ORTHOGONAL MATCHING PURSUIT ALGORITHM FOR SPARSE SIGNAL RECOVERY

J. FOLBERTH AND E. YASUTAKE

Contents

1. Introduction

1.1. **Purpose of the Project.** The main goal of this paper is to discuss and replicate the results from "Signal Recovery from Random Measurements via Orthogonal Matching Pursuit" (Tropp,Gilbert, 2007).

1.2. **Introduction.** Suppose that have an unknown vector $\boldsymbol{s} \in \mathbb{R}^d$ that has m nonzero elements where, in most applications, $m << d$. Our goal is to recover $\boldsymbol{s}$ with a sequence of *measurement vectors* $\{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^d$ such that we know the values

$$\langle \boldsymbol{s}, \boldsymbol{x}_i \rangle \quad i = 1, \dots N.$$

There are two main questions involved with this type of problem:

(1) What value of N would one need to recover $\boldsymbol{s}$?
(2) What algorithms can we use to recover $\boldsymbol{s}$?

To answer the first of these questions, it's clear that we would want at least m measurement vectors, or else we would have an underdetermined system. On the other hand, simply taking any basis, such as the canonical basis, in $\mathbb{R}^d$ is sufficient. However, it turns out that we can do much better than this.

The latter question has many answers. In this paper, our main focus is on an algorithm called Orthogonal Matching Pursuit (OMP). There are a number of algorithms that have been developed for this problem. We will compare OMP with a similar algorithm known as Basis Pursuit (BP).

Date: April 25, 2014.

2. Orthogonal Matching Pursuit

Once we fix the number, N, of measurement vectors we want, the idea is to choose a subset of these measurement vectors to recover the signal $\boldsymbol{s}$. This can be done by first taking the measurement vectors $\{\boldsymbol{x}_i\}_{i=1}^N$ and forming an $N \times d$ matrix Φ. The rows of Φ are the measurement vectors. We will henceforth denote the columns by $\{\varphi_i\}_{i=1}^d$. We can then collect the measurements $\langle \boldsymbol{s}, \boldsymbol{x}_i \rangle$ into a vector $\boldsymbol{v}$, which leads to the system of equations$\boldsymbol{v} = \Phi \boldsymbol{s}$. The goal of OMP is to determine which columns of Φ are necessary for the matrix equation. OMP does this in a greedy fashion by choosing the columns of Φ which are most stonrgly correlated to the remaining parts of $\boldsymbol{v}$. The below algorithm explains the full OMP method.

2.1. **Algorithm.**

(1) Fix an $N \times d$ matrix Φ, $\boldsymbol{v} \in \mathbb{R}^d$, and sparsity level m.
(2) Initialize the residual $\boldsymbol{r}_0 = \boldsymbol{v}$, index set $\Lambda_0 = \emptyset$, and counter $t = 1$.
(3) Find
$$\lambda_t = argmax_{j=1,\dots,d} \left| \langle \boldsymbol{r}_{t-1}, \phi_j \rangle \right|$$
(4) Set $\Lambda_t = \Lambda_{t-1} \cup \{\lambda_t\}$ and $\Phi_t = [\Phi_{t-1} \phi_{\lambda_t}]$ where Φ_0 is an empty matrix.
(5) Solve the least squares problem
$$x_t = argmin_{\boldsymbol{x}} \left|\left| \boldsymbol{v} - \Phi_t \boldsymbol{x} \right|\right|_2 .$$
(6) Calculate the t-th approximation to $\boldsymbol{v}$ and residual via
$$\boldsymbol{a}_t = \Phi_t \boldsymbol{x}_t$$
$$\boldsymbol{r}_t = \boldsymbol{v} - \boldsymbol{a}_t.$$
(7) So long as $t < m$, return to Step 2.

2.2. **Operations Count.**

There are four main steps involved in the algorithm from above. The first is Step 3, which relies on d inner products of computational cost mN. Step 3 calls for a QR decompositions via adding columns. In practice, this should only cost **FILL THIS IN**. We will later discuss why our implementation of OMP costs more (see Further Extensions). Step 5 solves the least squares problem, costing **FILL THIS IN**. Finally, we must compute the t-th approximation in Step 6, which costs $2mn$. This leads to a total cost of

$$+2mn \approx mNd$$

3. Applications

4. Further Extensions

4.1. **Implement QR via insertion.** MATLAB SUCKS, MUWAHAHAHAHA! I hate MATLAB.