of 0.975 with $N_s = 5$. As expected, when N_s increases, the performances of these two approaches increase also.

From a computational running-time point of view, the best results are obtained by the approaches based on deterministic and randomized sampling strategy with stability criterion, achieving a gain up to a factor 4 with respect to the exact Frank-Wolfe algorithm. Interestingly, this comes with no

compromise on the recovery performance for the deterministic approach. For the randomized strategies, increasing the parameter N_s leads to a slight increase of running time, hence for these methods, a trade-off can eventually be found.

When comparing the *stability* and the *error bound* stopping criterion, the latter seems rather inefficient. While based on theoretical analysis, the bound is loose enough to be non-informative. Indeed, a careful inspection shows that most inexact gradient computations use about 98% of the residuals. In addition, extra computations needed for the bound estimation, makes the approach less efficient than the exact FW.

In summary, from this experiment, we can conclude that the deterministic and randomized sampling strategies are the most efficient. However, they should be used in conjunction with the *stability* stopping criterion.

4.3. Sparse Approximation with OMP

Here, we evaluate the effect of using an inexact gradient in a greedy algorithm framework like OMP. The toy problem is similar to the one used above except that we analyze the performance of the algorithm for an increasing number k of active atoms. The inexact gradient methods are evaluated and compared in terms of efficiency and correctness to the true gradient in the OMP algorithm.

For OMP, the stopping criterion is based on a fixed number of iteration corresponding to the desired sparsity k. For all sampling approaches, the stopping criterion for gradient accumulation is based on the stability criterion, and the parameter N_S adaptively set at 1% of the dictionary size.

Results are reported in Figure 2. In terms of support recovery, when the number of active atoms is small, it seems that the deterministic approach performs better than the randomized sampling strategy. Note that for any value of k, the randomized strategies suffer more than the exact and deterministic sampling strategies for recovering the true vector $\mathbf{w}^\star$ support. From a running-time point of view, again, we note that sampling strategies are more efficient than the exact approach, with a slight advantage to the deterministic one.

5. CONCLUSION

In this paper, we introduced novel methodologies for speeding up sparsity-inducing algorithms, by making use of an inexact (but fair) estimation of the gradient. Two stopping criteria adapted to these methodologies were also introduced. For all the contributions, we provided some theoretical justification. The experimental results are quite encouraging as they strongly suggest that coupling inexact gradient methods with the stability condition yields similar results as the exact gradient one, with a significant gain in the running-time. Future works will focus on improving the error bound criterion, since, in its current form, it does not perform well. Other works will concern the adaptation of successive reject bandit strategies to extreme gradient component estimation.

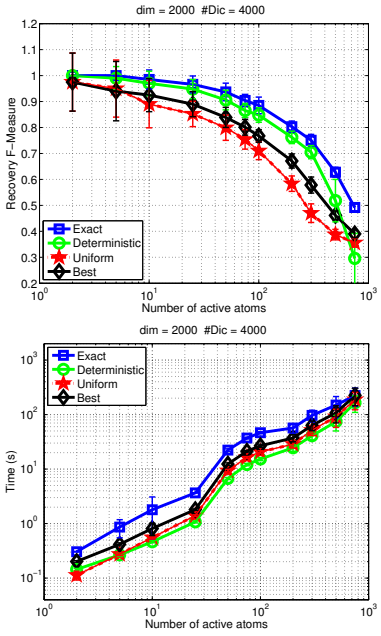

Fig. 2. Comparing OMP with different ways for computing the inexact gradient, with $n = 2000$ and $d = 4000$.

REFERENCES

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

[2] Y.C. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit : Recursive function approximation with applications to wavelet decomposition," in *Proc. of the 27th Ann. Asilomar Conf. on Signals, Syst. and Comp.*, 1993.

[3] T. Blumensath and M. E. Davies, "Gradient pursuits," *IEEE Trans. on Signal Processing*, vol. 56, no. 6, 2007.

[4] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression (with discussion)," *Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[5] M. Jaggi, "Revisiting frank-wolfe : Projection free sparse convex optimization," in *Proceedings of the International Conference on Machine Learning*, 2013.

[6] J. Guélat and P. Marcotte, "Some comments on wolfe's away step," *Math. Programming*, vol. 35, no. 1, 1986.

[7] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication," *SIAM Journal on Computing*, vol. 36, no. 1, pp. 132–157, Jan. 2006.