Greedy Optimization and Applications to Structured Tensor Factorizations

Master Thesis
Francesco Locatello
September 07, 2016

Advisors: Prof. Dr. Martin Jaggi, Rajiv Khanna, Michael Tschannen, Prof. Dr. Gunnar Rätsch, Prof. Dr. Thomas Hofmann
Department of Computer Science, ETH Zürich
Abstract

Efficiently representing real world data in a succinct and parsimonious manner is of central importance in many fields. We present a generalized greedy framework, which allow us to efficiently solve generalized structured matrix and tensor factorization problems, where the factors are allowed to be from arbitrary sets of structured vectors. Such structure may include sparsity, non-negativeness, order, or a combination thereof. The algorithm approximates a given $N$-dimensional tensor by a linear combination of few rank-1 tensors, each factorized into an outer product of $N$ vector atoms of the desired structure. We prove strong convergence rates for variants of the algorithm in Hilbert spaces, which generalize orthogonal matching pursuit and are useful beyond tensor problems. Our $O(1/t)$ convergence rate for general smooth convex functions is the first for this level of generality. In the case of general smooth and strongly convex functions over the linear span of arbitrary dictionaries, we prove faster $O(\rho^t)$ linear rates which hold for any structural constraints on the factors. We also comment on the tightness of this result in some cases using a lower bound on the error exponential decay. We finally give an overview of greedy optimization algorithms such as Frank-Wolfe, Matching Pursuit and Boosting, analyzing their similarities and differences from a theoretical perspective. Further, for the non-convex subproblems of obtaining good rank-1 structured matrix atoms, we employ and analyze a general atomic power method.
Contents

1 Introduction 1
   1.1 Introduction .............................................. 1

2 Notation 5
   2.1 Notation .................................................. 5

3 Frank-Wolfe 7
   3.1 Frank-Wolfe Algorithm .................................... 7
      3.1.1 Inexact Oracle ....................................... 8
      3.1.2 Convergence .......................................... 8

4 Matching Pursuit 11
   4.1 Matching Pursuit Algorithm ............................. 11
      4.1.1 Inexact Oracle ....................................... 12
      4.1.2 Convergence .......................................... 12

5 General Greedy 15
   5.1 Greedy Algorithms in Hilbert Spaces ................. 15
      5.1.1 Inexact Linear Oracles and Atom Corrections .... 18
      5.1.2 Sublinear Convergence Rates ....................... 19
      5.1.3 Linear Convergence Rates .......................... 21
      5.1.4 On the Relationship Between Matching Pursuit and Frank Wolfe .................................................. 26
      5.1.5 Boosting and Leveraging ............................. 28

6 Background on Tensors Algebra 31
   6.1 Introduction and Notation ............................. 31
   6.2 Dual Vector Space and Rank ............................ 32
      6.2.1 Matrices: Vectorization ............................ 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.2</td>
<td>Tensors: Flattening</td>
<td>33</td>
</tr>
<tr>
<td>6.3</td>
<td>More on Tensor Rank and CP decomposition</td>
<td>34</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Comments on Tensor Problems</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>A Generalized Greedy Algorithm for Structured Non-Convex Matrix and Tensor Factorizations</td>
<td>37</td>
</tr>
<tr>
<td>7.1</td>
<td>Matrix Greedy Algorithm</td>
<td>37</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Designing the LMO</td>
<td>39</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Atom Correction Variants</td>
<td>40</td>
</tr>
<tr>
<td>7.1.3</td>
<td>Implementing the Linear Minimization Oracle for Matrix Factorizations</td>
<td>41</td>
</tr>
<tr>
<td>7.2</td>
<td>Tensor Factorization</td>
<td>41</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Generalised Tensor Greedy Algorithm</td>
<td>41</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Implementing the Linear Minimization Oracle for Tensor Factorizations</td>
<td>43</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Optional Correction of Selected Atoms</td>
<td>44</td>
</tr>
<tr>
<td>8</td>
<td>Experiments: Proof of Concept</td>
<td>47</td>
</tr>
<tr>
<td>8.1</td>
<td>Tensor Factorization</td>
<td>47</td>
</tr>
<tr>
<td>8.1.1</td>
<td>Tensor Completion</td>
<td>51</td>
</tr>
<tr>
<td>8.2</td>
<td>Additional Experiments on Matrices</td>
<td>51</td>
</tr>
<tr>
<td>9</td>
<td>Conclusion</td>
<td>53</td>
</tr>
<tr>
<td>A</td>
<td>Appendix</td>
<td>55</td>
</tr>
<tr>
<td>A.1</td>
<td>Proofs of the Main Results</td>
<td>55</td>
</tr>
<tr>
<td>A.1.1</td>
<td>Proof of Sublinear Convergence Rates</td>
<td>55</td>
</tr>
<tr>
<td>A.1.2</td>
<td>Proof of Corollary 10</td>
<td>59</td>
</tr>
<tr>
<td>A.1.3</td>
<td>Proof of Linear Convergence Rates</td>
<td>59</td>
</tr>
<tr>
<td>A.1.4</td>
<td>Proof of Theorem 17</td>
<td>63</td>
</tr>
<tr>
<td>A.1.5</td>
<td>On the Relationship Between Matching Pursuit and Frank Wolfe</td>
<td>64</td>
</tr>
<tr>
<td>B</td>
<td>Bibliography</td>
<td>67</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Introduction

Optimization under a structured low-rank constraint is a cornerstone problem in a huge variety of data-driven applications. It subsumes many common problems, including structured tensor (and matrix) factorization and completion. For example, structured low-rank factorization of a given N-th order tensor can be seen as a projection onto a linear combination of few rank-1 tensors, each of which being an outer product of N vectors from a structured set. Examples of structure in the vectors can be sparsity, group sparsity, non-negativeness etc. The structure is generally encoded as a constraint on each of the N factors of the factorization problem. Even without imposing a structure, the rank-constrained problem is already NP-hard to solve in general even if N is small (E.g. N=2 \(^1\)). Suitable convex relaxations are therefore typically applied, for tensor completion see, e.g., [57, 41] and for matrix factorization [7, 56, 47]. Recovery and performance conditions are an active area of research. However, this involves giving up explicit control over the rank. Another approach to tensor factorization is to use the unfolding of the tensor as in [63, 22] to reduce the problem to the matrix case, where factorization algorithms are better understood.

In this work, we take a different approach. We keep explicit control over the rank (the number of rank-1 tensors used in the approximation), as well as the precise structure of the used factors (vectors), which we call atoms. Our new proposed algorithm is a greedy method that adds one rank-1 atom per outer iteration, generalizing matrix and tensor variants of matching pursuit [61, 64] (which were only available for least-squares objectives so far) as well as Frank-Wolfe algorithms on factorizations [27, 33, 18, 31, 3] (which consider the convex hull case). By keeping the explicit low-rank factorization into the vector atoms at all times, we can study general algorithms and correction variants applying directly to the original non-convex problems.

\(^1\) For example, least-squares matrix completion is NP-hard even for a rank-1 factorization, as shown by [23].
1. **Introduction**

Taking a matrix perspective, iteratively adding a rank-1 atom for such generalized structured matrix factorization falls into the purview of pursuit algorithms. Each update is obtained from a linear minimization oracle (LMO), which outputs the best rank-1 atom with respect to a linearized version of the objective. Each iteration hence increases the rank by 1, while improving the approximation quality. We will systematically study this tradeoff between rank and approximation quality, by providing convergence rates of the iterates to the optimum of the function being minimized.

We provide a $O(1/t)$ convergence rate of our algorithm for general smooth objective functions defined on a Hilbert space, under any structural constraint on the atoms. Explicit constants depend on the geometry of the constraint set. Our algorithm in the matrix case is a strict generalization of [65] because we allow for any structured constraint on the atoms, whereas [65] and [61] were restricted to the Euclidean sphere case.

For the special case in which the tensor-variate function in question is strongly convex in the respective Hilbert space, we prove linear (geometric) convergence of our algorithm giving also a lower bound on the error’s decay. These results have a much wider scope than just our application to tensor pursuits. For the distance function, for example, our result is a generalization of the convergence rate of pursuit algorithms studied in the compressed sensing literature [12, 44, 24, 6] as it does not make assumptions about the coherence of the atoms.

The general nature of our convergence result allows its application to any set of atoms which induce an inner product – which in turn induces a distance function, that can be minimized by greedy pursuit. For the specific case of atomic vector sets being unit 2-norm balls without any further structure, this setup was used by [61], who showed linear convergence for matrix pursuit on 2-norm balls as vector atomic sets. This is a special case of our framework because we show linear convergence with any compact vector atomic sets.

For structured matrix and tensor problems, the linear problem being solved by the LMO itself may be NP-hard, although efficient solutions are available for several cases [4, 49]. We generalize all our convergence results to also hold if only an approximate version of the LMO is available.

**Related work:** There exists an extensive literature on structured matrix factorizations, which are a special case of our work. In our cases, the most relevant are the lines of research with iterative rank-1 greedy approximations such as the Frank-Wolfe algorithm [27, 33, 18, 31, 1]. In the general tensor case, a very similar approach has recently been investigated by [64], while [61, 65] discuss linear convergence for matrices, but neither of them applies to arbitrarily structured factorizations like we do here nor extends to general order tensors. In [66], it is presented a Frank-Wolfe-based greedy algorithm that minimizes a smooth risk function regularized using the tensor spectral norm to obtain a low-rank solution. Instead, we keep explicit control over the rank and the structure of the atoms for a general convex and smooth function. A cornerstone in the tensor factorization literature is represented by [36]. This work
explores the landscape of greedy algorithms for orthogonal tensor factorization questioning whether they are a suitable approach for such a task. The orthogonal tensor factorization is a harder problem which we do not address in this work. The problem of selecting the best rank-1 approximation of a given tensor is explored in [13]. On the other hand, our general rank-1 approximation depends on the function that is to be minimized. We retain [13] as a particular case of our atomic power method. [53] gives a provable sparse tensor decomposition obtained by using a truncated power method similar to the one we use here that was first introduced in [11]. We strictly generalize their approach, allowing for general convex functions. Rather than clustering a multitude of rank-1 tensors obtained with the power method we compute the weights at each iteration and we allow for a correction on the selected atoms. [50] tackles the sparse tensor decomposition problem using the unfolded matrices. The approaches of [50] and [53] are limited to sparseness while we generalize to any structural constraint. To the best of our knowledge, our algorithm is the first one allowing general structured tensor decomposition.

Considering the matrix factorization problem for specific atomic sets, there is a large body of literature, e.g., see [67] for Sparse PCA, [51, 2] for sparse non-negative PCA, and references therein.

There is a significant amount of research on greedy and pursuit algorithms, even more so on most commonly used flavors such as matching pursuit variants. [12] proves geometric convergence of matching pursuit and its orthogonal counterpart for finite dictionaries, while [44] and [24] give convergence results for (quasi-)incoherent dictionaries in Hilbert spaces of finite or infinite dimension. However, all of these are restricted to the least-squares optimization objective and assume the observed vector to lie in the dictionary span, which is typically not the case in the context of matrix and tensor factorizations. For infinite-dimensional pursuit, [34] showed convergence without providing rates. In functional analysis, the line of research of [55, 16, 54] is the closest to this approach. Our new algorithm is considerably more efficient per iteration, replacing the costly re-optimization of the original objective in every iteration as in [65, 55] by a simple least-squares solution instead.

The matrix completion problem has gained significant interest recently, motivated by powerful applications in recommender systems, signal processing, and most recently word-embeddings in NLP. A body of work cast the matrix completion as minimization of the rank of the matrix (or a convex surrogate) under the constraint that the observed entries are reproduced exactly or approximately [7, 48, 25]. A matrix pursuit view of the problem was taken by [61] by adding rank-1 updates iteratively to decrease the reproduction error on the observed entries.
Chapter 2

Notation

2.1 Notation

We represent vectors as small bold letters, e.g., \( \mathbf{u} \). Matrices are represented by capital bolds, e.g., \( \mathbf{X} \). Tensors are represented by cursive, e.g., \( \mathcal{X} \). Sets and spaces use the same notation but we will always make clear what we refer to. For a matrix \( \mathbf{A} \in \mathbb{R}^{m \times n} \) and a set of index pairs \( \Omega \), \( \mathbf{A}_\Omega \) stands for the matrix that is equal to \( \mathbf{A} \) at the entries indexed by \( \Omega \), and 0 elsewhere. Let \( [d] \) be the set \( \{1, 2, \ldots, d\} \). Given a subset \( \mathcal{A} \) of some vector space, let \( \text{conv}(\mathcal{A}) \) be the convex hull of the set \( \mathcal{A} \), and let \( \text{lin}(\mathcal{A}) \) denote the linear span of the elements in \( \mathcal{A} \). \( \mathbf{u} \otimes \mathbf{v} \) represents the rank-1 matrix given by the outer product \( \mathbf{uv}^\top \) of two vectors \( \mathbf{u}, \mathbf{v} \). Analogously we write \( \mathcal{A}_1 \otimes \mathcal{A}_2 = \{ \mathbf{u} \otimes \mathbf{v} \mid \mathbf{u} \in \mathcal{A}_1, \mathbf{v} \in \mathcal{A}_2 \} \) for the outer products from two sets \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \). Given a set \( \mathcal{A} \) we call its diameter \( \text{diam}(\mathcal{A}) \). Note that for convex hulls we have \( \text{diam}(\text{conv}(\mathcal{A})) = \text{diam}(\mathcal{A}) \), i.e., the diameter is attained at two vertices \([70]\).
Chapter 3

Frank-Wolfe

In this chapter we review part of the literature on Frank-Wolfe methods, also known as conditional gradient. The Frank-Wolfe algorithm and the analysis in [32], [39] and [38] are the first building block of our work.

3.1 Frank-Wolfe Algorithm

The Frank-Wolfe algorithm as depicted in [21, 32] is applicable to optimization problems of the following form:

$$\min_{x \in \text{conv}(\mathcal{A})} f(x)$$  \hspace{1cm} (3.1)

where $f$ is a convex and continuously differentiable function and $\text{conv}(\mathcal{A})$ is the convex hull of a bounded atoms’ set $\mathcal{A} \subset \mathcal{H}$ in the Hilbert space $\mathcal{H}$. The Frank-Wolfe method is a simple iterative and projection-free algorithm to solve such a problem and is given in Algorithm 1. In each iteration, it relies on a linear minimization oracle which we denote as LMO which solves the problem:

$$\text{LMO}(y) := \arg \min_{z \in \mathcal{A}} \langle y, z \rangle$$  \hspace{1cm} (3.2)

for given $y \in \mathcal{H}$.

**Algorithm 1** Frank-Wolfe in Hilbert Space

1: init $x_0 \in \text{conv}(\mathcal{A})$
2: for $t = 0 \ldots T$
3: \hspace{1cm} $\lambda = \frac{2}{t+2}$
4: \hspace{1cm} Find $z_t := (\text{Approx-})\text{LMO}_\mathcal{A}(\nabla f(x_t))$
5: \hspace{1cm} Optional: $\lambda = \arg \min_{0 \leq \lambda \leq 1} f(x_t + \lambda (z_t - x_t))$
6: \hspace{1cm} Update $x_{t+1} := x_t + \lambda (z_t - x_t)$
7: end for
At each iteration, the linearized objective is minimized over the convex feasible domain and the new iterate is computed as a convex combination of the new atom $z_t$ and the old iterate $x_t$. As discussed in [32], the convex update can be performed also by using a line search or as a convex combination of the previous atoms $z_{t'}$ for $t' \leq t$ that have been selected until iteration $t$. Due to its simplicity and broad applicability several variants are known, more specifically, the most relevant ones for our work are presented in [38].

3.1.1 Inexact Oracle

For Frank-Wolfe, solving the linear subproblem exactly can be too expensive, therefore it is desirable to allow an error on the LMO result. Such an error can be measured either additively or multiplicatively. Here in particular, we are interested in multiplicative errors, relative to the true solution. To formally define the multiplicative error in the Frank Wolfe literature we need to introduce a certificate of the approximation quality which is called duality gap [32]:

$$g(x) := \max_{z \in A} \langle \nabla f(x), x - z \rangle$$

(3.3)

Due to convexity it holds that $g(x) \geq f(x) - f(x^*)$ where $x^*$ is the minimizer of (3.1). Therefore, we assume that the inexact LMO introduces a multiplicative error on the duality gap. Formally, assuming a given quality parameter $\delta_{FW} \in (0, 1]$, the inexact LMO for any $d \in H$ returns a vector $\tilde{z} \in A$ such that:

$$\langle d, \tilde{z} - x \rangle \leq \delta_{FW} \langle d, z - x \rangle$$

(3.4)

3.1.2 Convergence

As shown in [39], Algorithm 1 exhibits the following (sublinear) convergence rate:

**Theorem 1** ([39]). *For each $t \geq 0$ and $\delta_{FW} \in (0, 1]$ the iterate $x_t$ of the Frank-Wolfe Algorithm (Algorithm 1) with a $\delta_{FW}$ inexact oracle (Equation (3.4)) satisfies:

$$f(x_t) - f(x^*) = \epsilon_t \leq \frac{2}{\delta_{FW} t} \left( \frac{L \text{diam}(A)^2}{\delta_{FW}^2} + \epsilon_0 \right)$$

There exists several corrective variants of Frank-Wolfe such as [30], [60], [28], [38] formally showed a linear convergence rate for corrective Frank-Wolfe algorithms for general strongly convex functions. We now briefly discuss their main results. The simplest corrective Frank-Wolfe variant presented in [38] is called pairwise Frank-Wolfe and is depicted in Algorithm 2.
Algorithm 2 Pairwise Frank-Wolfe in Hilbert Space

1: \text{init } x_0 \in \text{conv}(A), S_0 = \{x_0\}, \alpha^0_v = 1 \text{ for } v = x_0 \text{ and zero otherwise}
2: \text{for } t = 0 \ldots T
3: \text{Find } z_t := (\text{Approx-})\text{LMO}_A(\nabla f(x_t))
4: \text{v}_t \in \text{arg max}_{v \in S_t} \langle \nabla f(x_t), v \rangle
5: \text{if } \langle -\nabla f(x_t), z_t - x_t \rangle < c \text{ return } x_t
6: d_t := z_t - v_t, \quad \gamma_{\text{max}} = \alpha^t_v
7: \gamma_t \in \text{arg min}_{\gamma \in [0, \gamma_{\text{max}}]} f(x_t + \gamma d_t)
8: \text{Update } x_{t+1} := x_t + \gamma_t d_t \text{ and } \alpha \text{ consistently (see text)}
9: \text{Update } S_{t+1} := \{v \in A | \alpha^t_v > 0\}
10: \text{end for}

At each iteration we compute the away direction \(d_t\) which is defined in line 6 of the Algorithm. The away direction is used to move weight mass between two atoms at each step. The weights of the atoms at iteration \(t\) are called \(\alpha_t\) and are computed such that \(x_t = \sum_{v \in S_t} \alpha^t_v v\) with \(\alpha^t_v > 0\). To move the mass from \(v_t\) to \(z_t\) we use: \(\alpha^t_v = \alpha^t_v - \gamma\) and \(\alpha^{t+1}_z = \alpha^t_z + \gamma\) for some step size \(\gamma \leq \alpha^t_v\). The classical Frank-Wolfe shrinks all active weights at every iteration instead.

We now present the linear convergence rate (that is a multiplicative decrease in error in every iteration) for Algorithm 2.

**Directional Width.** \cite{38} The directional width of a set \(A\) with respect to a direction \(r\) is defined as \(\text{dirW}(A, r) := \max_{z_t \in A} \langle r, z_t - v_t \rangle\). The width of \(A\) is the minimum directional width over all possible directions in its affine hull.

