Proceedings of Machine Learning Research 129:753–768, 2020 ACML 2020

Scalable Calibration of Affinity Matrices from Incomplete Observations

Wenye Li WYLI@CUHK.EDU.CN
The Chinese University of Hong Kong, Shenzhen, Guangdong, China
Shenzhen Research Institute of Big Data, Guangdong, China

Editors: Sinno Jialin Pan and Masashi Sugiyama

Abstract

Estimating pairwise affinity matrices for given data samples is a basic problem in data processing applications. Accurately determining the affinity becomes impossible when the samples are not fully observed and approximate estimations have to be sought. In this paper, we investigated calibration approaches to improve the quality of an approximate affinity matrix. By projecting the matrix onto a closed and convex subset of matrices that meets specific constraints, the calibrated result is guaranteed to get *nearer* to the unknown true affinity matrix than the un-calibrated matrix, except in rare cases they are identical. To realize the calibration, we developed two simple, efficient, and yet effective algorithms that scale well. One algorithm applies cyclic updates and the other algorithm applies parallel updates. In a series of evaluations, the empirical results justified the theoretical benefits of the proposed algorithms, and demonstrated their high potential in practical applications.

Keywords: Missing Data; Matrix Calibration; Affinity Matrix

1. Introduction

Developing techniques to processing incompletely observed data is a fundamental task in statistics (Little and Rubin, 2019). Missing data pose nontrivial challenges to data analysis in numerous science and engineering domains. One application of particular interest in this paper is on estimating the affinity (or similarity) between pairs of data samples, which is of key importance and lays a foundation in many machine learning models, as commonly seen in supervised learning and unsupervised learning algorithms (Jain et al., 1999; Duda and Hart, 2000).

A lot of work has been devoted to model the affinity between data samples. When data samples are represented as algebraic vectors, the inner product model, the cosine similarity, the radial basis function, etc., are popularly adopted in practice (Salton et al., 1975; Scholköpf and Smola, 2001). When each vector element takes binary values only, the Jaccard index is often used (Jaccard, 1912; Rogers and Tanimoto, 1960).

These models work empirically well when the data are fully observed. However, if missing values are taken into consideration, which often occur when certain features aren't observed or don't exist in data samples, these models typically become not directly applicable and nontrivial challenges arise to the learning algorithms that reside on the estimation of pairwise affinities between data samples.

To handle the difficulty, significant attention has been devoted on the study of missing data from the early years of statistics. As a routine treatment, various imputation techniques were designed with great influences on various disciplines. These techniques try to make the data *full* by replacing the un-observed feature values with substituted values based on various assumptions, such as distribution assumptions or low-rank assumptions (Little and Rubin, 2019; Enders, 2010; Wright et al., 2009).

The pairwise affinities can be calculated based on the imputed *full* samples. Unfortunately, there still exist significant difficulties even if the assumptions hold. With a large portion of missing observations, the required computation becomes expensive or even infeasible. There is no tight guarantee on the quality of the imputation, needless to say the quality of the subsequent pairwise affinities that rely on the imputed data.

Instead of imputing missing data, this paper follows a different line which calibrates an approximate affinity matrix (Li, 2015). Comparing with the classical imputation techniques, the proposed approach makes little assumption about the missing values and has wide application scenarios. Moreover, the proposed approach has another strong advantage in that the calibrated matrix is guaranteed to be better than, or at least identical to, in special cases, the un-calibrated matrix in terms of a shorter Frobenius distance to the unknown true affinity matrix.

A major challenge to the calibration approach is the expensive computation required. As a remedy, we designed two simple algorithms for scalable calibration. By dividing a large affinity matrix into smaller ones and solving them one by one, it is guaranteed to build an affinity matrix that is better than the initial approximation. The proposed algorithms are empirically verified to be effective and efficient.

The paper is organized as follows. Section 2 reviews the related background. Section 3 introduces the proposed model and algorithms in detail. Section 4 reports the evaluation results, followed by conclusion in Section 5.

2. Background

2.1. Missing Data and Imputation Approaches

When data samples are fully observed, the affinity matrix can be obtained trivially by calculating their inner product, cosine similarity, Jaccard index, and so on, between each pair of samples. Unfortunately, missing observations are common in reality (Little and Rubin, 2019; Enders, 2010), which poses nontrivial challenges to affinity estimation. Consequently, many data analysis algorithms that rely on pairwise similarities cannot be directly applied any more. To handle the difficulty, one naïve approach is to ignore the features with missing values. Only those fully-observed features are kept in data analysis. Unfortunately, if the feature values are randomly missing in a large dataset, this method is likely to throw away all the features, which makes the approach not applicable at all.

