Optimization of measurement matrices for CS
Project Report

Advanced Image Processing (CS754, Spring 2022)
Guide: Prof. Ajit Rajwade

Dhvanit Beniwal (200050035)
Dhananjay Kejriwal (200050034)

Overview

The goal is to construct sensing matrices $D = \Phi\Psi$ that are better suited for compressive sensing tasks, where the metric to optimise is mutual coherence (μ). The reason for this is because whenever the inequality

$$||\alpha||_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(D)}\right)$$

holds, α is "necessarily the sparsest reconstructed signal" for $y = D\alpha$ (quoting from the paper). This means that compressive measurements with a D that has small mutual coherence are reconstructible successfully for a wide range (in terms of sparsity) of signals α

Optimization Algorithm

The mutual coherence is given by

$$\mu(D) = \max_{i \neq j}\left\{\frac{|d_i^\top d_j|}{||d_i|| \cdot ||d_j||}\right\}$$

Where d_i is the i-th column of D. The term inside the parenthesis is the (i, j)-th term of the gram matrix of D, written as $G = \tilde{D}^\top \tilde{D}$ where $\tilde{D}$ is the column normalised version of D, and so we talk about the maximum (magnitude) of the off-diagonal ($i \neq j$) terms of the gram matrix. The diagonal elements of a gram matrix are always 1 anyways. The objective function to minimize in order to get a D with small μ is

$$E = ||G - I||_F^2$$

The algorithm used to minimize E is gradient descent, where we consider the gradient of E with respect to $\tilde{D}$,

$$\frac{\partial E}{\partial \tilde{d_{ij}}} = 4\tilde{D}(\tilde{D}^\top \tilde{D} - I)$$

For the gradient descent step we use a fixed stepsize η. Note that we have a constraint that the columns of $\tilde{D}$ are unit norm which may not hold after a gradient descent step. This is why we normalise the columns of the new $\tilde{D}$ after each gradient descent step.

Since in most applications we only want to optimise the sensing matrix Φ since we already have a choice for the representing matrix Ψ, we may finish by calculating Φ as $D\Psi^{-1}$. This is assuming Ψ is square. For overcomplete dictionaries Ψ the paper suggests gradient descent directly on Φ with the gradient step as:

$$\Phi \leftarrow \Phi - \eta(\Phi\Psi)((\Phi\Psi)^\top \Phi\Psi - I)\Psi^\top$$

However it was not clear how we have to impose unit norm constraints on $\Phi\Psi$ and our attempts based on guesswork did not produce valid results.

Instead, we argue that for overcomplete dictionaries, we run the same algorithm as before but at the end write $\Phi = D\Psi^\dagger$ (pseudo-inverse). This is motivated from the fact that if Ψ has a right inverse (which it can) it will be equal to its pseudo inverse. We go forward with this approach as it is easy enough to implement and it happens to give valid results

Elad's method

The paper does not talk about this method of optimization but compares results from Elad's method of optimization to results from its proposed algorithm. We have therefore implemented Elad's method as per the cited paper. The algorithm is as follows:

1. Initialise Φ to a random matrix
2. Repeat steps 3-7 an $Iter$ number of times
3. Normalise columns of $\Phi\Psi$ to get $\tilde{D}$
4. If given relative threshold, calculate corresponding fixed threshold as the value at t%-th percentile.
5. "Shrink" the gram matrix to get $\tilde{G}$, here $\tilde{g_{ij}}$ is made γg_{ij} or $\gamma t \text{sign}(g_{ij})$ or g_{ij} according to where $|g_{ij}|$ lies compared to t and γt
6. Find a rank p approximation to $\tilde{G}$ in the form of $S^\top S$ where S is $p \times k$. This is done by taking the SVD decomposition and choosing the p largest singular values
7. Set Φ to $S\Psi^\dagger$ (pseudo-inverse) since that is the value that minimizes $||S - \Phi\Psi||_F^2$
8. return Φ

Validation of proposed method

To demonstrate the validity of the proposed optimization algorithm the paper uses a series of generated test cases which are as follows:

- Generate Φ (100×200) and Ψ (200×400) with i.i.d. gaussian elements. Run the proposed algorithm with stepsize $\eta = 0.02$ and 60 iterations. Run Elad's method with $\gamma = 0.5$, $t = 0.25$, 60 iterations. Plot the distribution of off-diagonal gram elements for all of them.
- Repeat after replacing Ψ with the 200×200 DCT matrix

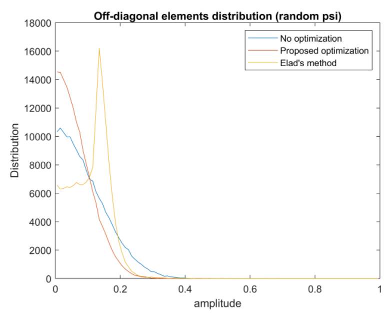

Figure 1: Off-diagonal elements distribution (gaussian)