**Pyramidal Directional Width.** \cite{38} We define the pyramidal directional width of a set \(A\) with respect to a direction \(r\) and a base point \(x_t \in \text{conv}(A)\) to be

\[
P_{\text{dirW}}(A, r, x_t) := \min_{S \in S_{x_t}} \text{dirW}(S \cup \{z(A, r)\}, r) = \min_{S \in S_{x_t}} \max_{s \in A, v_t \in S} \langle \frac{r}{\|r\|}, z_t - v_t \rangle,
\]

where \(S_{x_t} := \{S | S \subseteq A \text{ such that } x_t \text{ is a proper convex combination of all the elements in } S\}\), and \(z_t(A, r) := \text{arg max}_{v_t \in A} \langle r, v \rangle\) is the FW atom used as a summit.

**Pyramidal Width.** \cite{38} To define the pyramidal width of a set, we take the minimum over the cone of possible feasible directions \(r\) (in order to avoid the problem of zero width).

A direction \(r\) is feasible for \(A\) from \(x_t\) if it points inwards \(\text{conv}(A)\), (i.e. \(r \in \text{pconv}(A)\)).
We define the *pyramidal width* of a set $\mathcal{A}$ to be the smallest pyramidal width of all its faces, i.e.

$$PWidth(\mathcal{A}) := \min_{K \in \text{faces}(\text{conv}(\mathcal{A})), x_t \in K} \min_{r \in \text{cone}(K - x_t) \backslash \{0\}} PdirW(K \cap \mathcal{A}, r, x_t).$$ (3.6)

We are now ready to present the exponential decay of the suboptimality of the iterates.

**Theorem 2 ([38])**. Suppose that $f$ has $L$-Lipschitz gradient and is $\mu$-strongly convex over $\text{conv}(\mathcal{A})$. Let $M = \text{diam}(\mathcal{A})$ and $PWidth(\mathcal{A})$ as defined by (3.6). Then the suboptimality $\epsilon$ of the iterates of the Pairwise Frank-Wolfe Algorithm (Algorithm 2) decreases geometrically at each step in which $\gamma_t < \gamma_{\text{max}}$, that is

$$\epsilon_{t+1} \leq (1 - \rho) \epsilon_t,$$

where $\rho := \frac{\mu}{4L} \left( \frac{PWidth(\mathcal{A})}{M} \right)^2$.

The pyramidal width was introduced in [38] to address the so-called *zig-zagging problem*. Indeed, the Frank-Wolfe is known to have sublinear convergence if the optimum lies on the boundary of $\text{conv}(\mathcal{A})$ [21, 8, 19, 32]. Indeed, the iterates start to zig-zag between the atoms defining the face in which the optimum lies. The variants of Frank-Wolfe presented in [38] such as Algorithm 2 attenuates the problem and exhibits linear convergence for all strongly convex problems, independently of the position of the optimum inside the convex hull of the atoms.
Chapter 4

Matching Pursuit

In this chapter we review the part of the traditional literature on Matching Pursuit which is more relevant to discuss our results, focusing in particular on [44].

4.1 Matching Pursuit Algorithm

The Matching-Pursuit is a well known Algorithm from the Signal Processing and compressed sensing community [44, 24, 46, 12, 6] aimed at representing a signal \( y \in \mathcal{H} \) as a potentially sparse linear combination of vectors (which we call atoms) taken from an atom set \( \mathcal{A} \subset \mathcal{H} \). The optimization problem can be depicted as follows:

\[
\min_{x \in \text{lin}(\mathcal{A})} \| y - x \|_2^2
\]  

(4.1)

The algorithm is depicted in its corrective variant (Orthogonal Matching Pursuit) in Algorithm 3.

**Algorithm 3** Orthogonal Matching Pursuit

1. **init** \( x_0 \in \text{lin}(\mathcal{A}) \) \( \mathcal{S} = \{ x_0 \} \)
2. **for** \( t = 0 \ldots T \)
3. \( \text{Find } z_t := \text{(Approx-)LMO}_{\mathcal{A}}(y - x_t) \)
4. \( \mathcal{S} = \mathcal{S} \cup z_t \)
5. \( x_{t+1} := \arg \min_{x \in \text{lin}(\mathcal{S})} \| y - x \|_2^2 \)
6. **end for**

At each iteration, an atom is added to the active set \( \mathcal{S} \). Therefore, at each iteration, the solution becomes less sparse. Note that the gradient (i.e. \( y - x_t \)) in Algorithm 3 is always orthogonal to the elements in the set \( \mathcal{A} \setminus \mathcal{S} \). The classical version of matching
pursuit computes the new update purely as a linear combination between the previous iterate $x_t$ and the new atom $z_t$ as $x_{t+1} = x_t + \lambda z_t$ where $\lambda \in \mathbb{R}$ is chosen to minimize $\|y - x_{t+1}\|^2$. Again, as for Frank-Wolfe (Chapter 3), the greedy optimization relies on the same linear minimization oracle which we denote as LMO. Such an oracle solves the problem:

$$\text{LMO}(y) := \arg \min_{z \in A} -|\langle y, z \rangle|$$

for given $y \in \mathcal{H}$. This definition of LMO differs from the Frank-Wolfe one (Equation (3.2)) because here we maximize the absolute value (or equivalently minimize the negative absolute value). Note that the LMO of Equation (4.2) can be obtained with the LMO of Equation (3.2) using a symmetric set.

At each iteration, the linearized objective is minimized over the feasible domain and the new iterate is computed as a linear combination of the few selected atom in $S$.

### 4.1.1 Inexact Oracle

A weak selection of the new atoms (inexact LMO) is allowed, assuming a multiplicative error on the LMO value. Let $\tilde{z}$ be the result of the inexact LMO and $z$ the result of $\arg \min_{z \in A} -|\langle y, z \rangle|$, then Algorithm 3 allows for inexact oracles which satisfies:

$$|\langle y, \tilde{z} \rangle| \geq \delta_{MP} |\langle y, z \rangle|$$

for some $\delta_{MP} \in (0, 1]$. Note that the notion of inexact oracle for Matching Pursuit is different from the Frank-Wolfe one (Equation (3.4)).

### 4.1.2 Convergence

The goal of Matching Pursuit Algorithms is to find the best orthogonal projection of a vector onto the linear span of the atom set. For such a reason, in [24] the authors are interested in giving exact recovery conditions along with the convergence rates.

**Theorem 3 ([24]).** Let $A$ be a dictionary in a finite or infinite dimensional Hilbert space and let $\mu := \max_{k \neq l} |\langle z_k, z_l \rangle|$ be its coherence. For any finite index set $I$ of size $\text{card}(I) = m < \frac{1}{2} \left(1 + \frac{1}{\mu}\right)$ and any $y \in \text{lin}(A)$, Matching Pursuit:

1. picks up only correct atoms at each step: $(\forall n, k_n \in I)$;
2. converges exponentially:

$$\|x_n - y\|^2 \leq \left(\left(1 - \frac{1}{m}\right)(1 + \mu)\right)^n \|y\|^2$$

Since all the residuals remain in the subspace span $(z_k, k \in I) \subset \mathcal{H}$ this result is a stability condition.
4.1. Matching Pursuit Algorithm

For the analysis of the Orthogonal Matching Pursuit depicted in Algorithm 3 we need to introduce the so-called cumulative coherence \[46\]. Both the coherence and the cumulative coherence serves as an approximation of the exact recovery condition given in \[24\]. On the other hand, while the coherence is a property of the whole dictionary, the cumulative coherence depends on the number of atoms which are used to solve the optimization problem of Equation (4.1).

**Definition 4** (\[58\]). Let \( \mathcal{I} \subset [n] \) be an index set. For an integer \( m \), cumulative coherence function is defined as \( \mu(m) := \max_{|\mathcal{I}|=m} \max_{k \in [n] \setminus \mathcal{I}} \sum_{i \in \mathcal{I}} |\langle s_k, s_i \rangle| \).

As a special case, for \( m = 1 \) the cumulative coherence becomes the coherence of the dictionary. We are now ready to present the convergence rate.

**Theorem 5** (\[24\]). Let \( m \) be an integer such that:

\[ \mu(m) + \mu(m-1) < 1. \]

Then for any index set \( \mathcal{I} \) of size at most \( m \), any \( y \in \text{span}(z_k, k \in \mathcal{I}) \) and \( \delta_{\text{MP}} > \mu(m)/(1 - \mu(m-1)) \):

1. **Matching Pursuit with an inexact LMO picks up a correct atom at each step, i.e., for all \( n \geq 1, k_n \in \mathcal{I} \).**

2. **(Orthogonal) Matching Pursuit converges exponentially to \( y \).** More precisely we have \( \|y - x_n\|^2 \leq (\beta_m(\delta_{\text{MP}}))^{n} \|y\|^2 \) with

\[ \beta_m(\delta_{\text{MP}}) := 1 - \delta_{\text{MP}}^2 (1 - \mu(m-1))/m \]

We will generalize this result to arbitrary strongly convex functions and structured atom set in Chapter \[5\]. Furthermore, we will introduce a new geometric complexity measure of the set \( \mathcal{A} \) which better captures the performances of the algorithm, in particular in the coherent setting (i.e. when the coherence is large).
5.1 Greedy Algorithms in Hilbert Spaces

In this section, we present an overview of greedy algorithms for convex objectives defined over a Hilbert space. Let $\mathcal{H}$ be a Hilbert space with associated inner product $\langle x, y \rangle$, $\forall \ x, y \in \mathcal{H}$. The inner product induces the norm $\| x \|_2^2 := \langle x, x \rangle$, $\forall \ x \in \mathcal{H}$. Let $A \subset \mathcal{H}$ be a bounded set (the set of ‘atoms’) and let $f : \mathcal{H} \to \mathbb{R}$ be convex and $L$-smooth ($L$-Lipschitz gradient in the finite dimensional case). If $\mathcal{H}$ is an infinite dimensional Hilbert space, then $f$ is assumed to be Fréchet differentiable. We address optimization problems of the form

$$\min_{x \in D} f(x). \quad (5.1)$$

where $D$ is the constraints set which in the following will typically be the convex hull of the atomic set (conv($A$)) or its linear span (lin($A$)).

We write $x^*$ for a minimizer of (5.1). For any $y \in \mathcal{H}$, and a bounded set $A \subset \mathcal{H}$, the linear minimization oracle (LMO) is defined as

$$\text{LMO}_A(y) := \arg \min_{z \in A} \langle z, y \rangle. \quad (5.2)$$

Note that it is convenient to assume a symmetric atom set $A$. The reason is the particular choice we make in the definition of the oracle. Indeed the definition of LMO in (5.2) is equivalent to the one used for Matching Pursuit (see Equation (4.2)) if the set $A$ is symmetric. On the other hand, we do not require symmetry here.

**Constrained Optimization.** We introduce a weaker variant which, instead of optimizing directly $f$, minimizes its quadratic upper bound:

$$g_{x_i}(x) = f(x_i) + \langle \nabla f(x_i), x - x_i \rangle + \frac{L}{2} \| x - x_i \|^2. \quad (5.3)$$
where $L$ is the smoothness constant. A function $f : \mathcal{H} \to \mathbb{R}$ is called $L$-smooth if its gradient is Lipschitz continuous with Lipschitz constant $L$.

Inspired by the Fully-Corrective Frank Wolfe variant which in each step re-optimizes the original objective function over the set of all previous atoms (see e.g. [30, 32]), here we propose a simpler variant which instead re-optimizes over a quadratic over-approximation of the objective. In other words, it relies on a quadratic surrogate function of the objective, defined at each iterate. We call this algorithm variant uniformly corrective Frank-Wolfe, as depicted in Algorithm 4.

Algorithm 4 Uniformly Corrective Frank-Wolfe in Hilbert Space

1: init $x_0 \in \text{conv}(A)$, and $S := \{x_0\}$
2: for $t = 0 \ldots T$
3: Find $z_t := (\text{Approx-})\text{LMO}_A(\nabla f(x_t))$
4: $S := S \cup \{z_t\}$
5: Let $b := x_t - \frac{1}{L} \nabla f(x_t)$
6: Update $x_{t+1} := \arg \min_{z \in \text{conv}(S)} \|z - b\|^2$
7: Optional: Correction of some/all $z_0 \ldots t$
8: end for

Algorithm 4 differs from Algorithm 1 also because at each iteration we recompute the weights of all atoms in $S$ and not just the one added at the current iteration (as in Algorithm 4 line 6), i.e., we correct the weights of atoms selected in earlier iterations.

The name ‘uniformly’ is used to illustrate that the algorithm employs a simple uniform quadratic as the surrogate function (or upper bound on $f$), as given simply by the smoothness constant $L$. This is in contrast to second-order optimization methods such as Newton’s method, which relies on a non-uniform quadratic surrogate at each step. Note that this simple quadratic surrogate also has the side-effect that the iterations become related to projected gradient descent steps, with a step-size of $1/L$, which is exactly how the vector $b$ is defined in the algorithms. However, the crucial difference is that the projection step is only partial (defined by the current small set $S$) and not onto the entire constraint set of the original problem (defined by $A$).

Optimization over the Linear Span. We now move to optimization problems over linear spans $\text{lin}(A)$, and present an analogous generalized variant of Matching Pursuit [10] in Algorithm 5 building upon the Frank-Wolfe variant presented in Algorithm 4.
5.1. Greedy Algorithms in Hilbert Spaces

Algorithm 5 Generalized Matching Pursuit in Hilbert Space

1: init $x_0 \in \text{lin}(A)$, and $S := \{x_0\}$
2: for $t = 0 \ldots T$
3: Find $z_t := \text{(Approx-)LMO}_A(\nabla f(x_t))$
4: $S := S \cup \{z_t\}$
5: Let $b := x_t - \frac{1}{T} \nabla f(x_t)$
6: Update $x_{t+1} := \arg\min_{z \in \text{lin}(S)} \|z - b\|^2$
7: Optional: Correction of some/all $z_{0 \ldots t}$
8: end for

In contrast to the corrective update in Algorithm 4 (line 6), the update here in Algorithm 5 (line 6) only optimizes over the single new atom selected in the current iteration, performing a line-search. Also, the optimization is unconstrained as opposed to FW. Note that the optimal choice of the new weight (line-search) is computed in closed form as $\lambda = -\frac{(z_t - b, z_t)}{\|z_t\|^2}$.

We now present the uniformly corrective variant of Matching Pursuit in Algorithm 6.

Algorithm 6 Uniformly Corrective Generalized Matching Pursuit

1: init $x_0 \in \text{lin}(A)$, and $S := \{x_0\}$
2: for $t = 0 \ldots T$
3: Find $z_t := \text{(Approx-)LMO}_A(\nabla f(x_t))$
4: $S := S \cup \{z_t\}$
5: Let $b := x_t - \frac{1}{T} \nabla f(x_t)$
6: Update $x_{t+1} := \arg\min_{z \in \text{lin}(S)} \|z - b\|^2$
7: Optional: Correction of some/all $z_{0 \ldots t}$
8: end for

Once again, the computation of the re-optimized weights (line 6) is very efficient in practice and amounts to solving a linear system.

Algorithm 6 is a generalization of Orthogonal Matching Pursuit (OMP) [10]. Indeed, we allow for general convex functions without being restricted to the least squares objective. It is important to stress the fact that the gradient of $f$ computed at the current iterate is not always orthogonal to the previous atoms as it is the case for OMP. This would happen naturally when minimizing the least squares objective. In such a case, the gradient coincides with what is known as the residual in the context of OMP. Therefore, the algorithm iteratively adds an atom and finds the best approximation of the residual in terms of $S$ (the already selected atoms) in the least squares sense and computes the new residual. Since we update the weight of every atom in $S$ at each iteration, the residual is always orthogonal to the set $S$ itself. While this is
true when minimizing the least squares objective, for general functions this does not hold in general. The reason is that it is not true that the gradient computed in $x_t$ depends exclusively on the gradient at the previous iterate and how much of it was explained computing $x_t$. Indeed, the curvature of the function plays an important role in the direction of the gradient. Therefore, we strictly generalize Orthogonal Matching Pursuit.

For all these algorithms, in each iteration $t$, the linearized objective at the previous iterate is minimized over the set $A$ (which is the purpose of the LMO). This procedure yields the next atom $z_t$ to be added to the set $S$.

In the Uniformly Corrective Frank-Wolfe, Matching Pursuit and Uniformly Corrective Matching Pursuit Algorithms (Algorithm 4, 5 and 6) we update the iterate, as shown in line 6, imitating a gradient descent step with projection onto a ‘varying’ target – i.e., the span or a convex combination of the currently active atoms $S$. Recall that this follows from the quadratic approximation we choose as a surrogate of the function $f$.

The total number of iterations $T$ of Algorithm 4, 5 and 6 controls the trade-off between approximation quality, i.e., how close $f(x_T)$ is to the optimum $f(x^*)$, and the “structuredness” of $x_T$. The structuredness is due to the fact that we only use $T$ atoms from $A$ and to the structure of the factors themselves (e.g., sparsity). The fixed step size can be relaxed resulting in faster convergence. On the other hand, assuming a fixed step size we give convergence rates that depend on such step size. An adaptive step size could free the dependence of the rate from the smoothness $L$.

5.1.1 Inexact Linear Oracles and Atom Corrections

Recall that the linear minimization oracle was defined in Equation (5.2). An exact oracle is solving the linear problem of finding the element in a structured set which has the smallest inner product with a target element. An exact LMO is often very costly, in particular when applied to matrix (or tensor) factorization problems, while approximate versions can be much more efficient. We now generalize all the presented Algorithms to allow for an inexact LMO. Such a LMO was already explored for the Frank-Wolfe framework in [39]. First of all, they considered an additive error to the minimum of the linear problem that is given by the true LMO. Then, they also considered a multiplicative error on the true duality gap. We only focus on multiplicative errors.

Formally, assuming a given quality parameter $\delta_{FW} \in (0, 1]$, the inexact LMO for Frank-Wolfe for any direction $d \in H$ returns a vector $\tilde{z} \in A$ such that:

$$\langle d, \tilde{z} - x_t \rangle \leq \delta_{FW} \langle d, z - x_t \rangle$$

(5.4)

For Matching Pursuit algorithms (i.e. Algorithm 5 and 6) we use a different definition of the inexact oracle. Assuming the quality parameter $\delta_{MP} \in (0, 1]$, the inexact
LMO for any direction $d \in \mathcal{H}$ returns a vector $\tilde{z} \in \mathcal{A}$ such that:

$$\langle d, \tilde{z} \rangle \leq \delta_{\text{MP}} \langle d, z \rangle$$  \hspace{1cm} (5.5)

where $z = \text{LMO}_A(d)$. In other words, we consider a multiplicative error on the LMO problem. We relate the two definitions in Section 5.1.4. Unless differently specified we will refer to the quality parameter simply as $\delta$. We will always clarify which of the two definitions is considered. Further, we also allow an (optional) additional correction as shown in line 7 to change some of the atoms of the expansion, see, e.g., [40], to obtain better objective cost while using the same (small) number of atoms.

### 5.1.2 Sublinear Convergence Rates

In this section we will prove that Uniformly Corrective Frank-Wolfe, Matching Pursuit and Uniformly Corrective Matching Pursuit (Algorithms 4, 5 and 6) all have a sublinear $O(1/t)$ convergence rate for any convex and $L$-smooth function.