In practice, imputation methods are widely applied. A missing value is simply replaced by a zero value, or by the feature's mean, median or most frequent value in the nearest neighboring samples or all observed samples. A more rigorous treatment that attracts much attention is based on the classical expectation maximization (*EM*) algorithm (Dempster et al., 1977). The algorithm assumes the existence of un-observed latent variables. It

operates by alternating between the expectation and the maximization steps, and seeks maximum likelihood or maximum a posterior estimates of the latent variables.

2.2. Matrix Calibration

Instead of throwing away the un-observed features or imputing the missing values, a completely different approach was designed in the work of (Li, 2015) to calibrate the pairwise affinities from missing data. The method starts from an approximate affinity matrix and calibrates it to meet model specific constraints. Commonly, such constraints include the *positive semi-definiteness* requirement on the affinity matrix, and the *boundedness* requirement for each matrix entry. A key result of the work is that, although the true affinity matrix from the incompletely observed samples is unknown, the method guarantees to calibrate the approximate matrix to be *nearer* to the ground-truth in terms of a shorter Frobenius distance, except in rare cases that the starting and the calibrated matrices are identical.

Despite the successful theoretical and empirical results, the method faces a nontrivial challenge in scalability to large-scale problems. To provide the *positive semi-definiteness* required by the affinity models, the method needs to decompose the affinity matrix repeatedly, which poses significant challenge in computation. When the sample size is large, say tens of thousands, the computation becomes even prohibitive on conventional computing platforms.

3. Scalable Matrix Calibration

3.1. Basic Model

For a given set of data samples $\{A_1, \cdots, A_n\}$, our work investigates the affinity matrix estimation problem. Denote by $J^* = \left\{J^*_{ij}\right\}$ an $n \times n$ affinity matrix, where J^*_{ij} is the true affinity value between two samples A_i and A_j $(1 \leq i, j \leq n)$. Due to the existence of missing observations, this ground-truth J^* is unknown. We are interested in *how to calibrate an approximate affinity matrix to be nearer to the unknown* J^*.

For simplicity, we specifically made two assumptions about J^*, although our proposed work can be applied in more general settings.

(1) *Positive semi-definiteness*: the matrix J^* is positive semi-definite, i.e., $J^* \succeq 0$.

(2) *Boundedness*: A lower bound ℓ_{ij} and an upper bound μ_{ij} are known for each element of J^*, i.e., $\ell_{ij} \leq J^*_{ij} \leq \mu_{ij}$ $(1 \leq i, j \leq n)$.

These two assumptions are mild. In many popular affinity models, such as the inner product model, the cosine similarity model and Jaccard index model, the affinity matrix calculated from full observed data is always positive semi-definite. But with missing observations, the approximate affinity matrix usually loses such *positive semi-definiteness*.

The second assumption also commonly holds in practice. For example, all cosine similarity values are within $[-1, 1]$. All Jaccard index values are within $[0, 1]$. Besides, even tighter bounds can be obtained for specific sample pairs based on their observed feature values (Li, 2015).

To formulate the problem, let M_n be the set of $n \times n$ symmetric matrices, equipped with an inner product that induces the Frobenius norm:

$$\langle X, Y \rangle = trace\left(X^T Y\right), \text{for } X, Y \in M_n.$$

Define two nonempty closed and convex subsets of M_n by: $S = \{X \mid X \in M_n, X \succeq 0\}$, and $T = \{X \mid X \in M_n, \ell_{ij} \leq X_{ij} \leq \mu_{ij}\,(1 \leq i, j \leq n)\}$ for given $\{\ell_{ij}\}_{ij=1}^n$ and $\{\mu_{ij}\}_{ij=1}^n$. Let $R = S \cap T$. Obviously, the true J^* that meets the two assumptions satisfies $J^* \in R$.

Our approach proceeds from an affinity matrix $J^0 \in M_n$ approximated from the observed features. Very likely J^0 loses *positive semi-definiteness* due to the existence of missing values, which makes $J^0 \notin S$ and $J^0 \notin R$.

Denote by P_R the projection operator onto R and denote by J_R^0 the projection of J^0 onto R, i.e., $J_R^0 = P_R\left(J^0\right)$. The work of (Li, 2015) utilizes the following result.

Theorem 1 $\left\|J^* - J_R^0\right\|_F^2 \leq \left\|J^* - J^0\right\|_F^2$, *where* $\|\cdot\|_F$ *denotes the Frobenius norm of a matrix. The equality holds if and only if* $J^0 \in R$, *i.e.,* $J^0 = J_R^0$.

This key observation shows that projecting J^0 onto the feasible region R will produce an improved estimate towards the unknown true J^*.

Now we can study the following minimization problem:

$$\min_{J \in R} \left\|J - J^0\right\|_F^2. \tag{1}$$

This is a standard convex problem (Boyd and Vandenberghe, 2004). The unique minimizer to the problem is the projection of J^0 onto R. Based on the observation in Theorem 1, seeking the minimizer to Eq. (1) will produce a *better* affinity matrix than J^0, in terms of a shorter distance from J^* measured in the Frobenius norm.

Considering that $R = S \cap T$, the problem becomes seeking the projection of J^0 onto the intersection of S and T with respect to the Frobenius norm. Directly computing J_R^0 is expensive, while computing the optimal projections of any matrix onto S and T respectively are relatively easier. Denote by P_S the projection operator onto S, and P_T the projection operator onto T. For projection onto S, we have:

Fact 1 *For a real symmetric matrix* $X \in M_n$ *and its singular value decomposition* $X = U\Sigma V^T$ *where* $\Sigma = diag\left(\lambda_1, \cdots, \lambda_n\right)$, *the projection of* X *onto* S *is given by:* $X_S = P_S\left(X\right) = U\Sigma' V^T$ *where* $\Sigma' = diag\left(\lambda_1', \cdots, \lambda_n'\right)$ *and*

$$\lambda_i' = \begin{cases} \lambda_i, & \text{if } \lambda_i \geq 0 \\ 0, & \text{otherwise} \end{cases}.$$

The matrix $X_S = P_S\left(X\right)$ gives the positive semi-definite matrix that most closely approximates X with respect to the Frobenius norm (Knol and ten Berge, 1989; Higham, 2002). For projection onto T, we have:

Fact 2 *For a given matrix* $X \in M_n$, *its projection onto* T, $X_T = P_T\left(X\right)$, *is given by*

$$\left(X_T\right)_{ij} = \begin{cases} X_{ij}, & \text{if } \ell_{ij} \leq X_{ij} \leq \mu_{ij} \\ \ell_{ij}, & \text{if } X_{ij} < \ell_{ij} \\ \mu_{ij}, & \text{if } X_{ij} > \mu_{ij} \end{cases}.$$

With the two facts, we are able to apply Dykstra's algorithm (Dykstra, 1983) to find the minimizer to Eq. (1). Starting from J^0, the algorithm generates two sequences, the iterates $\left\{J_S^t, J_T^t\right\}$ and the increments $\left\{I_S^t, I_T^t\right\}$, by:

$$\begin{aligned} J_S^t &= P_S\left(J_T^{t-1} - I_S^{t-1}\right) && (2)\\ I_S^t &= J_S^t - \left(J_T^{t-1} - I_S^{t-1}\right) && (3)\\ J_T^t &= P_T\left(J_S^t - I_T^{t-1}\right) && (4)\\ I_T^t &= J_T^t - \left(J_S^t - I_T^{t-1}\right) && (5) \end{aligned}$$

where $J_T^0 = J^0$, $I_S^0 = \mathbf{0}$ (an $n \times n$ zero matrix), and $t = 1, 2, \cdots$. The two sequences $\left\{J_S^t\right\}$ and $\left\{J_T^t\right\}$ converge to the optimal solution J_R^0 as $t \to \infty$.

3.2. Scalability Challenge

The matrix calibration approach given in Section 3.1 provides a principled way to calibrate an affinity matrix with an improved estimate. It runs reasonably efficient on a mainstream workstation for problems with a few thousand or less samples. However, when the number of samples grows, significant challenge arises in scalability issues.

The challenge comes from the projection of a matrix onto the subset S. It involves singular value decomposition of the matrix, which has a time complexity of $O\left(n^3\right)$ for an $n \times n$ matrix (Golub and Van Loan, 1996; Cline and Dhillon, 2006). The required computation grows fast and becomes infeasible when n is large.

A natural solution to the challenge is to resort to parallel computing. Unfortunately, the development of parallel algorithms for singular value decomposition is nontrivial. Existing results mostly focus on the decomposition of sparse matrices, while developing a parallel solution to decomposing dense matrices, as in our case, still faces significant difficulty (Berry et al., 2005).

3.3. Projection onto Supersets

To develop a scalable matrix calibration method, our proposal is based on the following observation.