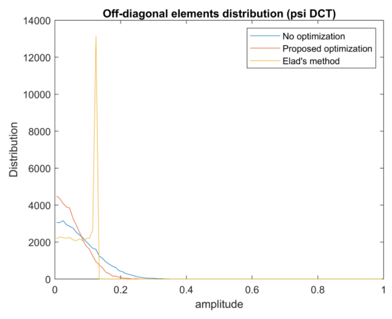

Figure 2: Off-diagonal elements distribution (DCT)

This shows that Elad's method would bring the off-diagonal gram elements close to a small value (like 0.1) but increasing the number of iterations only lifts the peak, doesn't shift it (the peak depends on γ and t). The advantage of the proposed method is that it brings all values closer to zero rather than a small positive value

To study the effects of reconstruction error on signals measured by these matrices, we generate 1000 sparse signals of length 120, chose Ψ as the DCT basis, run proposed algorithm with $\eta = 0.01$ and 150 iterations, run Elad's method with $\gamma = 0.95$, $t = 20\%$, 1000 iterations. The variation in error rate for the 1000 signals against variation in sparsity (s) and variation in number of measurements (p) is found and the conclusion drawn is that:

The proposed method works just as good as Elad's method, if not better. And there is a noticeable decrease in error-rate from non-optimized matrices

The reconstruction algorithms used for these purposes are OMP (orthogonal matching pursuit) and BP (Basis Pursuit). A note on implementing BP in matlab, we convert the optimization problem into a linear programming problem as described in the lectures and use matlab's implementation from linprog()

Extension of the proposed algorithm

The proposed algorithm is not just a gradient descent algortihm as we have to normalise the columns after every step. This is because we are doing gradient descent w.r.t the variable $\tilde{D}$ when the 'free' variable is D.

It was observed that if instead of optimizing a i.i.d. gaussian matrix, if we start with a matrix whose elements are a uniform distribution between 0 and 1, we find that the proposed algorithm does the opposite of what we want. It makes the off-diagonal elements close to 1.

A possible fix is to calculate the gradient of $E = ||G - I||_F^2$ w.r.t D (and not $\tilde{D}$) and do gradient descent directly on D with no normalization steps in between. We will find that this performs almost identical to the proposed algorithm in case of iid gaussian elements, but it also optimises the matrix with uniformly distributed elements (which the proposed algorithm can't)

The gradient can be expressed as follows:

$$\frac{\partial E}{\partial d_{ij}} = 4\tilde{D}(\tilde{D}^\top \tilde{D} - \text{diag}(\tilde{D}^\top \tilde{D}\tilde{D}^\top \tilde{D})) \cdot N \tag{1}$$

Where $\tilde{D}$ is the column normalised version of D. diag() here means to only keep the diagonal elements and make the rest equal to zero. The matlab function diag() is to be called twice because it reshapes it into a vector if called once. N is a diagonal matrix where $N_{ii} = 1/(||d_i||)$, d_i is the i-th column of D.

To derive this, we need to write the objective function as:

$$\begin{aligned} E &= \sum_{i \neq j} \left(\frac{|d_i^\top d_j|}{||d_i|| \cdot ||d_j||} \right)^2 \\ &= \sum_{i \neq j} \frac{(d_i^\top d_j)^2}{||d_i||^2 \cdot ||d_j||^2} \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial d_i} &= 2 \sum_{j, j \neq i} \frac{\partial}{\partial d_i} \left(\frac{(d_i^\top d_j)^2}{||d_i||^2} \right) \cdot \frac{1}{||d_j||^2} \\ &= 2 \sum_{j, j \neq i} \left(\frac{2(d_i^\top d_j) d_j}{||d_i||^2} - \frac{2(d_i^\top d_j)^2 d_i}{||d_i||^4} \right) \cdot \frac{1}{||d_j||^2} \\ &= 4 \sum_{j, j \neq i} \frac{(d_i^\top d_j)}{||d_i||^2 ||d_j||^2} \left(d_j - d_i \frac{(d_i^\top d_j)}{||d_i||^2} \right) \\ &= 4 \sum_{j} \frac{(d_i^\top d_j)}{||d_i||^2 ||d_j||^2} \left(d_j - d_i \frac{(d_i^\top d_j)}{||d_i||^2} \right) \end{aligned}$$

This is the 'derivative' w.r.t a column of D. We have to concatenate these columns to get the final gradient. We know that $d_i/||d_i||$ is the i-th column of $\tilde{D}$. Using this we can transform this expression into a form like (1)