CS754 Project Proposal

Arpon Basu- 200050013
Shashwat Garg- 200050130

Spring 2022

Welcome to our report on CS754 Project Proposal. We plan to implement, experiment and extend the work done in the following paper, Enhancing Sparsity by Reweighted 1 Minimization.

The paper basically introduces an iterative procedure where the **weighted l_1-norm minimization** problem is solved, where the weights are updated in every iteration, and the process is terminated when convergence is achieved and/or our iteration budget is exhausted.

Many heuristic arguments, as well as a few concrete arguments have been given in favour of this method, especially why it performs better than the vanilla **Basis Pursuit Method** (which minimizes the l_1 norm).

1 Our proposals regarding the implementation and extension of the algorithms given in the paper

Coming to our contributions, we seek to extend their work in the following ways:

- We want to see how the algorithm behaves if we replace the weighted l_1 norm by fractional norms such $l_{1/2}$ and $l_{2/3}$. We want to explore whether this will lead to:
  - Faster convergence, ie:- do using values closer to zero lead to faster convergence?
  - We want to explore if changing the index of the norm will significantly increase the amount of computational resources required (especially since we're dealing with fractional norms here).
  - Note that if it so happens that "better" results are obtained for some non-zero, non-one value of the index of norm, we shall know that beneath the linear exterior of the algorithm, some interesting non-linear behavior is taking place (due to the change of weights in every iteration, for example).
- Another area in which we can tweak the algorithm is the update step: Note that in this algorithm, the weights at every iteration are updated according to a certain rule, ie:- at the end of every iteration,

$$w_i^{(l+1)} = g(x_i^{(l)}, \epsilon)$$

where $\boldsymbol{x}^{(l)}$ is the vector which minimized the norm at the l^{th} iteration. Now, the function $g(x_i, \epsilon)$ can be shown to be the derivative (w.r.t the first variable x_i) of our **cost function** $f(x_i, \epsilon)$, which in case of the algorithm presented in the paper is $f(x_i, \epsilon) := \log(|x_i| + \epsilon)$. Thus another natural avenue for extension is the exploration of other cost functions (this is suggested in the paper itself) typically used in machine learning, for example the arctan function ($f(x_i, \epsilon) :=$atan(x_i/ϵ). This was suggested by the authors themselves), ReLU function ($f(x_i, \epsilon) := \max(0, x_i/\epsilon)$) and the tan-hyperbolic function ($f(x_i, \epsilon) := \tanh(x_i/\epsilon)$), where ϵ is a hyperparameter to be chosen carefully.

- We want to explore how slightly tweaked versions of the algorithm behave vis-a-vis convergence, ie:-
  - To achieve the same amount of accuracy $1-\varepsilon$, how many iterations are needed? How much better can we do than what is required by the algorithm given in the paper?
  - We can also answer the above question by answering it's "dual" question, ie:- given a fixed iterative budget, which version of the algorithm achieves the maximum accuracy at the end of the iteration?
  - We also want to explore how stable our algorithm's convergence is, ie:- does the accuracy of our estimates increase consistently as the number of iterations increase, or do they oscillate violently while they converge?
  - What are the computational cost trade-offs as we change our norm-metrics and cost functions?
- Finally, we would try to establish some theoretical results which would explain the results of our numerical experiments above. This could be done by taking inspiration from some results already established by the authors in their paper.

2 Dataset and evaluation/validation strategy

We plan to apply our algorithms on a small sampling of good-quality, real life images obtained from some standard image database such as ImageNet.
Once we have an image, we plan to take a "compressed snapshot" of that image, ie:- let $\boldsymbol{x}$ be the vectorized form of our image. Then, we shall apply the compressed sensing matrix A on $\boldsymbol{x}$ to obtain our compressed vector $\boldsymbol{y} := A\boldsymbol{x}$.
The matrix A itself is obtained via multiplying our sensing matrix Φ and basis matrix Ψ. Thus on the basis of standard compressed sensing theory, we can take Ψ to be the Fourier basis, DCT basis or Wavelet basis, and we can choose Φ accordingly so that $A = \Phi\Psi$ has low coherence and other desirable properties typically expected of a compressed sensing matrix.
Thus, after we have obtained $\boldsymbol{y}$, we'll apply our algorithms on $\boldsymbol{y}$ to obtain the reconstructed image $\tilde{\boldsymbol{x}}$, whose quality we can judge by checking out the MSE errors, ie:- $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2 / \|\boldsymbol{x}\|_2$.