Recall that $\tilde{z}_t := \text{(Approx-)LMO}_A(\nabla f(x_{t-1}))$ is the direction given by the LMO used in the algorithm, and $z_t := \arg\min_{z \in \mathcal{A}} \langle z, \nabla f(x_{t-1}) \rangle$ denotes an exact solution of the LMO. The accuracy of the approximate LMO solution $\tilde{z}_t$ compared to $z_t$ will be measured either by the multiplicative error on the current duality gap as given in Equation (5.4) (for the FW algorithms), or a direct multiplicative error on the linear problem solved by the LMO as in Equation (5.5) (for the MP algorithm variants). In the following, $\|x\|_A = \inf\{c > 0 : x \in c \cdot \text{conv}(\mathcal{A})\}$ denotes the atomic norm of $x$ over $\mathcal{A}$.

**Frank-Wolfe Algorithm Variants.** We first give the convergence theorem for the Frank-Wolfe algorithm variants for the task of solving a constrained optimization problem $\min_{x \in \text{conv}(\mathcal{A})} f(x)$. We write $x^* := \arg\min_{x \in \text{conv}(\mathcal{A})} f(x)$ for an optimal solution.

**Theorem 6.** Let $\mathcal{A} \subset \mathcal{H}$ be a bounded set and let $f : \mathcal{H} \to \mathbb{R}$ be a $L$-smooth and convex function. Then, the Uniformly Corrective Frank-Wolfe method (Algorithm 4) when using a LMO of duality-gap-accuracy parameter $\delta \in (0, 1]$ (Equation 5.4) converges for $t \geq 0$ as

$$f(x_t) - f(x^*) \leq \frac{2 \left( \frac{L \cdot \text{diam}(\mathcal{A})^2}{\delta} + \epsilon_0 \right)}{\delta t + 2}$$

where $\epsilon_0 := f(x_0) - f(x^*)$ is the initial error in objective.

The proof follows from Theorem C.1 [39] and is given in Appendix A.1.1.
Matching Pursuit Algorithm Variants. We now move on to Matching Pursuit (MP) algorithm variants for the task of solving an optimization problem \( \min_{x \in \text{lin}(A)} f(x) \) over the unbounded linear space \( \text{lin}(A) \).

Optimizing the function \( f \) in the affine space is a totally different problem than optimizing it over a convex space. It also yields totally different applications, such as matrix and tensor factorization. In the following, \( \|x\|_A := \inf\{ c > 0; \ x \in c \cdot \text{conv}(A) \} \) is the atomic norm of \( x \) over \( A \) (also known as the gauge function of \( \text{conv}(A) \)). We say that the atomic norm is well-defined for \( x \in \mathcal{H} \) if \( \|x\|_A < \infty \).

We will again write \( x^\star := \arg\min_{x \in \text{lin}(A)} f(x) \) for an optimal solution in this case. Note that \( x^\star \) is not unique in general since the factorization problem is non convex. We therefore consider the solution with largest atomic norm.

**Theorem 7.** Let \( A \subset \mathcal{H} \) be a bounded and symmetric set and let \( f : \mathcal{H} \to \mathbb{R} \) be a \( L \)-smooth convex function. Let \( \rho := \max \{ \|x^\star\|_A, \|x_0\|_A, \ldots, \|x_T\|_A \} < \infty \). If the optimum is not unique we consider \( x^\star \) to be the one with largest atomic norm. Then, Matching Pursuit and Uniformly Corrective Matching Pursuit (Algorithms 5 and 6) do converge for \( t \geq 1 \) as

\[
f(x_t) - f(x^\star) \leq \frac{2L\rho^2 \text{diam}(A)^2}{\delta^2(t + 2)}.
\]

The proof is given in Appendix A.1.1. Our proof generalizes the FW convergence rate in the affine space relying on a rescaled convex set that includes both \( x^\star \) and \( x_t \forall t \). Therefore, the rate depends on the atomic norm of the iterates and the optimum. A similar dependence on the atomic norm can be found in the convergence rate presented in [26]. Their result holds for norm regularized convex functions to be optimized over a closed convex cone. On the other hand, we solve a different problem allowing for general convex functions minimized in the affine space of a bounded and symmetric set.

The sublinear convergence rate of Matching Pursuit algorithms (Theorem 7) is related to the Frank-Wolfe one (Theorem 6). Indeed, as depicted in Theorem 7 the rate reflects a different definition of the inexact oracle (which is to say Equation (5.5) instead of (5.4)) and a different relationship with the geometry of the set expressed by the atomic norm of the optimum and of the iterates (\( \rho \)). Furthermore, we notice how Theorem 7 does not depend on the initial error \( \epsilon_0 \) but holds only for \( t \geq 1 \). We give a different rate that also depends on \( \epsilon_0 \) and also holds for \( t = 0 \).

**Theorem 8.** Let \( A \subset \mathcal{H} \) be a bounded and symmetric set and let \( f : \mathcal{H} \to \mathbb{R} \) be a \( L \)-smooth convex function. Let \( \rho := \max \{ \|x^\star\|_A, \|x_0\|_A, \ldots, \|x_T\|_A \} < \infty \). If the optimum is not unique we consider \( x^\star \) to be the one with largest atomic norm. Then, Matching Pursuit and Uniformly Corrective Matching Pursuit (Algorithms 5 and 6) with a relative accuracy \( \delta \in (0, 1] \) on the LMO optimization problem (Equa-
5.1. Greedy Algorithms in Hilbert Spaces

Theorem (5.5) do converge for $t \geq 0$ as:

$$f(x_t) - f(x^*) \leq \frac{2 \left( L\rho^2 \text{diam}(A)^2 \right)}{\delta t + 2} + \epsilon_0.$$ 

The proof is given in Appendix A.1.1 and combines the ideas used in [39] in order to obtain a valid rate also for $t = 0$ with the generalization to the affine space introduced in Theorem 7. The relationship between Frank-Wolfe and Matching Pursuit will be systematically studied in Section 5.1.4.

Using well-known results from convex optimization, we can particularize Theorem 7 for $f(x) = \frac{1}{2}d^2(x,y)$ and obtain explicit constants as follows.

**Definition 9.** The effective inradius of a convex set $A$, denoted by $\text{inr}(A)$, is the radius of the largest $d$-dimensional Euclidean ball which can be inscribed in $A$, where $d$ is the dimension of the subspace spanned by $\text{lin}(A)$.

We henceforth assume that inradius is computed on the subspace spanned by $A$ if the elements in $A$ do not span the ambient space.

**Corollary 10.** Let $A \in \mathbb{R}^n$ be a finite symmetric set of atoms, or the convex hull of a finite set of atoms, and let $\rho := \max \{ \|x^*\|, \|x_0\|, \ldots, \|x_T\| \} < \infty$. Then, under the conditions of Theorem 7, Matching Pursuit and Uniformly Corrective Matching Pursuit (Algorithm 5 and 6) converges both with $f(x_t) - f(x^*) \leq \frac{2 \rho^2 \text{diam}(A)^2}{\delta \text{inr}(\text{conv}(A))^2(t+2)}$. If further $f(x) = \frac{1}{2}d^2(x,y)$, then $2\rho^2$ can be replaced by $2\|y\|^2$.

The effective inradius $\text{inr}(\text{conv}(A))$ generally depends on the ambient space dimension $d$. For example, the effective inradius of the $\ell_1$-ball scales as $O(\sqrt{d})$. Hence, if $A$ is the $\ell_1$-ball, Corollary 10 tells us that we need to take $T$ at least on the order of $d$ to obtain an $O(1)$ error $f(x_T) - f(x^*)$.

### 5.1.3 Linear Convergence Rates

It is possible to obtain faster convergence rates for specific objective functions, still over arbitrary atom structures. In particular, Algorithm 4 is known to have linear convergence in the case of $\mu$-strongly convex function.

In this section, we prove the linear convergence rates of Matching Pursuit (Algorithm 5) and Uniformly Corrective Matching Pursuit (Algorithm 6) for the optimization problem $\min_{x \in \text{lin}(A)} f(x)$ over the unbounded linear space $\text{lin}(A)$ for strongly convex $f$. To reiterate, the key distinction from the Frank-Wolfe variants (Algorithms 1, 4) and their known convergence rates is that the latter operate on a bounded set $\text{conv}(A)$. The relationship is further formalized in Section 5.1.4.

We begin our analysis by exploring a geometric property of the atom set which we call directional width.
5. General Greedy

Definition 11. The width of the set $\mathcal{A}$ as a function of a given non-zero vector $d$ is defined as:

$$W_{\mathcal{A}}(d) := \max_{z \in \mathcal{A}} \langle \frac{d}{\|d\|}, z \rangle$$

In general, the directional width can be zero depending on the choice of $d$.

Building upon the concept of directional width of a set, we define our most important geometric complexity constant of a set $\mathcal{A}$. We call this quantity the minimal intrinsic directional width:

Definition 12. Given a bounded set $\mathcal{A}$, we define its minimal intrinsic directional width as:

$$mDW(\mathcal{A}) := \min_{d \in \text{lin} (\mathcal{A}) \setminus \{0\}} W_{\mathcal{A}}(d).$$

Note that if the minimum was over all directions, the directional width would often be zero (whenever $d$ is orthogonal to all atoms $\mathcal{A}$). However, the crucial aspect of our above definition of width is that only directions in the span $\text{lin} (\mathcal{A})$ are allowed, hence the name intrinsic. For reasons that will become apparent soon, we are interested in sets which exhibit $mDW(\mathcal{A}) > 0$.

Examples. Note that a symmetric set satisfies the property $mDW(\mathcal{A}) > 0$. If there was a direction $d \in \text{lin} (\mathcal{A})$ and an atom $\bar{z}$ for which it holds that $\bar{z} = \arg \max_{z \in \mathcal{A}} \langle \frac{d}{\|d\|}, z \rangle$ and $\langle \frac{d}{\|d\|}, \bar{z} \rangle < 0$ then by linearity of the inner product $\langle \frac{d}{\|d\|}, -\bar{z} \rangle > 0$. But since the set is symmetric $-\bar{z} \in \mathcal{A}$ therefore $\bar{z}$ does not have the maximum inner product with $d$. On the other hand, the assumption $mDW(\mathcal{A}) > 0$ is less restrictive than requiring symmetry. An example is the set

$$\mathcal{A} = \left\{ \begin{pmatrix} 1 \\ 1 \\ 0 \\ -1 \\ -1 \\ 0 \end{pmatrix} \right\} \subset \mathbb{R}^2$$

which spans $\mathbb{R}^2$ and exhibits $mDW(\mathcal{A}) > 0$ without being symmetric.

Some examples of $mDW(\mathcal{A})$ computation are the L1 ball in $d$ dimension which has $mDW(\mathcal{A}) = \frac{1}{\sqrt{d}}$ and the L2 ball which has $mDW(\mathcal{A}) = 1$. For matrices and tensors it is easy to see that by definition of inner product the $mDW(\mathcal{A})$ can be computed using the vectorized form of the atoms. Indeed, let $a, b \in \mathcal{H}$ then $\langle a, b \rangle = \langle \text{vec}(a), \text{vec}(b) \rangle$. Using this fact, the set of tensors with norm 1 has $mDW(\mathcal{A}) = 1$ regardless of the order of the tensor. For the formal definition of inner product and norm between tensors we refer to [37].

Note that the $mDW(\mathcal{A})$ is monotone under adding more atoms, i.e. the quantity will only increase (or stay equal) if an additional atom is added to $\mathcal{A}$. The increase of the $mDW(\mathcal{A})$ depends on the direction of the atoms which are added to $\mathcal{A}$. Recall that a larger ratio $\frac{mDW(\mathcal{A})}{\text{radius}(\mathcal{A})}$ yields faster convergence for Theorem 13.
We exploit this fact in the procedure which we call symmetrizing the set $\mathcal{A}$. Indeed, the weights are allowed to be negative, hence, symmetrizing the set is potentially improving the $mDW(\mathcal{A})$ yielding faster convergence while the optimization problem remains equivalent to the original one. On the other hand, if the set does not originally have $mDW(\mathcal{A}) > 0$ the symmetrized set does every time the origin is in $\text{conv}(\mathcal{A})$ but it is not an atom. By symmetrizing we mean adding a negative copy of each atom in the set, i.e. considering the set $\mathcal{A} \cup -\mathcal{A}$.

The $mDW(\mathcal{A})$ is designed aiming for an all sets, both in the overcomplete or undercomplete cases, possibly continuous, and plays a similar role to the coherence in more classical linear convergence rate known for MP. This discussion is deepened in Theorem 17 where we relate the $mDW(\mathcal{A})$ with the cumulative coherence [24].

We next present and discuss the linear convergence of Algorithm 5 and 6. We again write $x^\star := \arg \min_{x \in \text{lin}(\mathcal{A})} f(x)$ for an optimal solution in this case. Note that as in the following we will only consider strongly convex objective functions $f$, the optimum $x^\star$ is actually unique here, as opposed to the above sublinear rates.

**Theorem 13.** Let $\mathcal{A} \subset \mathcal{H}$ be a bounded set such that $mDW(\mathcal{A}) > 0$ and let the objective function $f : \mathcal{H} \to \mathbb{R}$ be both $L$-smooth and $\mu$-strongly convex. Then, for $t \geq 0$ the suboptimality of the iterates of Matching Pursuit and Uniformly Corrective Matching Pursuit (Algorithm 5 and 6) with a relative accuracy $\delta \in (0, 1]$ on the LMO optimization problem (Equation (5.5)) decays exponentially as:

$$
\epsilon_{t+1} \leq \left(1 - \frac{\mu \delta^2 mDW(\mathcal{A})^2}{L \text{radius}(\mathcal{A})^2}\right) \epsilon_t
$$

where $\epsilon_t := f(x_t) - f(x^\star)$ denotes the suboptimality at step $t$.

The proof is presented in Appendix A.1.3. We now require the function to be globally $L$-smooth and strongly convex. We leave as future work a more general formulation in which the two properties are required only on a certain region. While we introduced a global convergence rate which is iteration independent it is interesting to present the exponential decay of the suboptimality as a function of the current iterate.

**Corollary 14.** When the conditions of Theorem 13 hold, Matching Pursuit and Uniformly Corrective Matching Pursuit (Algorithm 5 and 6) with a relative accuracy $\delta \in (0, 1]$ on the LMO optimization problem (Equation (5.5)) exhibits also a more strict iteration dependent exponential decay of the suboptimality of the iterates:

$$
\epsilon_{t+1} \leq \left(1 - \frac{\mu \delta^2 \mathcal{W}_A(-\nabla f(x_t))}{L \|x_t\|^2}\right) \epsilon_t \leq \left(1 - \frac{\mu \delta^2 mDW(\mathcal{A})^2}{L \text{radius}(\mathcal{A})^2}\right) \epsilon_t
$$

where $\epsilon_t := f(x_t) - f(x^\star)$ denotes the suboptimality at step $t$, $\nabla f(x_t)$ is the orthogonal projection of $\nabla f(x_t)$ onto $\text{lin}(\mathcal{A})$ and $\tilde{z}_t$ is the atom selected by the LMO at iteration $t$. 

23
5. General Greedy

The proof is included in the one of Theorem 13 in Appendix A.1.3.

Since we introduced the normalized width and the minimal directional width as new concepts to measure the geometric properties of the set of atoms we now show why these are important measures. We only cover the case of Matching Pursuit for the distance function since the update can be easily computed in closed-form. Furthermore, we consider only the case of the exact oracle (\( \delta = 1 \) in Equation (5.5)). We now present a lower bound of the decay of the suboptimality of the iterates for Matching Pursuit.

**Theorem 15.** Let \( \mathcal{A} \subset \mathcal{H} \) be a bounded set and let the objective function \( f : \mathcal{H} \to \mathbb{R} \) be both \( L \)-smooth and \( \mu \)-strongly convex. Assume, \( x^* := \arg \min_{x \in \mathcal{H}} f(x) = \arg \min_{x \in \mathcal{H}} f(x) \). Let \( \epsilon_t := f(x_t) - f(x^*) \) be the suboptimality of the iterates and \( z_t \) the atom selected at iteration \( t \) by the LMO. Then, for \( t \geq 0 \) the suboptimality of the iterates of Matching Pursuit (Algorithm 5) with an exact LMO does not decay faster than:

\[
\epsilon_{t+1} \geq \left( 1 - \frac{W_{\mathcal{A}}(-\nabla f(x_t))^2 L}{\frac{L}{2L - \mu}} \right) \epsilon_t
\]

The proof is given in Appendix A.1.3.

**Discussion.** The lower bound on the exponential decay given in Theorem 15 is iteration dependent, since we assume to know \( x_t \) and that we can compute the gradient of \( f \) at \( x_t \). We now discuss an example.

Let \( \Delta = \{ e_1, \ldots, e_d \} \) be the natural basis of \( \mathbb{R}^d \) and assume we are minimizing \( f(x) = \|y - x\|^2 \) over the set \( \mathcal{A} = \text{lin}(\Delta) \). Assume that each component of the target vector \( y \) is equal to 1. This satisfies the assumptions of Theorem 15. To estimate the lower bound constant we note that if \( \mu = L \) then \( L \frac{2L - \mu}{\mu} = 1 \). We now have to bound \( \frac{W_{\mathcal{A}}(-\nabla f(x_t))^2}{\|z_t\|^2} \). At iteration \( t < d \) we know that \( \nabla f(x_t) = -(y - x_t) \).

By the specific assumptions we made on the set \( \mathcal{A} \) and since \( y \) has a 1 in each component \( x_t \) has exactly \( t \) values which are equal to 1 and \( d - t \) zeros. Note that the minimization over the span of the symmetrized natural basis ensures that Matching Pursuit and Uniformly Corrective Matching Pursuit coincide by Gram-Schmidt. Indeed, this case is equivalent to computing the representation of the vector \( y \) in the natural basis one component at the time. Each step affects only one of the components of the residual and keeps the rest untouched, which is to say that if they were zero at iteration \( t - 1 \) they are zero also at iteration \( t \). Therefore, the other \( d - t \) non zero components have the same value they had at the first iteration, which is to say they are all equal to \( -1 \). Then, we have

\[
\frac{W_{\mathcal{A}}(-\nabla f(x_t))^2}{\|z_t\|^2} = \frac{1}{d - t} \quad (5.6)
\]
5.1. Greedy Algorithms in Hilbert Spaces

The whole lower bound becomes:

$$
\epsilon_{t+1} \geq \left( 1 - \frac{1}{d-t} \right) \epsilon_t
$$

Let us now consider the upper bound. Using the same argument of Equation (5.6) we have that the exponential decay is:

$$
\epsilon_{t+1} \leq \left( 1 - \frac{1}{d-t} \right) \epsilon_t
$$

Therefore, the exponential decay is tight in this example. In Theorem 13 we further bound the decay so that it depends only on the geometry of the atoms set. In this example we have $m\text{DW}(\mathcal{A})^2 = \frac{1}{d}$. Hence, we have:

$$
\epsilon_{t+1} \leq \left( 1 - \frac{1}{d-t} \right) \epsilon_t \leq \left( 1 - \frac{1}{d} \right) \epsilon_t
$$

Therefore, the ratio $\frac{m\text{DW}(\mathcal{A})}{\text{radius}(\mathcal{A})}$ in the linear convergence rate (Theorem 13) makes it loose in this example. On the other hand, it does not depend on the particular iterate and yields the global linear convergence rate. Note that the $m\text{DW}(\mathcal{A})$ is a geometric quantity and does not depend on either $x_0$ or $y$. On the other hand in the case of the symmetrized natural basis we can give an explicit value for $W_A(-\nabla f(x_t))$ at every iteration.

Corollary 16. Let $\Delta$ be the natural basis $\{e_1, \ldots, e_d\}$ of $\mathbb{R}^d$. Let $\mathcal{A}$ be the set obtained symmetrizing $\Delta$. Suppose we are minimizing $f(x) = \|y - x\|^2$ over the linear span of $\mathcal{A}$ with $y \in \mathbb{R}^d$. Let $x_0$ be the starting point of the Matching Pursuit Algorithm and assume that $\forall i \in [d], x_{0i} \neq y_i$. Then the error decay at iteration $t < d$ of Matching Pursuit (Algorithm 5) presented in Corollary 14 is tight and:

$$
\epsilon_{t+1} = \left( 1 - \frac{1}{d} \right) \epsilon_t
$$

where $I$ is the set of indexes of the components spanned by the atoms in $\mathcal{S}$ at iteration $t$.

