Blendenpik Problem Set

Malachi Demmin, Dong Hu, Aaron Micah Green
Sharmishtha Dutta, Bingsheng Yao

December 2, 2020

1 Problem set

For all of these problems you will use the Blendenpik algorithm to solve various Least Squares problems. You will create an algorithm that matches the pseudo code found in the paper to implement Blendenpik. That is, you will include all parameters and options for various Unitary transformations.

1. This problem concerns the creation of your algorithm.

   (a) Create a Blendenpik object that can be created with the following inputs.

   ```
   A, b, gamma, transform type, solver, tolerance,
   and maximum iterations
   ```

   (b) Your transform parameter should have 3 possibilities. It should choose between the 3 transformations in the paper, DCT, DHT, and WHT. Note that this affects the padding step so that will also have to rely on this option.

   (c) Your should implement both solvers LSQR and LSMR, along with the fall back option LAPACK for solving least squares problems.

   (d) Maximum iterations refers to the number of times we allows the preconditioning step to run if it fails to lower the condition number satisfactorily before the solving step, this should default to 3.

   (e) You will also need to implement 2 functions, first a Diagonal matrix creator that accepts a dimension number and outputs a dimension $\times$ dimension matrix with entries either 1 or -1 with equal probability. Second a function that creates the sampling matrix needed before factoring the matrix.

2. Now that you have created the Blendenpik solver you will attempt to recreate some of the papers experiments to confirm performance and create a baseline for comparison.

   (a) Run an experimental loop that tests convergence for gamma values between 1.5 and 10. Create graphs that represent the convergence rates for each gamma. This is to confirm that $\gamma = 4$ is near optimal for all methods.

   (b) Create three scripts to generate matrices. The first script creates a incoherent matrix simply by creating a random matrix. The second creates Semi-coherent matrices where the upper left submatrix is a random matrix, and the bottom right matrix is the identity matrix. The dimensions are

$$S = \begin{bmatrix} R & \\ & I \end{bmatrix} + 10^{-1}L$$

   where R is $(m - n/2) \times n/2$, I is thus $n/2$ and L is a matrix of ones. The third script creates a coherent matrix of the form

$$Z_{m\times n} = \begin{bmatrix} D_{n\times n} \\ 0_{m-n\times n} \end{bmatrix} + 10^{-8}L$$

   . Where D is a random diagonal matrix.

   (c) Use these three forms of matrices to test the performance speed of this algorithm. Create a matrix from each script of size $m \times m/40$ and test it up to $m = 80,000$.

   (d) Perform a similar convergence experiment on LSQR vs (OTHER METHOD) where you take one of each type of matrix of size $100,000 \times 2500$ and calculate

$$\frac{\|A^T r^{(i)}\|_2}{\|A\|_F \|r^{(i)}\|_2}$$

   versus the iteration of the algorithm.

3. Now that you have compared your implementation to the papers, use your code to run some real world experiments.

   (a) Download the data set CIFAR-10 and CIFAR-100, and apply Blendenpik to predict the class of the images therein.

   (b) Does the 10 class set perform better than the 100 class set?

   (c) If so why do you think so? If not, why is the 100 class set better with this algorithm?