gU 6001 Final Project: Reduced Basis Decomposition

A recurrent theme and interest in data science is the process of dimensionality reduction. Linear Algebra methods such as SVD, PCA, and RBD are the mathematical crux of dimensionality reduction with applications in compression, predictive modeling, feature discovery.

In this course, we have seen how SVD and PCA can create adequate approximations of data with a minimal number basis vectors. SVD computes all of the eigenvalues and eigenvectors of matrix in order of primary importance. In the case of image compression, a basis vector count of 40% of the original image can constitute a compression that is indistinguishable to the human eye.

If so few basis vectors are needed to produce an adequate image compression, why compute a full set of eigenvalues and vectors with SVD? RBD can substitute SVD and outperform it in cases where we want to limit compression size.

RBD is a greedy lossy algorithm that uses the modified Gramm-Schmidt process to build a lean set of optimized basis vectors. Its virtues are its speed and efficiency in the early stages of a decomposition.

Pseudocode:

Y,T = RBD(X, Error_Requirement, Max_Basis)

While the Error_Requirement and Max_Basis conditions are unmet:

1. **Pick a vector from X with the optimal error correction power to orthogonally project onto the Y space.**
2. **Add the vector to Y using the Gramm-Schmidt process. Add the dot product of the transpose of the vector and X to T, enabling future reconstruction.**
3. **Check to make sure the newest column vector in Y hasn't solved the decomposition problem already.**

Computation Cost:

Given a matrix ***X*** with ***m x n*** columns and ***d*** number of basis in the decomposition:

- Orthogonalize each incoming basis against the current basis set = **m*d^2**
- Compare each column of ***X*** with compressed version into the current basis space.
    - Diagonalize X.T.dot(I).dot(X) = **sqrt(m^2 * n^2) * n**
    - Expansion of Y.T.dot(I).dot(X) = **sqrt(m^2 * n^2)*d * m**
    - Expansion of Y.T.dot(I).dot(Y) = **sqrt(m^2 * n^2)* d^2**
- Search for optimal vectors = **n * d^3**

Total RBD cost = **O(m + sqrt(m^2 * n^2))*d^2 + n*(d^3 + sqrt(m^2 * n^2))))**

Total SVD cost = O(m^2 n + n^3)

Performance as d increases: