Principle of Data Science Project Proposal

Bhumrapee Soonjun

May 18, 2023

1 Topic and problem statement

1.1 What is your project title?

Comparative Analysis of Dimensionality Reduction Algorithms

1.2 What is the major problem/research question that you seek to answer?

Let's note that the research question I am proposing here will be more on the theoretical side of the field, rather than being applied. I want to compare the performance of the dimensionality reduction algorithm derived from the Johnson-Lindenstrauss lemma, let's call Johnson-Lindenstrauss (J-L) transforms - to the other - more traditional - dimensionality reduction algorithms such as but not limited to PCA, SVD, and LDA. The performance in question is characterized namely by the relative error after applying the transformation, the running time, the number of resulting dimensions, and, if time permits, measuring the accuracy and speed of the transformed data using ML algorithms. We are also interested in seeing how the algorithms performed given different kinds of datasets, for example, a sparse dataset and non-sparse dataset, and compact and non-compact dataset.

1.3 What is the rationale for the problems?

PCA and SVD are known to be expensive operations on large datasets since the former requires the eigenvalues and eigenvectors to be calculated on the covariance matrix, and the latter requires matrix factorization. The Johnson-Lindenstrauss transforms on the other hand requires only matrix multiplication which is much less expensive than the formerly discussed methods; however, since the nature of the J-L transforms is stochastic, there could be some drawbacks in terms of error as well. Moreover, the constant hiding in the Big-O is unknown. Also, the asymptotic bounds look promising, the if the constant c is big, there could be a lot of overhead. Hence, this research question will help us decide which dimensionality reduction algorithm to use given different kinds of datasets in order to reduce the data processing time.

2 Introduction and relevant literature

In this term project, I will be exploring the performance difference between each dimensionality reduction algorithm. Namely, we will compare the performance of PCA, SVD, and the variants of J-L transforms.

2.1 Johnson-Lindenstrauss Lemma

For $0 < \epsilon < \frac{1}{2}$, given any set of points $X = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^D$, there exists a linear mapping $A : \mathbb{R}^D \to \mathbb{R}^k$ with $k = O(\frac{1}{\epsilon^2} \log n)$ s.t. $\forall i, j \in \{1, ..., n\}$ and $i \neq j$

$$(1 - \epsilon)\|A(x_i) - A(x_j)\|_2^2 \leq \|x_i - x_j\|_2^2 \leq (1 + \epsilon)\|A(x_i) - A(x_j)\|_2^2$$

2.2 Distributional Johnson-Lindenstrauss Lemma

For $0 < \epsilon, \delta < \frac{1}{2}, k = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$, $x \in \mathbb{R}^D$ is a unit vector, then there exists a distribution over $R^{k \times D}$ from which the matrix A is drawn such that

$$\mathbf{Pr}[|\|A(x)\|_2^2 - \|x\|_2^2| \leq \epsilon \|x\|_2^2] \geq 1 - \delta$$

This lemma can then be used to prove the first lemma (the Johnson-Lindenstrauss lemma). Then it follows from the proof of this lemma that the construction of the linear map A can be random values sampled from the standard Gaussian curve. Moreover, if $\delta \sim \frac{4}{n^2} \rightarrow k = O(\frac{1}{\epsilon^2} \log n)$ and it follows from the lemma that there exists such a linear mapping w.p. $1 - \frac{4}{n^2} \geq \frac{1}{2}$; hence, the expected number of trials is 2.

3 Describe your research design

3.1 Research Outline

We will implement each algorithm from scratch. Each implementation will be "textbook" based so that there is no further optimization made that will affect the results of the research. However, we will also test it against a standard package such as SciPy's built-in PCA and SVD, but the main results of this project will follow from the textbook implementation of the algorithms. The variants of the J-L projection we will be implementing are the Gaussian projection, Sparse random projection, the Fast Johnson-Lindenstrauss transform, and if time permits the extremely sparse Johnson-Lindenstrauss transform.

For each of the datasets, we will feed them one by one to each algorithm and record the performance mentioned earlier. After having the results for every algorithm, the results will then be compared at the last step to see how the Johnson-Lindenstrauss performs in comparison to the other algorithms given different kinds of datasets.

3.2 Where is the source of data collection?

The data can be generated by ourselves since we just need to test the efficiency of each algorithm on different kinds of data. Moreover, by generating the data ourselves, we can easily control the structure and its nature as well.

3.3 What is the data about?

Any real-valued sets of vectors in the Euclidean space each spanning the same dimension will do.

3.4 Define values in the data

1. Independent Variables include data (matrix) input sizes, the sparseness of the data, compactness of the data, and numerical stability of the data (largeness of the coefficient of the components of each vector).

2. Dependent Variables include a relative of the transformed data, running time, and memory usage.

3. Controlled Variables include the same machine, background tasks, and the relatively same level of idle CPU usage.

3.5 What will the method of analysis be used?

We can directly compare the results.

3.6 Which model will be used?

No model will be used. This is a comparative analysis.

4 Expected outcome

After the conclusion, we will know how the Johnson-Lindenstrauss dimensionality reduction compared to the more traditional methods and whether should it be used and if it should, in what situation.

5 Citation

References

[1] R. Yin, Y. Liu, W. Wang and D. Meng, "Extremely Sparse Johnson-Lindenstrauss Transform: From Theory to Algorithm," 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 2020, pp. 1376-1381, doi: 10.1109/ICDM50108.2020.00180.

[2] Ailon, N., amp; Chazelle, B. (2009). The fast johnson–lindenstrauss transform and approximate nearest neighbors. SIAM Journal on Computing, 39(1), 302–322. https://doi.org/10.1137/060673096