The proof is trivial considering that at iteration $t$ the gradient changed from the first iteration only in the indexes in $I$ (they became zero).

Relationship between minimal directional width and coherence. It is interesting to compare the rate of Theorem 7 with the coherence-based rates that are popular in literature, such as [24]. In order to relate the two notions of cumulative coherence and directional width, we need some additional assumptions. First of all,
5. General Greedy

Let us consider only the least squares function in a $d$ dimensional euclidean space and assume that its minimizer lays in the span of the atomic set. We also require symmetry so that the traditional definition of LMO given in Equation (4.2) is equivalent to the LMO we introduced in Equation (5.2).

**Theorem 17.** Let $A \in \mathbb{R}^d$ be a symmetric and bounded set. Let also $B$ be a set such that $A = B \cup -B$ with $B \cap -B = \emptyset$ and $|B| = n$. Then, the cumulative coherence of the set $B$ is bounded by: $\mu(A, n - 1) \geq 1 - n \cdot \text{mDW}(A)^2$.

The proof is depicted in appendix A.1.4. In other words, in Theorem 17 we formally showed that if the directional width is close to zero, the coherence is close to 1 with a factor that depends on $n$. Note that by increasing the number of atoms, both the cumulative coherence and the $\text{mDW}(A)$ grow. Recall that when the cumulative coherence is 1, according to the rate of [24] there is no linear convergence. Furthermore, our rate is more robust than the one presented in [24]. Indeed, an adversarial could add an atom to the dictionary making the coherence 1. This can not happen in our setting: adding an atom can not make the $\text{mDW}(A) = 0$ and if the atom is added so that the $\text{mDW}(A)$ is arbitrarily small, the cumulative coherence is arbitrarily close to 1 by Theorem 17.

5.1.4 On the Relationship Between Matching Pursuit and Frank Wolfe

The convergence rates of Matching Pursuit and Frank Wolfe are related by the constant $\rho$ that essentially simulates a “blown up” set in which the analysis of Frank Wolfe can be applied. In this section, we explore this relationship.

Let us introduce the set $\alpha A = \{x : \exists z \in A \text{ s.t. } x = \alpha z\}$. In the following we assume $\alpha \geq \frac{\rho}{\delta}$. We will consider Algorithm 4 on the set $\alpha A$ and show its behavior when $\alpha$ grows.

We can now relate the step of Algorithm 4 with the one of Algorithm 5.

**Theorem 18.** Let $A \subset \mathcal{H}$ be a bounded set and let $f : \mathcal{H} \rightarrow \mathbb{R}$ be a $L$-smooth convex function. Let $\alpha > 0$ and let us fix an iteration $t > 0$ and the iterate computed at the previous iteration $x_t$. If $-\frac{\langle \nabla f(x_t), az_t - x_t \rangle}{L||az_t - x_t||^2} \leq 1$ the new iterate $x_{t+1}^{FW} = x_t + \lambda(az_t - x_t) \big|_{\lambda \in [0,1]}$ of Frank-Wolfe (Algorithm 1) using the set $\alpha = \{x : \exists z \in A \text{ s.t. } x = \alpha z\}$ converges to the new iterate $x_{t+1}^{MP} = x_t + \lambda(az_t) \big|_{\lambda \in \mathbb{R}}$ of Matching Pursuit (Algorithm 5) applied on the linear span of the set $A$ with rate:

$$\|x_{t+1}^{FW} - x_{t+1}^{MP}\| \in O \left( \frac{1}{\alpha} \right)$$

In particular when $\alpha$ grows to infinity the condition $-\frac{\langle \nabla f(x_t), az_t - x_t \rangle}{L||az_t - x_t||^2} \leq 1$ always holds (for all steps $t$). If the condition $-\frac{\langle \nabla f(x_t), az_t - x_t \rangle}{L||az_t - x_t||^2} \leq 1$ is not satisfied at step $t$
5.1. Greedy Algorithms in Hilbert Spaces

then the difference of the iterates increases linearly:

$$\|x_{t+1}^{FW} - x_{t+1}^{MP}\| \in O(\alpha)$$

The proof is given in Appendix A.1.5.

We also relate the two different notion of approximation quality. Considering again the set $\alpha A$ we can write:

$$\delta^{FW} = \min_{t \geq 0} \frac{\langle \nabla f(x_t), \alpha \tilde{z}_t - x_t \rangle}{\langle \nabla f(x_t), \alpha z_t - x_t \rangle}$$

and

$$\delta^{MP} = \min_{t \geq 0} \frac{\langle \nabla f(x_t), \tilde{z}_t \rangle}{\langle \nabla f(x_t), z_t \rangle}$$

Note that these are the largest values for $\delta$ such that Equation (5.4) and (5.5) respectively hold. Therefore, it holds that:

$$\lim_{\alpha \rightarrow \infty} |\delta^{FW} - \delta^{MP}| = 0$$

Which is to say that the inexact oracle definition of Matching Pursuit algorithms (Equation (5.5)) [24] is the equivalent to the Frank Wolfe definition (Equation (5.4)) [32] in the limit of $\alpha$ growing to infinity.

Discussion In some sense, Frank Wolfe can be suitable to solve the optimization problem:

$$\min_{\text{lin}(A)} f(x) \quad (5.7)$$

Indeed, if we knew in advance the atomic norm of the optimum, then, we could just consider a large enough convex set and run Frank-Wolfe (Algorithm 1) on $\alpha A$ with $\alpha = \rho \geq \|x^*\|_A$ assuming for simplicity an exact oracle. Note that Matching Pursuit always assumes that the optimum is finite (the residual is always finite) and so are the iterates. Using Frank-Wolfe in the linear space assumes that $\rho$ is accessible, which of course is not the case in practice.

The convergence rate of Frank-Wolfe on the set $\rho A$ would be the same convergence rate of Matching Pursuit depicted in Theorem [7]. Even though the iterates would differ, we have shown in Theorem [18] that the larger is $\rho$ the smaller is the distance between the iterates of the two algorithms at each step. Therefore, Matching Pursuit can be interpreted as Frank Wolfe on a very large convex set (infinitely large). Matching Pursuit safely assumes an a priori potentially infinite $\alpha$, even if in such a case the Frank-Wolfe algorithm would be helpless. In practice $\alpha < \infty$, and that is why Matching Pursuit converges.
5. General Greedy

Similarly to what presented in [38] our linear rate of (Uniformly Corrective) Matching Pursuit depends on the directional width which is a version of their pyramidal width centered at zero. Indeed, whenever the atomic set has a small width the steps towards the optimum are smaller in that direction.

We want to conclude this section with a short discussion on \( \rho \), the constant introduced in the sublinear rate (6), and \( \text{mDW}(A) \). Both the constants capture some geometric properties of the set. While the \( \text{mDW}(A) \) is purely geometrical, it represents also a lousy bound. Indeed, the set might have a very small \( \text{mDW}(A) \) but the algorithm would still converge linearly. The reason is that this constant is used to bound the inner product of the atoms with the normalized gradient (Definition 11), therefore the actual convergence rate it is highly dependent on the shape of the function as well. For the sublinear rate, we take a different perspective and take into account the specificity of the problem to be solved. Indeed, the constant \( \rho \) does not only reflect a sort of width of the set but is a width in the directions that are relevant for the instance of the problem we want to solve. Therefore, while the \( \text{mDW}(A) \) is easy to compute but can result in a lousy bound, \( \rho \) can not be computed \textit{a priori}.

5.1.5 Boosting and Leveraging

In this section we want to connect Algorithm 6 with the well known Boosting and Leveraging framework [45]. Let \( \mathcal{D} \) be a dataset consisting of \( N \) labelled data points \( \{(d_n, y_n)\}_{n=1}^N \) sampled iid from a sample space according to some unknown distribution \( P(d_n) \). We aim to find a linear combination of \( T \) weak hypothesis \( h_i \) parametrized by a vector \( z_t \) such that \( y_n = \sum_{i=1}^T \alpha_i (d_n, z_i) \) with \( \alpha_i \in \mathbb{R} \forall i = 1 \ldots T \). The algorithm has access to an oracle that provides the correct label of each data point. Unfortunately, the target concept in the instance space used by the oracle is unknown, and the generalization precision of an algorithm in connection with a training dataset in not accessible. The standard approach is to minimize the empirical risk whose generalization error bounds are widely discussed in the Statistical Learning Theory literature (E.g. [45] and [59]). Taking the point of view of [45] and [62] we intend boosting as an optimization algorithm minimizing a particular loss function \( f(z, \mathcal{D}) \) on the training sample. We assume that \( f \) is additive on the training sample, convex and \( L \)-smooth. In particular, the general boosting algorithm is presented in Algorithm 7.

\begin{algorithm}
\caption{Boosting}
1: \textbf{init} \( \mathcal{D} = \{(x_n, y_n)\}_{n=1}^N \), \( x_0 := 0 \), and \( \mathcal{S} := \{x_0\} \)
2: \textbf{for} \( t = 1 \ldots T \)
3:  \quad \text{train} \( z_t \) on \( \{S, \nabla f(S, x_{t-1})\} \)
4:  \quad \mathcal{S} := \mathcal{S} \cup \{z_t\} \\
5:  \quad \text{Update} \ x_t := \arg \min_{z \in \text{lin}(\mathcal{S})} f(S, z) \\
6: \textbf{end for}
\end{algorithm}

28
5.1. Greedy Algorithms in Hilbert Spaces

Regularization is frequently used to prevent overfitting as in [45] and [69]. Our approach differs from boosting because we allow for structure on the factors, hence, we do not need to rely on a regularizer but we can directly constrain the optimization problem. Furthermore, boosting can be obtained as a particular case of Algorithm 6. Indeed, at each iteration, the boosting algorithm finds the hypothesis with the maximal edge. It was shown in [45] that the hypothesis $h_t$ obtained at iteration $t$ is the one that has the largest inner product with the negative gradient computed at the previous iterate, which is precisely the role of the LMO. The new classifier can be seen as a vector containing the result of $h_t(x_n)$ at the index $n$. Our approach is more general since we allow for optimization over a richer class of factors (E.g. rank-1 matrices and tensors). Also, the inexact LMO definition recovers the $\beta$-relaxed edge maximization as is presented in [45]. Indeed, we get the same asymptotic convergence rate presented in [45] in the cases of general convex function [68] and strongly convex [42] considering both the case of exact and inexact oracle.
Chapter 6

Background on Tensors Algebra

This survey on tensors Algebra is inspired by [37] and the lecture publicly available at www.youtube.com/watch?v=F-eORHovOe0.

6.1 Introduction and Notation

Definition 19. We define the tensor product $\otimes$ of vector spaces as the map $A \times B \to A \otimes B$, $\forall x \in A$, $y \in B$ $(x,y) \mapsto x \otimes y$. For two vectors the tensor product is their outer product, i.e. $(x \otimes y)_{(i,j)} = x_i y_j$.

Proposition 20. $\otimes$ is $F$-bilinear. $\forall x, x' \in A, y, y' \in B, \lambda \in F$ it holds that:

$$(\lambda x + x') \otimes y = \lambda (x \otimes y) + x' \otimes y$$

and

$$x \otimes (\lambda y + y') = \lambda (x \otimes y) + x \otimes y'$$

Proposition 21. $\otimes$ is associative: $(A \otimes B) \otimes C = A \otimes (B \otimes C)$

Definition 22. 

- Given a field $F^{m \times n \times p}$ a tensor is defined as a collection of numbers $a$ arranged in a multidimensional array: $(X_{ijk}) \in F^{m \times n \times p}$.
- Given three vector spaces $A, B$ and $C$ (an arbitrary number in general) over a field $F$, a tensor is defined as $X \in A \otimes B \otimes C$. In other words, a tensor is an element in a tensor product (Definition 19) of vector spaces (a tensor space).

Proposition 23. Let $a_1, \ldots, a_m$, $b_1, \ldots, b_n$ and $c_1, \ldots, c_k$ be the basis of $A$, $B$ and $C$ respectively. Then $\{a_i \otimes b_j \otimes c_k\}$ is a basis of the tensor space of $X$. Therefore $X$ can be written as:

$$X = \sum_{k=1}^{p} \sum_{j=1}^{n} \sum_{i=1}^{m} X_{ijk} a_i \otimes b_j \otimes c_k$$
Definition 24. The generalization to matrices of the outer product $\otimes$ of vectors is called Kronecker product and is denoted with $\otimes^{\text{Kr}}$. The Kronecker product of two matrices gives the matrix of the tensor product with respect to a standard choice of basis. In general, given two matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{K \times L}$, their Kronecker product is $A \otimes^{\text{Kr}} B \in \mathbb{R}^{(IK) \times (JL)}$: 

$$A \otimes^{\text{Kr}} B = \begin{pmatrix}
    a_{11}B & a_{12}B & \cdots & a_{1J}B \\
    a_{21}B & a_{22}B & \cdots & a_{2J}B \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{I1}B & a_{I2} & \cdots & a_{IJ}B
\end{pmatrix} = \begin{bmatrix}
    a_1 \otimes^{\text{Kr}} b_1, a_1 \otimes^{\text{Kr}} b_2, \ldots, a_J \otimes^{\text{Kr}} b_{L-1}, a_J \otimes^{\text{Kr}} b_L
\end{bmatrix}$$

Definition 25. The order of a tensor is the number of dimensions (also called ways or modes).

Definition 26. The fiber of a tensor is a vector obtained by fixing every index but one (E.g. $x_{ijk}$).

Definition 27. The slice of a tensor is a matrix defined by fixing every index but two (E.g. $X_{ij,\ast}$).

Definition 28. The norm of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is defined as:

$$\|\mathcal{X}\| = \sqrt{\sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1 \cdots i_N}^2}$$

Definition 29. The inner product of two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is defined as:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1 \cdots i_N} y_{i_1 \cdots i_N}$$

It follows that $\langle \mathcal{X}, \mathcal{X} \rangle = \|\mathcal{X}\|^2$

Definition 30. An $N$th-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is rank-1 if it can be written as the outer product of $N$ vectors, i.e.

$$\mathcal{X} = v^{(1)} \otimes \cdots \otimes v^{(N)}$$

In other words, $\forall 1 \leq i_n \leq I_n$:

$$x_{i_1 \cdots i_N} = v^{(1)}_{i_1} \cdots v^{(N)}_{i_N}$$

6.2 Dual Vector Space and Rank

Definition 31. Let $A$ be an $m$-th dimensional vector space. We call dual vector space of $A$ the set of linear maps $A^* = \{ \alpha : A \to \mathbb{F} \}$. The dual basis for $A$ is $\alpha_1, \ldots, \alpha_m$ where $\alpha_i(a_j) = \delta_{ij}$ and $\delta_{ij}$ is the Kronecker delta.
6.2.1 Matrices: Vectorization

Let \( a_i \in A \) and \( b_j \in B \) be the \( i \)-th and \( j \)-th vectors from a basis of \( A \) and \( B \) respectively. Be \( M \in A \otimes B \Rightarrow M = \sum_{i=1}^{n} \sum_{j=1}^{m} M_{ij} a_i \otimes b_j \) a linear map. We then have the linear extension of the map \( M \) as (with abuse of notation) \( M : A^* \rightarrow B, \alpha_s \mapsto \sum_{i=1}^{n} M_{ij} \alpha_s(a_i) b_j = \sum_{j=1}^{m} M_{si} b_j \) where \( s \leq m \). This means that in the \( s \)-th columns and row index \( j \) we have \( M_{sj} \) (therefore the map is parametrized by the transpose of \( M \)). One could do the same starting from the vector space \( B^* \) as \( M : B^* \rightarrow A, \beta_s \mapsto \sum_{i=1}^{n} M_{is} a_i \) where now \( s \leq n \) and \( \beta_1, \ldots, \beta_n \) is the dual basis of \( B \) (therefore the map is parametrized by \( M \)).

**Definition 32.** The vectorized version of a matrix \( M \) is a linear map \( M_F : (A \otimes B)^* \rightarrow \mathbb{F} \). Therefore \( M_F = (M_{11}, \ldots, M_{1n}|M_{21}, \ldots, M_{2n}| \ldots |M_{m1}, \ldots, M_{mn}) \).

This tells us how to take an abstract tensor in the tensor space and write it as a matrix as follows.

6.2.2 Tensors: Flattening

Recall that \( \mathcal{X} \in A \otimes B \otimes C = (A \otimes B) \otimes C \). Therefore, we define \( X_C \) as the linear map \( X_C : (A \otimes B)^* \rightarrow C \). Moreover, \( \mathcal{X} = \sum_{k} \left( \sum_{i} X_{ijk} a_i \otimes b_j \right) \otimes c_k \). The result is:

\[
X_F = \begin{pmatrix}
  x_{111} & \cdots & x_{1n1} & x_{211} & \cdots & x_{2n1} & \cdots & x_{m11} & \cdots & x_{mm1} \\
  x_{112} & \cdots & x_{1n2} & x_{212} & \cdots & x_{2n2} & \cdots & x_{m12} & \cdots & x_{mm2} \\
  \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
  x_{11p} & \cdots & x_{1n1} & x_{21p} & \cdots & x_{2n1} & \cdots & x_{m1p} & \cdots & x_{mm1}
\end{pmatrix}
\]

where \( X_{11\ldots1}, X_{21\ldots1}, \ldots, X_{m1\ldots1} \) are the slices according to Definition 27. The example in Equation (6.2.1) refers to the map \( X_A : (B \otimes C) \rightarrow A \). The same can be done for \( X_B : (A \otimes C) \rightarrow B \) and \( X_C : (A \otimes B) \rightarrow C \). In general \( \mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N} \) be a tensor of arbitrary dimension. The \( n \)-th unfolding \( X_{(n)} \) is computed placing the mode-\( n \) fibers (Definition 26) to be the columns of a matrix. The tensor element \( x_{1i_1\ldots i_N} \) maps to the matrix element \( (i_n, j) \) where:

\[
j = 1 + \sum_{k=1}^{N} (i_k - 1) J_k
\]

with:

\[
J_k = \prod_{m=1}^{k-1} I_m
\]

In this way the result is a linear map \( X_{(n)} : \mathbb{R}^{I_1 \times \cdots \times I_N} \rightarrow \mathbb{R}^{I_n} \). We will also need to find the vectorized representation of the outer product of the atoms in the subspace.
6. Background on Tensors Algebra

\[ \mathbb{R}^{I_k \times n}. \] This is again the Kronecker product of the atoms in the reversed order:

\[ \text{vec}(\otimes_{k \neq n} i_k) = \otimes_{k = 1}^{N} I_k \]

**Definition 33.** The collection of ranks of the different unfoldings defined above of a tensor \( X \) is called multilinear rank:

\[ \text{MR}(X) = (\text{rank}(X_A), \text{rank}(X_B), \text{rank}(X_C)) \]

Note that in general those ranks do not agree.

**Example 1.** Be \( X = a_1 \otimes b_1 \otimes c_1 + a_2 \otimes b_2 \otimes c_1 \)

\[
X_A = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}
\]

\[
X_B = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}
\]

\[
X_C = \begin{pmatrix}
1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]

So \( \text{MR}(X) = (2, 2, 1) \). Indeed \( \text{Img}(X_A) = \{a_1, a_2\} \), \( \text{Img}(X_B) = \{b_1, b_2\} \) and \( \text{Img}(X_C) = \{c_1\} \). In conclusion \( X \in \{a_1, a_2\} \otimes \{b_1, b_2\} \otimes \{c_1\} \)

### 6.3 More on Tensor Rank and CP decomposition

**Definition 34.** Given \( \lambda \in \mathbb{R}' \), \( A \in \mathbb{R}^{m \times r} \), \( B \in \mathbb{R}^{n \times r} \) and \( C \in \mathbb{R}^{k \times r} \) we define \( [[\lambda; A, B, C]] \in \mathbb{R}^{m \times n \times k} \) as:

\[ [[\lambda; A, B, C]] = \sum_{s=1}^{r} \lambda_s A_s \otimes B_s \otimes C_s \]

**Definition 35.** We say that a tensor \( X \in \mathbb{R}^{m \times n \times k} \) is in Kruskal form if:

\[ X = [[\lambda; A, B, C]] \]

**Definition 36.** We define the vectorization of a tensor \( X \) \( \text{vec}(X) \) as a vectorization of the associated mode-1 unfolded matrix \( X_A \)

**Proposition 37.** if \( X = [[\lambda; A, B, C]] \in \mathbb{R}^{m \times n \times k} \), then:

\[ X_{ijk} = \sum_{s=1}^{r} \lambda_s A_{is} B_{js} C_{ks} \]

\[ \text{vec}(X) = \sum_{s=1}^{r} \lambda_s C_{s} \otimes \text{Kr}_{B_{is}} \otimes \text{Kr}_{A_{is}} \]
6.3. More on Tensor Rank and CP decomposition

Let us define some other operators between tensors that will be used.

**Definition 38.** The Khatri-Rao Product \( \odot \) for \( A = [a_1, a_2, \ldots, a_J] \in \mathbb{R}^{I \times J} \) and \( B = [b_1, b_2, \ldots, b_J] \in \mathbb{R}^{T \times J} \) with the same number of columns \( J \) performs the following operation:

\[
A \odot B = \begin{bmatrix}
    a_1 \otimes b_1, a_2 \otimes b_2, \ldots, a_J \otimes b_J
\end{bmatrix}
= \begin{bmatrix}
    \text{vec}(b_1a_1^T), \ldots, \text{vec}(b_Ja_J^T)
\end{bmatrix}
\]

**Proposition 39.** The Khatri-Rao Product is:

- **associative:** \( A \odot (B \odot C) = (A \odot A) \odot C \)
- **distributive:** \( (A + B) \odot C = A \odot C + B \odot C \)
- **non-commutative:** \( A \odot B \neq B \odot A \)
- **simplified cross-product:** \( (A \odot B)^T (A \odot B) = A^T A \ast B^T B \)
- **if** \( x \in \mathbb{R}^{mn} \) **and** \( y = (B \odot C)^T x \) **then if** \( X \) **is the matricized version of** \( x \) **we have:**

\[
y = \begin{bmatrix}
    c_1^T X b_1 \\
    \vdots \\
    c_r^T X b_r
\end{bmatrix}
\]

**Definition 40 (CP-approximation problem).** Given \( A \in \mathbb{R}^{m \times n \times p} \) and \( r \in \mathbb{N} \), find \( \lambda \in \mathbb{R}^r, A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{n \times r} \) and \( C \in \mathbb{R}^{p \times r} \) so that:

\[
A \approx \left[ [\lambda; A, B, C] \right] = X
\]

Or in its least squares formulation find \( \lambda, A, B, C \) s.t.

\[
\| A - X \|^2_F = \| \text{vec}(A) - \sum_{s=1}^r \lambda_s C_s \otimes Kr B_s \otimes Kr A_s \|^2_F
\]

is minimized

The modal unfolding of \( X \) is:

\[
X_A = \sum_{s=1}^r \lambda_s A_s \otimes (C_s \otimes B_s)^T = \text{Adiag}(\lambda)(C \circ B)^T
\]

\[
X_B = \sum_{s=1}^r \lambda_s B_s \otimes (C_s \otimes A_s)^T = \text{Bdiag}(\lambda)(C \circ A)^T
\]

\[
X_C = \sum_{s=1}^r \lambda_s C_s \otimes (B_s \otimes A_s)^T = \text{Cdiag}(\lambda)(B \circ A)^T
\]
The alternating least squares solution framework then is:
\[
\|A - X\|_F = \|A - A diag(\lambda)(C \odot B)^T\|_F \Leftarrow \text{fix } B \text{ and } C \text{ and improve } \lambda \text{ and } A \\
= \|A - B diag(\lambda)(C \odot A)^T\|_F \Leftarrow \text{fix } A \text{ and } C \text{ and improve } \lambda \text{ and } B \\
= \|A - C diag(\lambda)(B \odot A)^T\|_F \Leftarrow \text{fix } A \text{ and } B \text{ and improve } \lambda \text{ and } C
\]

Let us consider the first step for example, the solution is found solving:
\[
(C \odot B)^T(C \odot B)\tilde{A} = (C \odot B)^T A^T_A
\]

where \(\tilde{A}\) is \(A\) normalized column-wise, therefore \(\lambda\) is the norm itself.

### 6.3.1 Comments on Tensor Problems

Simple matrix problems like solving a linear system or finding the top eigenvalues of a matrix are extremely common in many scientific and engineering applications. Recently, tensor methods became very popular for the very same reason. As presented in [29] the range of possible applications is wide: computational biology, data analysis, pattern recognition, computer vision, signal processing, wireless communication just to name a few. On the other hand, in [29] it is showed how tensor problems are “almost invariably computationally hard”. For example, the Eigenvalue decomposition, the rank over both \(\mathbb{R}\) and \(\mathbb{C}\) and the best rank 1 approximation are all NP-hard [29, 14]. Furthermore, greedy approaches to the rank problem where shown to be ineffective [36]. We do not solve the rank problem. Indeed, we only care about minimizing a convex cost function on a constrained set and we do not require orthogonality on the factors. An orthogonal tensor decomposition is a much harder problem. Finding an exact decomposition of a tensor is still NP-hard, but greedy approaches are available to compute good approximations.
Chapter 7

A Generalized Greedy Algorithm for Structured Non-Convex Matrix and Tensor Factorizations

In this chapter we reproduce part of our technical report [35]. We extend this work to tensor factorizations as well, and introduce several types of practical problems suitable for this setting, namely structured matrix and tensor factorizations.

7.1 Matrix Greedy Algorithm

We now leverage the general algorithm framework introduced in Chapter 5 for structured matrix factorizations. In order to encode interesting structure for matrix factorizations, we consider the set of rank-1 matrices \( \mathcal{A}_1 \otimes \mathcal{A}_2 \) as atom set, which is simply constructed by two arbitrary sets of vector atoms \( \mathcal{A}_1 \subseteq \mathbb{R}^n \) and \( \mathcal{A}_2 \subseteq \mathbb{R}^m \). Specializing the general optimization problem (5.1) to sets \( \text{lin}(\mathcal{A}_1 \otimes \mathcal{A}_2) \), we obtain the following structured matrix factorization notion: Given an objective function \( f : \mathbb{R}^{n \times m} \rightarrow \mathbb{R} \), we want to find a matrix \( X \) optimizing

\[
\min_{X \in \text{lin}(\mathcal{A}_1 \otimes \mathcal{A}_2)} f(X).
\]

When restricting (7.1) to candidate solutions of rank at most \( T \), we obtain the following equivalent and more interpretable factorized reformulation:

\[
\min_{u_i \in \mathcal{A}_1, \forall i \in [T], \quad v_i \in \mathcal{A}_2, \forall i \in [T], \quad \alpha \in \mathbb{R}^T} f\left( \sum_{i=1}^{T} \alpha_i u_i \otimes v_i \right).
\]

**Symmetric factorizations.** The above problem structure is also interesting in the special case of symmetric matrices, when restricting to just one set of vector atoms \( \mathcal{A}_1 \) (and \( u = v \)), which results in symmetric matrix factorizations build from atoms of the form \( uu^\top \).
7. **A Generalized Greedy Algorithm for Structured Non-Convex Matrix and Tensor Factorizations**

Table 7.1: Some examples of application of our matrix greedy framework (Algorithm [8]) taken from [35]. The table characterizes the applications by the set of atoms used as to enforce matrix structure (rows), and two prominent optimization objects (columns) for a given observed \( Y \) or \( Y_\Omega \), being low-rank matrix approximation and low-rank matrix completion (MC). All cases apply both for symmetric and for general rectangular matrices. \( B_F(\cdot, g(\cdot)) \) is a Bregman Divergence [25], uniquely identified by the function \( F \), while \( g(\cdot) \) is derived from \( F \). It is a generalization of the squared distance which is recovered with \( F(x) = x^2 \). We only assume \( B_F(\cdot) \) is smooth, or restricted smooth on the domain.

(a) **Symmetric Structured Matrix Factorizations**, \( u_t \in \mathcal{A}_1 \) \( \forall t, X = \sum_t \alpha_t u_t \otimes u_t \)

<table>
<thead>
<tr>
<th>Atoms ( \mathcal{A}_1 )</th>
<th>( \sum_{ij} B_F(Y_{ij}, g(X_{ij})) )</th>
<th>( \sum_{(i,j) \in \Omega} B_F(Y_{ij}, g(X_{ij})) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( { u : | u |_2 = 1 } )</td>
<td>Exp. family PCA</td>
<td>Exp. family MC</td>
</tr>
<tr>
<td>( { u : | u |_2 = 1, | u |_0 = k } )</td>
<td>Exp. family sparse PCA</td>
<td>Exp. family structured MC</td>
</tr>
</tbody>
</table>

(b) **Non-Symmetric Structured Matrix Factorizations**, \( u_t \in \mathcal{A}_1, v_t \in \mathcal{A}_2 \) \( \forall t, X = \sum_t \alpha_t u_t \otimes v_t \) and for any set \( S^+ := S \cap \{ u : u \geq 0 \} \)

<table>
<thead>
<tr>
<th>Atoms ( \mathcal{A}_1 )</th>
<th>Atoms ( \mathcal{A}_2 )</th>
<th>( | Y - X |_F^2 )</th>
<th>( \sum_{(i,j) \in \Omega} B_F(Y_{ij}, g(X_{ij})) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( { u : | u |_2 = 1 } )</td>
<td>( { v : | v |_2 = 1 } )</td>
<td>SVD</td>
<td>Exp. family MC</td>
</tr>
<tr>
<td>( { u : | u |_2 = 1, | u |_0 = k } )</td>
<td>( { v : | v |_2 = 1, | v |_0 = q } )</td>
<td>sparse SVD</td>
<td>Exp. family structured MC</td>
</tr>
<tr>
<td>( { u : | u |_2 = 1, }^+ )</td>
<td>( { v : | v |_2 = 1 }^+ )</td>
<td>NMF</td>
<td>Exp. family structured MC</td>
</tr>
<tr>
<td>( { u : | u |_2 = 1, | u |_0 = k }^+ )</td>
<td>( { v : | v |_2 = 1, | v |_0 = q }^+ )</td>
<td>sparse NMF</td>
<td>Exp. family structured MC</td>
</tr>
</tbody>
</table>

**Applications.** We present some prominent applications of structured matrix factorizations within our greedy framework in Table 7.1.

The vector atom sets \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) encode the desired matrix factorization structure. For example, in the special case \( f(\sum_t \alpha_t u_t \otimes v_t) = \|Y - \sum_t \alpha_t u_t \otimes v_t\|_F^2 \) for a given matrix \( Y \), and atoms \( \mathcal{A}_1 = \mathcal{A}_2 = \{ x : \| x \|_2 = 1 \} \), problem (7.1) becomes the standard SVD. With \( f \) as choice of Bregman divergences other than the least squares, we also recover variants of exponential family PCA [11].

Typical structures of interest for matrix factorizations include sparsity of the factors in various forms, including group/graph structured sparsity (see [5] and references therein for more examples). Furthermore, non-negative factorizations are widely used, also ordered vectors and several other structures on the atom vectors. In our framework, it is easy to also use combinations of several different vector structures. Also, note that the sets \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) are by no means required to be of the same structure. For the rest of the paper, we assume that the sets \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) are compact. Note that none of our convergence rates holds for the non-negative matrix factorization case. On the other hand, since we allow the weights to be negative, we can assume the set to be symmetric.
Algorithm. The main matrix greedy algorithm derived from Algorithm 6 applied to problems of form (7.2) is presented in Algorithm 8.

Algorithm 8 Generalized Matrix Greedy (GMG)

Require: \( u_0, v_0, T, S_1 = \{ u_0 \}, S_2 = \{ v_0 \}, X_0 = u_0 v_0^\top \)
1: for \( t = 1..T \) do
2: \( u_t, v_t := \text{Approx}\text{-LMO}_{A_1 \otimes A_2} (\nabla f(X_{t-1})) \)
3: \( S_1 := S_1 \cup u_t; S_2 := S_2 \cup v_t \)
4: \( B := X_{t-1} - \frac{1}{t} \nabla f(X_{t-1}) \)
5: Update \( X_t := \arg \min_{Z \in \text{lin}(S_1 \otimes S_2)} \| Z - B \|_2^2 \)
6: Optional: Correction of some/all \( u_i, v_i \).
7: end for

In practice, the atom-correction step (Step 6) is particularly important for maintaining iterates of even smaller rank in practice, as also highlighted by our experiments. Local corrections are made to the already chosen set of atoms to potentially improve the quality of the (current) rank-\( t \) solution.

Matrix completion. Variants of (structured) matrix completion are obtained for the objective function
\[
f \left( \sum_i \alpha_i u_i \otimes v_i \right) = \| Y - \sum_i \alpha_i u_i \otimes v_i \|_\Omega^2,
\]
where \( \Omega \) is set of observed indices. Here the norm on the vector space is defined with respect only to the observed entries. Formally, \( \| Z \|_\Omega^2 = \| Z_\Omega \|_F^2 \) is induced by the inner product \( \langle A, B \rangle_\Omega := \text{tr}(A_\Omega^\top B_\Omega) \). Note that (7.3) can be generalized by using Bregman divergences other than the least squares to recover exponential family matrix completion [25] (See Table 7.1).

Convergence. The linear rate of convergence proved in Theorem 13 is directly applicable to Algorithm 8 provided that a linear oracle (LMO) is available for the used atoms.

Generalized rank for structured factorizations. For the case of given \( y \in \text{lin}(A) \), the number of iterations performed by Algorithm 8 can be thought of as a complexity measure of generalized matrix rank, specific to our objective function \( f \) and the atomic sets.

7.1.1 Designing the LMO

The LMO is required to solve a non-convex problem of finding a best rank-1 approximation as per the structural constraint. We propose use of a generalization of...

the standard power method to design the LMO for any structure inducing sets. Say $X_t \in \mathbb{R}^{n \times m}$ is the current iterate of Algorithm 8. In the symmetric case (the non-symmetric case is analogous), for arbitrary $A_1$, the LMO can be equivalently written as finding the vector which solves

$$\arg \max_{u \in A_1} \langle -\nabla f(X_t), u \otimes u \rangle.$$  \hspace{1cm} (7.4)

We solve for $u$ iteratively. Say $u^{(t)}$ is the $t$th iterate ($u^{(0)} \in \mathbb{R}^n$ is the initialization). The next iterate is obtained as

$$u^{(t+1)} \leftarrow \arg \max_{u \in A_1} \langle u, -\nabla f(X_t)u^{(t)} \rangle.$$  \hspace{1cm} (7.5)

We call the update step (7.5) an atomic power iteration. It is easy to see that it recovers the standard power method as a special case, as well as the Truncated Power Method for sparse PCA suggested by [67], the sparse power methods suggested by [43], and the cone constrained power method suggested by [15]. It can be shown that the iterations are Frank-Wolfe steps with fixed step size 1 and that the iterates monotonically increase the function value.

7.1.2 Atom Correction Variants

Algorithms 6 and 8 guarantee linear convergence in terms of number of calls made to the LMO oracle, each iteration increasing the rank of the iterate by one. In many applications, such as low-rank PCA, low-rank matrix completion etc., it is desirable to have iterates being a linear combination of only as few atoms as possible. As discussed in the previous Section 5.1, we do allow for corrections to possibly obtain a much lower function cost with a given fixed rank approximation. The more severe corrections of atoms themselves in step 4 (as opposed to just their weights) can be made by updating them one or a few atoms at a time, keeping the rest of them fixed. For the symmetric case, the update of the $i$th atom can be written as

$$u_i^+ := \arg \min_{u \in A_1} f \left( \sum_{j \neq i} \alpha_j u_j \otimes u_j + \alpha_i u \otimes u \right).$$  \hspace{1cm} (7.6)

The update for non-symmetric case is analogous. The complexity of atom corrections depends very strongly on the structure of the used atomic sets $A_1$ and $A_2$. One general way is to call the LMO again, but assuming the current iterate is $\sum_{j \neq i} \alpha_j u_j \otimes u_j$. For the non-symmetric case, techniques such as alternating minimization between $u_i$ and $v_i$ are also useful if the call to the LMO is more expensive. Note that for the nuclear norm special case $A_1 = A_2 = \{ x : \|x\|_2 = 1 \}$, variants of such corrections of the atoms were studied by [40].

In contrast to the non-convex corrections of atoms $u_i$, the optimal updates of the weights $\alpha$ alone as in line 3 can be performed very efficiently after every atom update (as e.g. in OMP), by simply solving a linear system of size $t \times t$. 

40
7.1.3 Implementing the Linear Minimization Oracle for Matrix Factorizations

[35] discusses several cases of the LMO implementation for matrix factorization problems. In particular, the atomic power method becomes the method of choice, as it is applicable to any structure of the two sets of atoms $A_1$ and $A_2$. We will discuss the more general case of LMO for tensor factorization problems below in Section 7.2.2.

7.2 Tensor Factorization

In this section, we extend part of the work presented earlier in this chapter to adapt it to solve the structured tensor factorization. Therefore, we use the tensor case to discuss the practical implementation of our Generalized Matching Pursuit Algorithm (Algorithm 6) for the cases of structured tensor factorization and completion. Since we want to minimize general convex and smooth functions, the weights update might not boil down to a linear system. For such a reason we introduced the Uniformly Corrective Matching Pursuit in Algorithm 6 which perform the optimization of the weights by minimizing a quadratic surrogate of the function given by smoothness (Equation (5.3)). It is easy to show that:

Proposition 41. Given a set $S \subset \mathcal{H}$ and a smooth convex function $f$, the projected gradient step on the set $\text{conv}(S)$ is:

$$
\text{PG}_{\text{conv}(S)}(x_t) := \arg \min_{z \in \text{conv}(S)} \| z - (x_t - \frac{1}{\ell} \nabla f(x_t)) \|_H^2
$$

$$
= \arg \min_{z \in \text{conv}(S)} g_{x_t}(z)
$$

where $g_{x_t}$ is the quadratic upper bound of $f$ computed in $x_t$ as presented in Equation (5.3).

In general, projected gradient descent and alternating minimization are two alternatives to our method. On the other hand, they require expensive projections at each iteration which makes them unattractive for the minimization over structured sets.

7.2.1 Generalised Tensor Greedy Algorithm

We now leverage the general algorithm framework introduced in Chapter 5 for structured tensor factorizations. As we did for matrices, in order to encode interesting structure for tensor factorizations, we consider the set of rank-1 tensors $A_1 \otimes A_2 \otimes \ldots \otimes A_N$ as atom set, which is simply constructed by N arbitrary sets of vector atoms $A_n \subset \mathbb{R}^{I_n}$ $\forall n \in [N]$. Specializing the general optimization problem (5.1) to the set $\text{lin}(A_1 \otimes A_2 \otimes \ldots \otimes A_N)$, we obtain the following structured tensor factorization

Given an objective function \( f : \mathbb{R}^{I_1 \times \cdots \times I_N} \rightarrow \mathbb{R} \), we want to find a tensor \( \mathcal{X} \) optimizing

\[
\min_{\mathcal{X} \in \text{lin}(A_1 \otimes \cdots \otimes A_N)} f(\mathcal{X}).
\]

When restricting (7.7) to candidate solutions of rank at most \( T \), we obtain the following equivalent and more interpretable factorized reformulation:

\[
\min_{z^{(n)} \in A_n \forall n \in [N], \lambda \in \mathbb{R}^T} f\left( \sum_{i=1}^{T} \lambda_i z_i^{(1)} \otimes \cdots \otimes z_i^{(N)} \right). \tag{7.8}
\]

For tensors we improperly use the term \textit{rank} to describe the number of atoms used to compute the approximation as we do for matrices, therefore the number of iterations \( T \) is an upper bound on the rank of the tensor. Unfortunately, the tensor rank is a very difficult problem which is not tackled by this work. Indeed we are not looking for an orthogonal decomposition as in [36].

**Algorithm.** The main tensor greedy algorithm derived from Algorithm 6 applied to problems of form (7.8) is presented in Algorithm 9.

**Algorithm 9 Generalized Tensor Greedy (GTG)**

\[\text{Require: } z^{(n)}_0 \forall n \in [N], T, S_0 = \{ z^{(n)}_0 \}, X_0 = z^{(1)}_0 \otimes \cdots \otimes z^{(N)}_0\]

\[\text{1: for } t = 1, T \text{ do }\]

\[\text{2: } z^{(1)}_t, \ldots, z^{(N)}_t := (\text{Approx})-\text{LMO}_{A_1 \otimes \cdots \otimes A_N}(\nabla f(X_{t-1})) \]

\[\text{3: } S_t := S_0 \cup z^{(n)}_t \forall n \in [N]; \]

\[\text{4: } B := X_{t-1} - \frac{1}{t} \nabla f(X_{t-1}) \]

\[\text{5: Update } X_t := \arg \min_{Z \in \text{lin}(S_t \otimes \cdots \otimes S_t)} \|Z - B\|_2^2 \]

\[\text{6: Optional: Correction of some/all } z^{(n)}_t.\]

\[\text{7: end for}\]

**Discussion** Already in the matrix case, the atom-correction step (Step 6) was important for maintaining iterates of smaller rank. For tensors, it is particularly fundamental. Note that the local corrections are made to the already chosen set of atoms to potentially improve the quality of the (current) approximation while keeping the number of factors fixed. In some sense, we are performing a step of the CP decomposition using \( S \) as a warm start to decrease the objective value.

While for matrices the best rank-\( k \) approximation is related to the best rank-1 approximation by the Eckart-Young decomposition [20], this is not the case for tensors. As shown in [36] there are examples in which the best rank one approximation of a tensor is not a factor of its best rank two approximation. It follows that the correction steps of the weights and of the atoms are particularly important in the tensor case. Indeed, in the CP decomposition all the rank-1 tensors have to be found simultaneously as explained in [37].
7.2. Tensor Factorization

Tensor completion. Variants of (structured) tensor completion for a partially observed $N$-order tensor $Y$ are obtained for the objective function

$$f\left(\sum_i \lambda_i z_i^{(1)} \otimes \ldots \otimes z_i^{(N)}\right) = \|Y - \sum_i \lambda_i z_i^{(1)} \otimes \ldots \otimes z_i^{(N)}\|^2_{\Omega}$$

where $\Omega$ is set of observed indices. Here the norm on the vector space is defined with respect to only the observed entries. Therefore, $\|Z\|^2_{\Omega} = \|Z_{\Omega}\|_F$.

7.2.2 Implementing the Linear Minimization Oracle for Tensor Factorizations

The LMO is required to solve a non-convex problem of finding a best rank-1 approximation as per the structural constraint (i.e. the constraints on the optimization problem of Equation (7.8)). We propose use of a generalization of the standard power method to design the LMO for any structure inducing sets. Say $X_t \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ is the current iterate of Algorithm 6. The LMO can be equivalently written as finding the vectors which solve

$$\arg \max_{z^{(n)} \in A_n \forall n \in [N]} \langle z^{(n)}, -\nabla f(X_t) \rangle$$

We solve for each atom in an alternating fashion. Say $z^{(n)}_t$ is the $t$th iterate ($z^{(n)}_0 \in A_n$ is the initialization). The next iterate is obtained as

$$z^{(n)}_{t+1} \leftarrow \arg \max_{z^{(n)} \in A_n} \langle z^{(n)}, -\nabla f(X_t)z^{(N)}_t \otimes \ldots \otimes z^{(N)}_{n+1} \otimes Kr \ldots \otimes Kr z^{(1)}_{t+1} \rangle.$$  \hspace{1cm} (7.10)

As we did in the matrix case, we call the update step (7.10) an atomic power iteration. The update of Equation (7.10) has to be performed for all $1 \leq n \leq N$ one atom at the time. For simplicity we start with $n = 1$ and proceed in order. Recall that $\nabla f(X_t)_{(n)}$ is the unfolding of the gradient along the $n$-th dimension, hence, it is a matrix. The new atom at iteration $t + 1$ is obtained using the other atoms computed at the previous iteration $t$ as depicted in Equation (7.10). It is easy to see that this procedure recovers the power method we introduced for matrices if $N = 2$ (Equation (7.5)).

An alternative approach is to use the even more fresh atoms already obtained during this round, that is the $t + 1$-th for the $n$'s we have already computed (i.e. $n' < n$):

$$z^{(n)}_{t+1} \leftarrow \arg \max_{z^{(n)} \in A_n} \langle z^{(n)}, -\nabla f(X_t)_{(n)}z^{(N)}_{t} \otimes \ldots \otimes z^{(N)}_{n+1} \otimes Kr \ldots \otimes Kr z^{(1)}_{t+1} \rangle.$$  \hspace{1cm} (7.10)

Since both approaches solve the same problem, we always refer to (7.10) for simplicity.

A similar power method decomposing a tensor was presented in [1] and then extended to structured decomposition by [53]. Their approach uses a power method.

directly on the tensor to be factorized rather than to the gradient. Doing so, they obtain multiple rank-one tensors which are then clustered in \( T \) clusters where \( T \) is the rank of the approximation. The centroids of such clusters represent the \( T \) rank-one tensors to be used in the approximation.

7.2.3 Optional Correction of Selected Atoms

The atoms’ correction attempts to reduce the value of the objective function \( f \), while the rank is kept fixed. This is by itself an hard non-convex problem. To approximate the solution we can sequentially optimize one atom at the time keeping the other to their old values. In the case of the distance function and no constraints this is equivalent to one iteration of the CP decomposition with rank \( t \). As done in the CP decomposition this process can be repeated multiple times. In other words if we want to correct \( z_i^{(n)} \) we solve the following optimization problem:

\[
(z_i^{(n)})_{\text{new}} = \arg\min_{z_i^{(n)}} f\left( \sum_{i \neq j} \lambda_j z_j^{(1)} \otimes \ldots \otimes z_j^{(N)} + \lambda_i z_i^{(1)} \otimes \ldots \otimes z_i^{(n-1)} \otimes z_i^{(n)} \otimes z_i^{(n+1)} \otimes \ldots \otimes z_i^{(N)} \right)
\]

(7.11)

It is interesting to relate the atoms’ correction with the atomic power method in case of the distance function. Let us consider the optimization problem of (7.11).

**Theorem 42.** Let \( f : \mathbb{R}^{I_1, \ldots, I_N} \rightarrow \mathbb{R} \) be the distance function (i.e. \( f(X) = \|Y - X\|_F^2 \) for a given tensor \( Y \)). Let also \( \mathcal{R}_i := Y - \sum_{j \neq i} \lambda_j z_j^{(1)} \otimes \ldots \otimes z_j^{(N)} \). If the atoms in \( \mathcal{A}_n \forall n \in [N] \) have fixed norm \( \epsilon \) then, an atomic power iteration step is an alternating minimization step, which is to say, the following two optimization problems have the same solution:

\[
\arg\min_{z_i^{(n)}} \left( -\operatorname{sgn}(\lambda_i) \mathcal{R}_i \right), \lambda_i z_i^{(1)} \otimes \ldots \otimes z_i^{(n-1)} \otimes z_i^{(n)} \otimes z_i^{(n+1)} \otimes \ldots \otimes z_i^{(N)}
\]

(7.12)

\[
\arg\min_{z_i^{(n)}} \| \mathcal{R}_i - \lambda_i z_i^{(1)} \otimes \ldots \otimes z_i^{(n-1)} \otimes z_i^{(n)} \otimes z_i^{(n+1)} \otimes \ldots \otimes z_i^{(N)} \|_F^2
\]

(7.13)

**Discussion** With Theorem 42 we have shown how, in the case of the distance function we can solve the atoms’ correction that was introduced as the optimization problem of (7.11) using a single atomic power iteration. On the other hand, when \( i = t \) the optimization problem that is solved by the LMO (\( \nabla f(X_{t-1}) \)) becomes an alternating minimization problem assuming \( \lambda_t > 0 \). In particular, since each power iteration has an equivalent alternating minimization step if we start from...
7.2. Tensor Factorization

the same vectors solving LMO(−Rt) has the same solution of solving:

$$\arg\min_{\mathbf{z}^{(n)} \in \mathcal{A}_n \forall \ n \in [N]} \left\| \mathbf{R}_t - \lambda_t \mathbf{z}^{(1)} \otimes \ldots \otimes \mathbf{z}^{(N)} \right\|_F^2$$

using alternating minimization and with $\lambda_t > 0$. This fact gives some important insights on the behavior of the LMO. Note how the LMO is implicitly assuming that $\lambda_t > 0$. Therefore, the LMO is making an additional assumption on the sign of the weight of the new factor. Since the function is convex the negative gradient points in the direction of the global minimum. Assuming a positive weight for the atom that we obtain when we minimize the inner product with the gradient means that we are indeed descending the objective. For such a reason, the algorithm always decreases the value of the objective at every iteration, unless no progress is made.

**Proof of Theorem 42**

**Proof.** First of all, note that for a general direction $\mathbf{T} \in \mathbb{R}^{I_1 \cdot \ldots \cdot I_N}$ the LMO solution is invariant to re-scaling:

$$\text{LMO}(\mathbf{T}) = \arg\min_{\mathbf{z}^{(n)} \in \mathcal{A}_n \forall \ n \in [N]} \langle \mathbf{z}^{(1)} \otimes \ldots \otimes \mathbf{z}^{(N)}, \mathbf{T} \rangle$$

$$= \arg\min_{\mathbf{z}^{(n)} \in \mathcal{A}_n \forall \ n \in [N]} \langle \mathbf{z}^{(1)} \otimes \ldots \otimes \mathbf{z}^{(N)}, \mathbf{T} \rangle \psi$$

$\forall \ \psi \in \mathbb{R}_{>0}$.

In particular, for the power iteration for $\mathbf{z}^{(n)}$, let us define:

$$\Pi_n := \mathbf{z}^{(N)}_n \otimes K \ldots \otimes K \mathbf{z}^{(n+1)}_n \otimes K \mathbf{z}^{(n-1)}_n \otimes K \ldots \otimes K \mathbf{z}^{(1)}_n.$$  

We then have:

$$\mathbf{z}^{(n)}_{i+1} = \arg\min_{\mathbf{z}^{(n)} \in \mathcal{A}_n} \langle \mathbf{z}^{(n)}, \mathbf{T}_{(n)} \Pi_n \rangle$$

$$= \arg\min_{\mathbf{z}^{(n)} \in \mathcal{A}_n} \langle \mathbf{z}^{(n)}, \mathbf{T}_{(n)} \Pi_n \rangle \psi$$

Let us now consider the atoms’ update optimization problem:

$$\left( \mathbf{z}_{i}^{(n)} \right)^{\text{new}} = \arg\min_{\mathbf{z}^{(n)} \in \mathcal{A}_n} \||\mathbf{Y} - \sum_{i \neq j} \lambda_j \mathbf{z}_{i}^{(1)} \otimes \ldots \otimes \mathbf{z}_{j}^{(N)} + \right. \left. - \lambda_i \mathbf{z}_{i}^{(1)} \otimes \ldots \otimes \mathbf{z}_{i}^{(n-1)} \otimes \mathbf{z}_{i}^{(n+1)} \otimes \ldots \otimes \mathbf{z}_{i}^{(N)} \parallel F^2 \right.$$
that is a step of alternating minimization that is solved as:

\[
\begin{align*}
    \left( z^{(n)}_i \right)_{\text{new}} &= \mathbb{P}_{\mathcal{A}_n} \frac{\mathcal{R}_{i(n)} \Pi_n}{\|\Pi_n\|^2 \lambda_i} \\
    &= \mathbb{P}_{\mathcal{A}_n} \frac{\mathcal{R}_{i(n)} \Pi_n}{c^2 \lambda_i} \\
    &= \arg\min_{z^{(n)} \in \mathcal{A}_n} \|z^{(n)} - \frac{\mathcal{R}_{i(n)} \Pi_n}{c^2 \lambda_i}\|^2 \\
    &= \arg\min_{z^{(n)} \in \mathcal{A}_n} \|z^{(n)}\|^2 + \left\| \frac{\mathcal{R}_{i(n)} \Pi_n}{c^2 \lambda_i} \right\|^2 - 2 \langle z^{(n)}, \frac{\mathcal{R}_{i(n)} \Pi_n}{c^2 \lambda_i} \rangle \\
    &= \arg\min_{z^{(n)} \in \mathcal{A}_n} \langle z^{(n)}, -\frac{\mathcal{R}_{i(n)} \Pi_n}{c^2 \lambda_i} \rangle \\
    &= \arg\min_{z^{(n)} \in \mathcal{A}_n} \langle z^{(n)}, -\sgn(\lambda) \mathcal{R}_{i(n)} \Pi_n \rangle \\
    &= \arg\min_{z^{(n)} \in \mathcal{A}_n} \langle z^{(n)}, -\sgn(\lambda) \mathcal{R}_{i(n)} \Pi_n \rangle \\
    &= -\sgn(\lambda) \mathcal{R}_{i(n)} \Pi_n \end{align*}
\]

hence the equivalence when we call \( T_i = -\sgn(\lambda_i) \mathcal{R}_{i(n)} \) and \( \psi = c^2 |\lambda_i| \). \( \square \)
Chapter 8

Experiments: Proof of Concept

In this chapter, we present an experimental proof of concept on the Uniformly Corrective Matching Pursuit Algorithm that we introduced in this work. A more extensive test is left as future work. The goal of this chapter is to briefly present some practical example of the algorithm and it is not meant to be an extensive test between the proposed method with the literature even though we do show how it compares to some other completion and factorization algorithms.

8.1 Tensor Factorization

We used different datasets for this evaluation. We used synthetic data to discuss the accuracy of the low-rank approximation. We further use two different datasets. First of all we used the L1000 dataset, which has been preprocessed and given to us in the form of a three dimensional tensor\(^1\). The tensor has dimensions of 978 genes by 11 cell types by 611 drugs. Each column along the gene dimension (for a fixed drug and cell type) corresponds to differential expression profile induced by that drug in that cell type. Large positive numbers correspond to strong up-regulation, and vice versa. The profiles are normalized to each have unit norm in gene space. In such a tensor some entire columns along the gene dimension are missing and the goal is to reconstruct them.

We also used the MRI from the KNIX dataset to test the low rank approximation. This dataset contains 24 MRI of a knee on different levels which are stacked to form a tensor of size \(512 \times 512 \times 24\) where each pixel intensity has been divided by the mean intensity.

All the results on tensor factorization and completion are reported as root mean squared errors over 10 runs with different random initialization for both the iterates and the atomic power method.

\(^1\)We warmly thank Rachel Hodos (NYU) for giving us the processed data.
First of all, we consider the low rank approximation problem of a random $30 \times 20 \times 50$ synthetic tensors generated with rank 18 and approximated with rank 6, 10 and 20. As we introduced in Chapter 6, the unconstrained decomposition can be solved using the CP decomposition. While with GTG (Algorithm 9) we still minimize the distance function (as in the CP decomposition) $f(X) = \|Y - X\|_2$ we use the constraints that $A_1 = \{a \in \mathbb{R}^{30} : \|a\| = 1\}$, $A_2 = \{a \in \mathbb{R}^{20} : \|a\| = 1\}$ and $A_3 = \{a \in \mathbb{R}^{50} : \|a\| = 1\}$. We also test our algorithm on the structured factorization problem with factors which are 20% sparse: $A_1 = \{a \in \mathbb{R}^{30} : \|a\| = 1, \|a\|_0 = 24\}$, $A_2 = \{a \in \mathbb{R}^{20} : \|a\| = 1, \|a\|_0 = 16\}$ and $A_3 = \{a \in \mathbb{R}^{50} : \|a\| = 1, \|a\|_0 = 40\}$.

We compare the structured factorization with [53] (using the same 20% sparse atoms set) that proposes a sparse power method for tensor factorization (SPM). They used only a set of rank-1 approximations from multiple starting points which are then clustered. Therefore, we found this approach not suitable when the rank of the approximation is large. The clustering of the multiple rank-one approximations is done using $k$-means to obtain $T$ factors which form the factorization. We also compare against the Power Method (PM) of [11] which is essentially the same algorithm of [53] but with dense atoms (i.e. $A_1$, $A_2$ and $A_3$ are the $L2$ ball as we did for GTG).

For both the methods the number of initializations is $\max(10, R^3)$ as suggested in [53] but the number of power iterations is fixed to 100 which is the same number we use in our power method. The result is depicted in Table 8.1. For GTG (Algorithm 9) we use the optional correction of the atoms performed only once per atom (see Section 7.2.3). Note that since we are minimizing the distance function the correction is efficiently done by a single atomic power iteration.

<table>
<thead>
<tr>
<th></th>
<th>rank 6</th>
<th>rank 10</th>
<th>rank 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTG</td>
<td>3.1184±0.000384</td>
<td>2.4341±0.0010258</td>
<td>0.025563±0.00097192</td>
</tr>
<tr>
<td>CP</td>
<td>3.0879±0.0050224</td>
<td>2.3817±0.0040547</td>
<td>0.00028215±2.5094e-07</td>
</tr>
<tr>
<td>GTG 20% sparse</td>
<td>3.1202±0.003757</td>
<td>2.4521±0.01484</td>
<td>0.50092±7.8678e-05</td>
</tr>
<tr>
<td>PM</td>
<td>3.7263±0.013027</td>
<td>3.4339±0.020805</td>
<td>-</td>
</tr>
<tr>
<td>SPM 20% sparse</td>
<td>3.7001±0.019972</td>
<td>3.2803±0.036796</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 8.1: Synthetic data, true rank 18 approximated with rank 6, 10 and 20. Values are mean and variance over 10 runs.

While the true rank is 18, our algorithm does not have an error which is as close to zero as the CP decomposition. The reason is the greedy approach that we use. On the other hand it is interesting to note how the residual significantly decreases, as depicted in Figure 8.1. Neither the CP nor the GTG achieve zero error being the optimization problem we want to solve NP-hard.

We also note that the reconstruction error computed as RMSE drastically change when reaching the true rank as depicted in Figure 8.2.

Then, the same tensor is made sparse by deleting 50% of the entries. Result is depicted in Table 8.2.
8.1. Tensor Factorization

Figure 8.1: Residual of the approximation with rank 20 of a tensor of rank 18

Figure 8.2: RMSE of the approximation with rank 20 of a tensor of rank 18
8. Experiments: Proof of Concept

<table>
<thead>
<tr>
<th></th>
<th>rank 6</th>
<th>rank 10</th>
<th>rank 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTG</td>
<td>2.5379 ± 1.6992e-04</td>
<td>2.3097 ± 2.6814e-04</td>
<td>1.9285 ± 1.2429e-05</td>
</tr>
<tr>
<td>CP</td>
<td>2.5206 ± 1.5938e-04</td>
<td>2.3024 ± 5.4672e-04</td>
<td>1.9268 ± 2.9401e-06</td>
</tr>
<tr>
<td>GTG 20% sparse</td>
<td>2.5386±0.00028842</td>
<td>2.3149±0.00027012</td>
<td>1.9467±1.2308e-05</td>
</tr>
<tr>
<td>PM</td>
<td>2.7683±0.0011029</td>
<td>2.74±0.011627</td>
<td>-</td>
</tr>
<tr>
<td>SPM 20% sparse</td>
<td>2.7589±0.0016665</td>
<td>2.722±0.0043248</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 8.2: Synthetic data, 50% sparse approximated with rank 6, 10 and 20 for GTG, CP and GTG with 20% sparse components. Values are mean and variance on 10 runs.

Figure 8.3: Difference between residual with and without reoptimization

Then, we tried our method on the KNIX dataset where we compare the CP decomposition with our GTG constrained with the sets $A_1$, $A_2$ and $A_3$ being the $L_2$ ball. We again use the correction of the atoms which plays an important role in decreasing the objective value also in real world data as depicted in Figure 8.3. The RMSE is depicted in Table 8.3.

<table>
<thead>
<tr>
<th></th>
<th>rank 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTG</td>
<td>0.3601 ± 8.0905e-07</td>
</tr>
<tr>
<td>CP</td>
<td>0.3529 ± 1.3448e-06</td>
</tr>
</tbody>
</table>

Table 8.3: KNIX data. Values are mean and variance on 10 runs.
8.1.1 Tensor Completion

Tensor completion is an important problem which can be solved within our framework. Furthermore, tensor completion is NP-hard already in the matrix case. We performed the tensor completion in the L1000 dataset and compare our result with the Fast Low Rank Tensor Completion algorithm (FaLRTC). Their approach involves the matricization of the tensor. Since entire columns are missing in one of the unfolding the matrix is filled with the mean and then a low rank approximation is computed. We used an approximation of rank 100. Since the atoms’ correction becomes more expensive the more iteration we perform, we do not use the correction until the last iteration, where is repeated 100 times. We again use the L2 ball for the sets $A_1$, $A_2$ and $A_3$.

The results are depicted in Table 8.4.

<table>
<thead>
<tr>
<th></th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTG rank 100</td>
<td>0.0284 ± 1.1317e-08</td>
</tr>
<tr>
<td>FaLRTC</td>
<td>0.0285 ± 6.8654e-10</td>
</tr>
</tbody>
</table>

Table 8.4: L1000 data. Values are mean and variance on 10 runs

8.2 Additional Experiments on Matrices

Additional experiments have been performed in [35] to test the performance of Algorithm for several matrix factorization problems.
Chapter 9

Conclusion and Future Work

We presented a greedy framework for structured matrix and tensor decomposition. We explored the landscape of the greedy optimization methods such as Frank-Wolfe, Matching Pursuit and Boosting showing how they relate to each other with their similarities and dissimilarities. Our novel approach allows controlling the tradeoff between rank and approximation quality. We presented convergence analysis and empirical evaluation for the same. For future work, we wish to extensively test the Algorithm on a wider class of real-world problem, such as representation learning for tensorial data and using different functions. From the theoretical point of view, we want to further study the assumptions on the atom set and possibly relate them with the complexity measures of $f$ in order to obtain affine invariant convergence rates. We also want to bring this theory further to nonconvex functions. It is already clear that, at least for some easier nonconvex function, Matching Pursuit exhibits sublinear convergence [64]. An interesting research direction is to include the possibility of an adaptive step size which can improve the constants in the convergence rates. Last but not least, the atomic power method that we use to approximate the LMO represent an open problem for the community. Giving guarantees on the error of the atomic power method (either multiplicative or additive error) would complete the analysis of the Algorithm in real world applications where an exact LMO is not available.
Appendix A

A.1 Proofs of the Main Results

A.1.1 Proof of Sublinear Convergence Rates

Frank-Wolfe Algorithm variants

**Theorem** Let $A \subset H$ be a bounded set and let $f : H \to \mathbb{R}$ be a $L$-smooth and convex function. Then, the Uniformly Corrective Frank-Wolfe method (Algorithm 4) when using an LMO of duality-gap-accuracy parameter $\delta \in (0,1]$ (Equation (5.4)) converges for $t \geq 0$ as

$$f(x_t) - f(x^*) \leq 2 \frac{L \text{diam}(A)^2}{\delta (t + 2)} + \epsilon_0 \delta t + 2 \epsilon_0$$

where $\epsilon_0 := f(x_0) - f(x^*)$ is the initial error in objective.

**Proof.** This proof can be seen as a slight modification of [39, Theorem C.1] adapted to the uniformly corrective Frank-Wolfe rather than the original variant. Define $f_* := f(x^*)$. Consider the $t$th iteration with $t \geq 0$. From the definition (5.4) of the relative error on the duality gap, we have:

$$\langle \nabla f(x_t), x_t - z_t \rangle \geq \delta \langle \nabla f(x_t), x_t - x^* \rangle \tag{A.1}$$

$$\geq \delta (f(x_t) - f_*) =: \delta \epsilon_t \tag{A.2}$$

where we used the fact that $x^* \in \text{conv}(A)$ to obtain (A.1), and (A.2) follows from the convexity of $f$. Let $g_{x_t}(x)$ be the quadratic approximation of $f$ at $x_t$ given by Equation (5.3). Therefore,

$$f(x_{t+1}) \leq \min_{0 \leq \lambda \leq 1} g_{x_t}(x_t + \lambda (z_t - x_t))$$

$$\leq f(x_t) + \min_{0 \leq \lambda \leq 1} \left\{ -\lambda \delta \epsilon_t + \frac{L \lambda^2}{2} \text{diam}(A)^2 \right\}, \tag{A.3}$$

55
where again \( \epsilon_t := f(x_t) - f(x^*) \), and the last inequality is implied by Equation (A.2) and \( \| \hat{z}_t - x_t \|_2 \leq \text{diam}(A) \) by definition of the diameter, and by the fact that for convex hulls we have \( \text{diam}(A) = \text{diam}(\text{conv}(A)) \), see e.g. [70]. Subtracting \( f_* \) from both sides of (A.3) yields:

\[
\epsilon_{t+1} \leq \epsilon_t + \min_{0 \leq \lambda \leq 1} \left\{ -\lambda \delta \epsilon_t + \frac{L \lambda^2}{2} \text{diam}(A)^2 \right\}
\]

Finally, as using a similar induction argument as in [39] we obtain

\[
\epsilon_t \leq \frac{2 \left( \frac{L \text{diam}(A)^2}{\delta^2} + \epsilon_0 \right)}{\delta t + 2}
\]

for \( t \geq 0 \).

\[\square\]

Matching Pursuit Algorithm Variants.

**Theorem** 7. Let \( A \subset \mathcal{H} \) be a bounded and symmetric set and let \( f : \mathcal{H} \to \mathbb{R} \) be a \( L \)-smooth convex function. Let

\[
\rho := \max \{ \| x^* \|_A, \| x_0 \|_A, \ldots, \| x_T \|_A \} < \infty
\]

If the optimum is not unique we consider \( x^* \) to be the one with largest atomic norm. Then, Matching Pursuit and Uniformly Corrective Matching Pursuit (Algorithms 5 and 6) with a relative accuracy \( \delta \in (0, 1] \) on the LMO optimization problem (Equation (5.5)) do converge for \( t \geq 1 \) as

\[
f(x_t) - f(x^*) \leq \frac{2L \rho^2 \text{diam}(A)^2}{\delta^2(t + 2)}.
\]

**Proof.** Define \( f_* := f(x^*) \). Consider the \( t^{th} \) iteration with \( t \geq 0 \). The new iterate of both algorithms is obtained minimizing the quadratic approximation of \( f \) at \( x_t \) given by Equation (5.3). The following inequality holds for the Matching Pursuit variant (Algorithm 5):

\[
f(x_t) \leq \min_{x \in \text{lin}(A)} g_{x_t}(x) \leq \min_{\lambda \in \mathbb{R}} g_{x_t}(x_t + \lambda \hat{z}_t)
\]

The same also is true for the uniformly corrective variant of MP (Algorithm 6), since the improvement in every step will be at least as good as by plain MP (Algorithm 5).

By symmetry of \( A \) and by definition of atomic norm both \( x_t \) and \( x^* \) are in \( \rho \cdot \text{conv}(A) \). Therefore:

\[
\langle \nabla f(x_t), -2 \rho \hat{z}_t \rangle = \langle \nabla f(x_t), -\rho \hat{z}_t \rangle + \langle \nabla f(x_t), -\rho \hat{z}_t \rangle \geq \langle \nabla f(x_t), -\rho z_t \rangle + \langle \nabla f(x_t), -\rho z_t \rangle \geq \langle \nabla f(x_t), x_t - x^* \rangle \geq f(x_t) - f_* =: \epsilon_t
\]
A.1. Proofs of the Main Results

where (A.4) follows from the definition of inexactness of the LMO (Equation (5.5)) and (A.5) from the fact that $-\rho z_t$ has the smallest inner product with the positive gradient with respect to all the elements in $\text{conv} (\rho A)$. Note that since the set is symmetric both $x_t$ and $x^*$ are in $\text{conv} (\mathcal{A})$. Equation (A.6) (known as weak duality) again follows from the convexity of $f$.

Using the quadratic approximation from smoothness as given by $g_{x_t}$, we get

$$f(x_{t+1}) \leq \min_{\lambda \in \mathbb{R}} g_{x_t} (x_t + \lambda \delta z_t)$$

$$\leq \min_{0 \leq \lambda \leq 1} g_{x_t} (x_t + \lambda \delta z_t)$$

$$\leq \min_{0 \leq \lambda \leq 1} f(x_t) + \lambda \left( \nabla f(x_t), \frac{2\rho}{\delta} \delta z_t \right) + \frac{L}{2} \left\| \frac{2\rho}{\delta} \delta z_t \right\|^2$$

$$\leq f(x_t) + \min_{0 \leq \lambda \leq 1} \left\{ -\lambda \epsilon_t + \frac{L}{2} \frac{\rho^2}{\delta^2} \text{diam}(\mathcal{A})^2 \right\}, \quad (A.7)$$

where the last inequality is implied by (A.6). Subtracting $f_*$ from (A.7) on both sides yields

$$\epsilon_{t+1} \leq \epsilon_t + \min_{0 \leq \lambda \leq 1} \left\{ -\lambda \epsilon_t + \frac{L}{2} \frac{\rho^2}{\delta^2} \text{diam}(\mathcal{A})^2 \right\}$$

$$\leq \epsilon_t - \frac{2}{t+2} \epsilon_t + \frac{L}{2} \left( \frac{2}{t+2} \right)^2 \frac{\rho^2}{\delta^2} \text{diam}(\mathcal{A})^2$$

Finally, we show by induction

$$\epsilon_t \leq 2L \rho^2 \text{diam}(\mathcal{A})^2 \frac{\delta^2 (t+2)}{t+1}$$

for $t \geq 1$.

When $t = 1$ we get $\epsilon_1 \leq \epsilon_0 - \epsilon_0 + \frac{L}{2} \frac{\rho^2}{\delta^2} \text{diam}(\mathcal{A})^2 < \frac{2L}{3} \frac{\rho^2}{\delta^2} \text{diam}(\mathcal{A})^2$. We now prove the induction step assuming $\epsilon_t \leq \frac{2L \rho^2 \text{diam}(\mathcal{A})^2}{\delta^2 (t+2)}$ for all $t > 1$.

$$\epsilon_{t+1} \leq \frac{t}{t+2} \epsilon_t + \frac{L}{2} \left( \frac{2}{t+2} \right)^2 \frac{\rho^2}{\delta^2} \text{diam}(\mathcal{A})^2$$

$$\leq \frac{t}{t+2} \frac{2L \rho^2 \text{diam}(\mathcal{A})^2}{\delta^2 (t+2)} + \frac{L}{2} \left( \frac{2}{t+2} \right)^2 \frac{\rho^2}{\delta^2} \text{diam}(\mathcal{A})^2$$

$$= \frac{2L \rho^2 \text{diam}(\mathcal{A})^2}{\delta^2 (t+2)} \left( t+1 \right) \frac{t+1}{(t+2)^2}$$

$$\leq \frac{2L \rho^2 \text{diam}(\mathcal{A})^2}{\delta^2 (t+3)}$$

\[ \square \]
A. APPENDIX

Theorem 8. Let $A \subset H$ be a bounded and symmetric set and let $f : H \to \mathbb{R}$ be a $L$-smooth convex function. Let $\rho := \max \{ \|x^\star\|_A, \|x_0\|_A, \ldots, \|x_T\|_A \} < \infty$. If the optimum is not unique we consider $x^\star$ to be the one with largest atomic norm. Then, Matching Pursuit and Uniformly Corrective Matching Pursuit (Algorithms 5 and 6) with a relative accuracy $\delta \in (0, 1]$ on the LMO optimization problem (Equation (5.5)) do converge for $t \geq 0$ as:

$$f(x_t) - f(x^\star) \leq 2 \left( \frac{L\rho^2 \operatorname{diam}(A)^2}{\delta} + \epsilon_0 \right) \delta t + 2.$$

Proof. The proof is based on the FW convergence rate on a rescaled convex set that includes both $x^\star$ and $x_t \forall t$ and is inspired by the work presented in [26]. Define $f_* := f(x^\star)$. Consider the $t^{th}$ iteration with $t \geq 0$. As explained in the proof of Theorem 7 it is not restrictive considering only Matching Pursuit (Algorithm 5). As we did at the beginning of the proof of Theorem 7 we have that:

$$\langle \nabla f(x_t), -2\rho z_t \rangle \geq \delta \langle \nabla f(x_t), -\rho z_t \rangle + \delta \langle \nabla f(x_t), -\rho z_t \rangle \quad (A.8)$$

$$\geq \delta \langle \nabla f(x_t), x_t - x^\star \rangle \quad (A.9)$$

$$\geq \delta (f(x_t) - f_*) =: \delta \epsilon_t \quad (A.10)$$

where (A.8) follows from the definition of inexact LMO (Equation (5.5)), (A.9) follows from the symmetry of $A$ and by definition of atomic norm and (A.10) follows from the convexity of $f$.

Using the approximation $g_{x_t}$ we get

$$f(x_{t+1}) \leq \min_{\lambda \in \mathbb{R}} g_{x_t}(x_t + \lambda z_t)$$

$$\leq \min_{0 \leq \lambda \leq 1} g_{x_t}(x_t + \lambda 2\rho z_t)$$

$$\leq \min_{0 \leq \lambda \leq 1} f(x_t) + \lambda \langle \nabla f(x_t), 2\rho z_t \rangle + \frac{L}{2} \lambda^2 \|2\rho z_t\|^2$$

$$\leq f(x_t) + \min_{0 \leq \lambda \leq 1} \left\{ -\lambda \delta \epsilon_t + \frac{L\lambda^2}{2} \rho^2 \operatorname{diam}(A)^2 \right\}, \quad (A.11)$$

where the last inequality is implied by (A.10). Subtracting $f_*$ from (A.11) on both sides yields

$$\epsilon_{t+1} \leq \epsilon_t + \min_{0 \leq \lambda \leq 1} \left\{ -\lambda \delta \epsilon_t + \frac{L\lambda^2}{2} \rho^2 \operatorname{diam}(A)^2 \right\}$$

$$\leq \epsilon_t - \frac{2\delta}{\delta t + 2} \epsilon_t + \frac{L}{2} \left( \frac{2}{\delta t + 2} \right)^2 \rho^2 \operatorname{diam}(A)^2$$

Finally, as done in [39], we show by induction

$$\epsilon_t \leq 2 \left( \frac{L\rho^2 \operatorname{diam}(A)^2}{\delta} + \epsilon_0 \right) \delta t + 2.$$
for $t \geq 0$. When $t = 0$ we get $\epsilon_0 \leq \left( \frac{L \rho^2 \text{diam}(A)^2}{\delta} + \epsilon_0 \right)$. Therefore, the base case holds for every point in $\text{lin}(A)$ analogously to the Frank-Wolfe case. We now prove the induction step assuming $\epsilon_t \leq 2 \left( \frac{L \rho^2 \text{diam}(A)^2}{\delta} + \epsilon_0 \right)$. 

$$
\epsilon_{t+1} \leq \left( 1 - \frac{2\delta}{\delta t + 2} \right) \epsilon_t + \frac{L}{2} \left( \frac{2}{\delta t + 2} \right)^2 \rho^2 \text{diam}(A)^2 
\leq \left( 1 - \frac{2\delta}{\delta t + 2} \right) \frac{2 \left( \frac{L \rho^2 \text{diam}(A)^2}{\delta} + \epsilon_0 \right)}{\delta t + 2} + \frac{L}{2} \left( \frac{2}{\delta t + 2} \right)^2 \rho^2 \text{diam}(A)^2 + \frac{2}{(\delta t + 2)^2} \delta \epsilon_0 
= \frac{2 \left( \frac{L \rho^2 \text{diam}(A)^2}{\delta} + \epsilon_0 \right)}{\delta t + 2} \left( 1 - \frac{2\delta}{\delta t + 2} + \frac{\delta}{\delta t + 2} \right) 
\leq \frac{2 \left( \frac{L \rho^2 \text{diam}(A)^2}{\delta} + \epsilon_0 \right)}{\delta (t+1) + 2}.
$$

\[ \square \]

A.1.2 Proof of Corollary 10

We have

$$
||x_t||_A \leq \frac{||x_t||}{\text{inr(conv}(A))}, \quad (A.12)
$$

for all $t = 1 \ldots T$, and hence $\rho \leq \frac{\rho}{\text{inr(conv}(A))}$, which yields the first upper bound on $f(x_t) - f(x^*)$ in Corollary 10. Here, (A.12) essentially follows from the definition of the atomic norm. A formal proof can be obtained by writing the atomic norm as the value of an $\ell_1$-norm minimization problem [9] and using arguments from the proof of Theorem 2.5 in [52].

To get upper bound on $f(x_t) - f(x^*)$ for $f(x) = \frac{1}{2}d^2(x, y)$, note that for this particular choice of $f$, $x_t - b = y$ and the update step in Algorithm 6 can be written as $x_{t+1} = P_t y$, where $P_t$ is the orthogonal projection operator onto $S$ in iteration $t$. Hence, we have $||x_t|| \leq ||y||$, for all $t \in [T]$, as a consequence of $\|P_t\|_{\text{op}} = 1$.

A.1.3 Proof of Linear Convergence Rates

**Theorem** 13. Let $A \subset \mathcal{H}$ be a bounded set such that $\text{mDW}(A) > 0$ and let the objective function $f : \mathcal{H} \to \mathbb{R}$ be both $L$-smooth and $\mu$-strongly convex. Then, for $t \geq 0$ the suboptimality of the iterates of Matching Pursuit and Uniformly Corrective Matching Pursuit (Algorithm 5 and 6) with a relative accuracy $\delta \in (0, 1]$ on the LMO optimization problem (Equation 5.5) decays exponentially as:

$$
\epsilon_{t+1} \leq \left( 1 - \frac{\mu \delta^2 \text{mDW}(A)^2}{L \text{radius}(A)^2} \right) \epsilon_t
$$
where $\epsilon_t := f(x_t) - f(x^*)$ denotes the suboptimality at step $t$.

The following proof crucially uses the new complexity notion of the minimal intrinsic directional width given in Definition 12. The proof technique is also loosely inspired by the ideas presented in [38] for the Frank-Wolfe case, which however considers bounded constrained optimization problems in the FW context.

**Proof.** Consider the atom $\tilde{z}_t \in \mathcal{A}$ selected by the LMO at iteration $t$. Due to the smoothness property of $f$ it holds that:

$$f(x_{t+1}) \leq \min_{x \in \text{lin}(\mathcal{A})} g_{x_t}(x) \leq \min_{\lambda \in \mathbb{R}} g_{x_t}(x_t + \lambda \tilde{z}_t) = f(x_t) + \langle \nabla f(x_t), \lambda \tilde{z}_t \rangle + \frac{L}{2} \| \lambda \tilde{z}_t \|^2.$$  

where $g$ is the quadratic approximation of the function $f$ presented in Equation (5.3) for the given smoothness parameter $L$. The above bound clearly holds for the Matching Pursuit algorithm variant, and it is therefore easy to see that the same also holds for its uniformly corrective version (as the correction will only improve the objective of the iterate compared to MP).

Solving the above upper bound for $\lambda$ yields $\lambda = -\frac{1}{L} \langle \nabla f(x_t), \tilde{z}_t \rangle$. Subtracting $f(x^*)$ from both sides and replacing the optimal $\lambda$ yields:

$$\epsilon_{t+1} \leq \frac{1}{2L} \left( \langle \nabla f(x_t), \tilde{z}_t \rangle \right)^2$$  

(A.13)

Now writing the definition of strong convexity, we have the following inequality holding for all $\gamma \in \mathbb{R}$:

$$f(x_t + \gamma(x^* - x_t)) \geq f(x_t) + \gamma \langle \nabla f(x_t), x^* - x_t \rangle + \frac{\gamma^2}{2} \| x^* - x_t \|^2$$

We now fix $\gamma = 1$ in the LHS and minimize with respect to $\gamma$ in the RHS:

$$\epsilon_t \leq \frac{1}{2\mu} \left( \langle \nabla f(x_t), \frac{x^* - x_t}{\| x^* - x_t \|} \rangle \right)^2$$

Combining this with (A.13) yields:

$$\epsilon_t - \epsilon_{t+1} \geq \frac{\mu}{L} \left( \frac{\tilde{z}_t}{\| \tilde{z}_t \|} \right)^2 \epsilon_t$$  

(A.14)

Now we crucially use that the optimization is over the linear subspace $\text{lin}(\mathcal{A})$. We will be able to write both inner products in the above fraction as restricted to the linear
A.1. Proofs of the Main Results

Subspace. To do so, we write the gradient vector \( \mathbf{d} := \nabla f(x_t) \) in two components, the parallel as well as the orthogonal part with respect the linear subspace \( \text{lin}(\mathcal{A}) \), that is \( \mathbf{d} = \mathbf{d}_{\parallel} + \mathbf{d}_{\perp} \) for \( \mathbf{d}_{\parallel} \in \text{lin}(\mathcal{A}) \) and \( \langle \mathbf{d}_{\perp}, \mathbf{v} \rangle = 0 \) holds \( \forall \mathbf{v} \in \text{lin}(\mathcal{A}). \)

Therefore, \( \langle \mathbf{d}, \mathbf{v} \rangle = \langle \mathbf{d}_{\parallel}, \mathbf{v} \rangle + \langle \mathbf{d}_{\perp}, \mathbf{v} \rangle = \langle \mathbf{d}_{\parallel}, \mathbf{v} \rangle \forall \mathbf{v} \in \text{lin}(\mathcal{A}). \)

Applying this to both the numerator for \( \mathbf{v} := \tilde{z}_t \in \mathcal{A} \) and the denominator for \( \mathbf{v} := \frac{x^* - x_t}{\|x^* - x_t\|} \in \text{lin}(\mathcal{A}), \) we have the equivalent expression

\[
\epsilon_t - \epsilon_{t+1} \geq \frac{\mu}{L} \left( \frac{\|d_{\parallel}\|}{\|d_{\parallel}\|} \right)^2 \epsilon_t
\]  

(A.15)

Now, we can bound \( \langle \mathbf{d}, \frac{x^* - x_t}{\|x^* - x_t\|} \rangle \leq \|\mathbf{d}\|^2 \) using Cauchy-Schwartz. Therefore, using the linearity of the inner product it holds that:

\[
\epsilon_t - \epsilon_{t+1} \geq \frac{\mu}{L} \left( \frac{\langle \mathbf{d}_{\parallel}, \tilde{z}_t \rangle^2}{\|\tilde{z}_t\|^2} \right) \geq \frac{\mu}{L \cdot \text{radius}(\mathcal{A})^2} \epsilon_t
\]

Recall the inexact linear oracle definition from Equation (5.5), which holds for the returned point \( \tilde{z}_t \) used in the algorithm, that is \( \langle \mathbf{d}, \tilde{z}_t \rangle = \langle \mathbf{d}_{\parallel}, \tilde{z}_t \rangle \leq \delta \langle \mathbf{d}, \mathbf{z}_t \rangle = \delta \langle \mathbf{d}_{\parallel}, \mathbf{z}_t \rangle \) where \( \mathbf{z}_t \) is the true minimizer of the linear subproblem over \( \mathcal{A} \) for given direction \( \mathbf{d} \). The quality parameter \( \delta = 1 \) corresponds to an exact LMO. Applying the definition we obtain:

\[
\epsilon_t - \epsilon_{t+1} \geq \frac{\mu \delta^2}{L \cdot \text{radius}(\mathcal{A})^2} \epsilon_t
\]

Now we apply the crucial definition of minimal intrinsic directional width \( m\text{DW}(\mathcal{A}) \) as given in Definition [12]. From the LMO definition at the negative direction of \( \mathbf{d}' := \frac{d_{\parallel}}{\|d_{\parallel}\|} \in \text{lin}(\mathcal{A}), \) we have \( \langle -\mathbf{d}', \mathbf{z}_t \rangle = \max_{\mathbf{z} \in \mathcal{A}} \langle -\mathbf{d}', \mathbf{z} \rangle. \) Observing that this quantity is directly included in the minimum in the definition of \( m\text{DW}(\mathcal{A}), \) we have \( \langle -\mathbf{d}', \mathbf{z}_t \rangle = \max_{\mathbf{z} \in \mathcal{A}} \langle -\mathbf{d}', \mathbf{z} \rangle \geq m\text{DW}(\mathcal{A}). \) This holds since \( \mathbf{d}' \in \text{lin}(\mathcal{A}) \) as well as \( \mathbf{z}_t \in \mathcal{A} \) as required by the definition. Note also that \( \max_{\mathbf{z} \in \mathcal{A}} \langle -\mathbf{d}', \mathbf{z} \rangle = \text{W}_A(-\mathbf{d}') \) which prove Corollary [14].

Therefore, we have obtained the claimed multiplicative improvement

\[
\epsilon_{t+1} \leq \left( 1 - \frac{\mu \delta^2 \text{mDW}(\mathcal{A})^2}{L \cdot \text{radius}(\mathcal{A})^2} \right) \epsilon_t,
\]

which finishes the proof.

\[\Box\]

**Theorem** [15]. Let \( \mathcal{A} \subset \mathcal{H} \) be a bounded set and let the objective function \( f : \mathcal{H} \to \mathbb{R} \) be both \( L \)-smooth and \( \mu \)-strongly convex. Assume, \( x^* := \arg \min_{x \in \text{lin}(\mathcal{A})} f(x) = \)
A. Appendix

\[ \text{arg min}_{x \in \mathcal{H}} f(x). \] Let \( \epsilon_t := f(x_t) - f(x^*) \) be the suboptimality of the iterates and \( z_t \) the atom selected at iteration \( t \) by the LMO. Then, for \( t \geq 0 \) the suboptimality of the iterates of Matching Pursuit (Algorithm 5) with an exact LMO does not decay faster than:

\[ \epsilon_{t+1} \geq \left( 1 - \frac{W_A(-\nabla f(x_t))^2}{\|z_t\|^2} \frac{L}{2L} \right) \mu \epsilon_t. \]

**Proof.** First of all we note how the distance function is both \( L \)-smooth and \( \mu \)-strongly convex with \( L = \mu \). Recall that the new iterate \( x_{t+1} \) is obtained as \( x_{t+1} = x_t + \lambda z_t \) where \( \lambda = -\langle \nabla f(x_t), z_t \rangle / L\|z_t\|^2 \). By strong convexity we get:

\[ f(x_{t+1}) \geq f(x_t) - \frac{\langle \nabla f(x_t), z_t \rangle^2}{L\|z_t\|^2} + \frac{\mu}{2} \left( \frac{\langle \nabla f(x_t), z_t \rangle}{L\|z_t\|^2} \right)^2 \|z_t\|^2 \]

and subtracting \( f(x^*) \) on both sides yields:

\[ \epsilon_{t+1} \geq \epsilon_t - \frac{\langle \nabla f(x_t), z_t \rangle^2}{L\|z_t\|^2} + \frac{\mu}{2} \left( \frac{\langle \nabla f(x_t), z_t \rangle}{L\|z_t\|^2} \right)^2 \|z_t\|^2 \]

\[ \geq \epsilon_t - \frac{1}{L} \langle \nabla f(x_t), z_t \rangle^2 \left( 1 - \frac{\mu}{2L} \right) \]

By Lipschitzness we obtain:

\[ f(x_t + \gamma(x^* - x_t)) \leq f(x_t) + \gamma \langle \nabla f(x_t), x^* - x_t \rangle + \gamma^2 L \frac{L}{2} \|x^* - x_t\|^2 \]

Now we can further lower bound the LHS by \( f(x^*) \) and minimize the RHS by \( \gamma = -\langle \nabla f(x_t), x^* - x_t \rangle / L\|x^* - x_t\|^2 \). Therefore:

\[ f(x^*) \leq f(x_t) - \frac{1}{L} \langle \nabla f(x_t), x^* - x_t \rangle^2 + \frac{1}{2L} \langle \nabla f(x_t), x^* - x_t \rangle^2 \]

which by definition of \( \epsilon_t \) becomes:

\[ \epsilon_t \geq \frac{1}{2L} \langle \nabla f(x_t), x^* - x_t \rangle^2 \geq \frac{1}{2L} \|\nabla f(x_t)\|^2 \|x^* - x_t\|^2 \]

Recall the first order optimality condition for constrained optimization. We have that \( \langle \nabla f(x_t), x^* - x_t \rangle = 0 \). In the last inequality of Equation A.17 we used the following arguments along with the first order optimality condition:

\[ -\langle \nabla f(x_t), x^* - x_t \rangle \geq \mu \|x^* - x_t\|^2 \]
multiplying and dividing by the norm of the gradient and rearranging we obtain:

\[- \left\langle \nabla f(x_t), \frac{x^* - x_t}{\|\nabla f(x_t)\|} \right\rangle \geq \mu \frac{\|x^* - x_t\|}{\|\nabla f(x_t)\|} \geq \frac{\mu}{L} \]

By taking the square we obtain the inequality used in Equation A.17.

Combining (A.16) and (A.17) we obtain:

\[\epsilon_{t+1} \geq \epsilon_t - 2 \frac{\langle \nabla f(x_t), z_t \rangle^2}{\|\nabla f(x_t)\|^2} \left(1 - \frac{\mu}{2L}\right) \epsilon_t \]

\[= \epsilon_t - \frac{\mu}{L} \frac{W(A)(-\nabla f(x_t))^2}{\|z_t\|^2} \frac{L^2 2L - \mu}{\mu^2} \epsilon_t \]

where in the last equality we used the fact that the inner product between the gradient and \(z_t\) is squared.

\[\square\]

### A.1.4 Proof of Theorem 17

Let \(A \in \mathbb{R}^d\) be a symmetric and bounded set which contains a basis for \(\mathbb{R}^d\). Let also \(B\) be a set such that \(A = B \cup -B\) with \(B \cap -B = \emptyset\) and \(|B| = n > d\).

Our proofs rely on the Gram matrix \(G(J)\) of the atoms in \(B\) indexed by \(J \subseteq [n]\), i.e., \(G(J)_{ij} := \langle s_i, s_j \rangle\), \(i, j \in J\).

To prove Theorem 17, we use the following known results.

**Lemma 43** ([58]). The smallest eigenvalue \(\lambda_{\min}(G(J))\) of \(G(J)\) obeys \(\lambda_{\min}(G(J)) > 1 - \mu(A, m - 1)\), where \(m = |J|\).

**Lemma 44** ([17]). For every index set \(J \subseteq [n]\) and every linear combination \(p\) of the atoms in \(B\) indexed by \(J\), i.e., \(p := \sum_{j \in J} v_j s_j\), we have \(\max_{j \in J} |\langle p, s_j \rangle| \geq \frac{\|p\|^2}{\|v\|_1} = \frac{\|v J(J) v\|_2}{\|v\|_1}\), where \(v \neq 0\) is the vector having the \(v_j\) as entries.

**Theorem 17**. Let \(A \in \mathbb{R}^d\) be a symmetric and bounded set. Let also \(B\) be a set such that \(A = B \cup -B\) with \(B \cap -B = \emptyset\) and \(|B| = n\). Then, the cumulative coherence of the set \(B\) is bounded by: \(\mu(A, n - 1) \geq 1 - n \cdot mDW(A)^2\).
Proof. For each direction $d \in \text{lin}(A)$ with $\|d\| = 1$ it holds that:

$$\max_{z \in A} |\langle d, z \rangle| \geq \frac{\|d\|^2}{\|v\|_1} \geq \frac{\sqrt{\langle v, G(J)v \rangle_2 \|d\|}}{\|v\|_1} \geq \frac{\sqrt{\langle v, G(J)v \rangle_2}}{\sqrt{n} \|v\|_2} \geq \sqrt{1 - \mu(A, n - 1)}$$

This holds for every direction in $\text{lin}(A)$, included the one that minimizes $\max_{z \in A} |\langle d, z \rangle|$. Therefore, we have:

$$\text{mDW}(A)^2 \geq \frac{1 - \mu(A, n - 1)}{n}$$

Rearranging we obtain:

$$\mu(A, n - 1) \geq 1 - n \cdot \text{mDW}(A)^2$$

\[\square\]

A.1.5 On the Relationship Between Matching Pursuit and Frank Wolfe

Theorem Let $A \subset \mathcal{H}$ be a bounded set and let $f : \mathcal{H} \to \mathbb{R}$ be a $L$-smooth convex function. Let $\alpha > 0$ and let us fix an iteration $t > 0$ and the iterate computed at the previous iteration $x_t$. If $- \langle \nabla f(x_t), az_t - x_t \rangle \leq 1$ the new iterate $x_{t+1}^{\text{FW}} = x_t + \lambda(az_t - x_t)$ of Frank-Wolfe (Algorithm 1) using the set $A$ converges to the new iterate $x_{t+1}^{\text{MP}} = x_t + \lambda z_t$ of Matching Pursuit (Algorithm 5) applied on the linear span of the set $A$ with rate:

$$\|x_{t+1}^{\text{FW}} - x_{t+1}^{\text{MP}}\| \in O\left(\frac{1}{\alpha}\right)$$

In particular when $\alpha$ grows to infinity the condition $- \langle \nabla f(x_t), az_t - x_t \rangle \leq 1$ always holds (for all steps $t$). If the condition $- \langle \nabla f(x_t), az_t - x_t \rangle \leq 1$ is not satisfied at step $t$ then the difference of the iterates increases linearly:

$$\|x_{t+1}^{\text{FW}} - x_{t+1}^{\text{MP}}\| \in O\left(\alpha\right)$$

Proof. In both the algorithms the new iterate is obtained by minimizing the quadratic approximation of $f$ computed at $x_t$. Let $g_{x_t}(x)$ be the quadratic approximation of $f$ at $x_t$ given by

$$g_{x_t}(x) = f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{L}{2}\|x - x_t\|^2.$$
At iteration $t > 0$ the new iterate $x_{t+1}^{FW}$ of Frank-Wolfe (Algorithm 1) on the set $\text{conv}(\alpha A)$ is computed as $x_{t+1}^{FW} = x_t + \lambda (\alpha z_t - x_t)$, where:

$$
\lambda = \arg \min_{0 \leq \lambda \leq 1} g_{x_t}(x_t + \lambda (\alpha z_t - x_t))
$$

$$
= f(x_t) + \min_{0 \leq \lambda \leq 1} \left\{ +\lambda \langle \nabla f(x_t), \alpha z_t - x_t \rangle + \frac{L \lambda^2}{2} \| \alpha z_t - x_t \|^2 \right\}
$$

Which solved for $\lambda$ yields:

$$
x_{t+1}^{FW} = \begin{cases} 
\frac{x_t - \langle \nabla f(x_t), \alpha z_t - x_t \rangle}{\| \alpha z_t \|^2} (\alpha z_t - x_t) & \text{if } -\frac{\langle \nabla f(x_t), \alpha z_t - x_t \rangle}{\| \alpha z_t - x_t \|^2} \leq 1 \\
\alpha z_t & \text{otherwise}
\end{cases}
$$

(A.18)

On the other hand, the new iterate $x_{t+1}^{MP}$ of Matching Pursuit (Algorithm 5) on the set $A$ is computed as $x_{t+1}^{MP} = x_t + \lambda z_t$, where:

$$
\lambda = \arg \min_{\lambda \in \mathbb{R}} g_{x_t}(x_t + \lambda z_t)
$$

$$
= f(x_t) + \min_{\lambda \in \mathbb{R}} \left\{ +\lambda \langle \nabla f(x_t), z_t \rangle + \frac{L \lambda^2}{2} \| z_t \|^2 \right\}
$$

which solved for $\lambda$ yields:

$$
x_{t+1}^{MP} = x_t - \frac{\langle \nabla f(x_t), z_t \rangle}{L \| z_t \|^2} z_t
$$

(A.19)

Now:

$$
\| x_{t+1}^{FW} - x_{t+1}^{MP} \| = \begin{cases} 
\| \frac{\langle \nabla f(x_t), z_t \rangle}{L \| z_t \|^2} z_t - \frac{\langle \nabla f(x_t), z_t \rangle}{L \| z_t \|^2} (\alpha z_t - x_t) \| & \text{if } -\frac{\langle \nabla f(x_t), \alpha z_t - x_t \rangle}{\| \alpha z_t - x_t \|^2} \leq 1 \\
\| \alpha z_t + \frac{\langle \nabla f(x_t), z_t \rangle}{L \| z_t \|^2} z_t \| & \text{otherwise}
\end{cases}
$$

Using the fact that $\| \alpha z_t - x_t \|^2 = \langle \alpha z_t - x_t, \alpha z_t - x_t \rangle$ it is easy to show that the distance depends on $1/\alpha$ when $-\frac{\langle \nabla f(x_t), \alpha z_t - x_t \rangle}{\| \alpha z_t - x_t \|^2} \leq 1$ while is linear in $\alpha$ in the other case. Furthermore, when $\alpha$ grows $-\frac{\langle \nabla f(x_t), \alpha z_t - x_t \rangle}{\| \alpha z_t - x_t \|^2} \leq 1$ always holds. Therefore:

$$
\lim_{\alpha \to \infty} \| x_{t+1}^{FW} - x_{t+1}^{MP} \| = 0
$$

(A.20)

Therefore, the Frank-Wolfe step converges to the Matching-Pursuit step when $\alpha$ (i.e. the diameter of the set) grows to infinity. \Box
Bibliography


Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor’s thesis, Master’s thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

GREEDY OPTIMIZATION AND APPLICATIONS TO STRUCTURED TENSOR FACTORIZATIONS

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):

FRANCESCO

First name(s):

LOCATELLO

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the ‘Citation etiquette’ information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

ZURICH

Signature(s)

Francesco Locatello

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.