Lemma 1 *Let R be a closed convex subset of M_n and $J^* \in R$. Let C be a closed convex superset of R and $C \subseteq M_n$. For any $J^0 \in M_n$, we have $\left\|J^* - J_C^0\right\|_F^2 \leq \left\|J^* - J^0\right\|_F^2$. The equality holds if and only if $J^0 \in C$, i.e., $J^0 = J_C^0$.*

Obviously this is a direct result from Theorem 1. Based on this observation, we consider a divide-and-conquer approach. Let $C_1, \cdots, C_r$ be r supersets of R that satisfy $\bigcap_{k=1}^r C_k = C$ and $\bigcup_{k=1}^r C_k \subseteq M_n$. Starting from $J^0 \in M_n$, Dykstra's algorithm generates two sequences, the iterates $\left\{J_k^t\right\}$ and the increments $\left\{I_k^t\right\}$, by:

$$\begin{aligned} J_0^t &= J_r^{t-1} && (6)\\ J_k^t &= P_{C_k}\left(J_{k-1}^t - I_k^{t-1}\right) && (7)\\ I_k^t &= J_k^t - \left(J_{k-1}^t - I_k^{t-1}\right) && (8) \end{aligned}$$

where $k = 1, \cdots, r$ and $t = 1, 2, \cdots$. The initial values are given by $J_r^0 = J^0$ and $I_i^0 = \mathbf{0}$.

By assuming that C is not empty (i.e., $C \neq \Phi$), the sequences of $\{J_k^t\}$ converges to J_C^0 with theoretical guarantee (Dykstra, 1983; Escalante and Raydan, 2011).

Theorem 2 *Let $C_1, \cdots, C_r$ be closed and convex subsets of M_n such that $C = \bigcap_{k=1}^r C_k \neq \Phi$. For any $J^0 \in M_n$ and any $k = 1, \cdots, r$, the sequence $\{J_k^t\}$ converges strongly to $J_C^0 = P_C(J^0)$, i.e., $\|J_k^t - J_C^0\|_F^2 \to 0$ as $t \to \infty$.*

3.4. A Cyclic Projection Algorithm

Algorithm 1: Cyclic projection of J^0 onto $C = \bigcap_{k=1}^r C_k$

function $[J_C^0] = CyclicProjection(J^0, \{C_1, \cdots, C_r\})$
  $t \leftarrow 0$
  $J_r^0 \leftarrow J^0$
  $I_i^0 \leftarrow \mathbf{0}$
  repeat
    $t \leftarrow t + 1$
    $J_0^t \leftarrow J_r^{t-1}$
    for $k \leftarrow 1$ **to** r **do**
      $J_k^t \leftarrow P_{C_k}\left(J_{k-1}^t - I_k^{t-1}\right)$
      $I_k^t \leftarrow J_k^t - \left(J_{k-1}^t - I_k^{t-1}\right)$
    end
  until *CONVERGENT*
  return $J_C^0 \leftarrow J_r^t$
end

Different designs are possible to realize the r supersets for given R. We consider the following scheme. Let r nonempty index sets be given as $I_1, \cdots, I_r$ which satisfy $\bigcup_{k=1}^r I_k = \{1, \cdots, n\}$. For any matrix $A \in M_n$, denote by A_k the principal submatrix formed by selecting the same rows and columns of A indicated by I_k. Then for each I_k $(1 \leq k \leq r)$, define

$$S_k = \{A \mid A \in M_n, \text{ and } A_k \succeq 0\}$$

and

$$C_k = S_k \cap T.$$

Consider a classical fact that a matrix is positive semi-definite if and only if all its principal submatrices are positive semi-definite (Horn and Johnson, 2012). In our problem, we can't directly apply this fact to ensure the positive semi-definiteness of a matrix, as it involves similar or even more computation (than performing singular value decomposition on the whole matrix) to calibrate, one by one, these $O(2^n)$ principal submatrices. Here we consider a relaxation approach. Instead of calibrating all principal matrices of a given matrix J^0 to be positive semi-definite, we calibrate a random selection of r principal submatrices only, which are indicated by the index sets $I_1, \cdots, I_r$. This can be done by projecting the matrix onto each S_k (and T to ensure the *boundedness* requirement) cyclically through Dykstra's algorithm.

Algorithm 2: Parallel projection of J^0 onto $C = \bigcap_{k=1}^{r} C_k$

function $\left[J_C^0\right] = ParallelProjection\left(J^0, \{C_1, \cdots, C_r\}\right)$
  $t \leftarrow 0$
  repeat
    $t \leftarrow t + 1$
    parfor $k \leftarrow 1$ **to** r **do**
      $J_k^t \leftarrow P_{C_k}\left(J^{t-1}\right)$
    end
    $J^t \leftarrow \frac{1}{r}\sum_{k=1}^{r} J_k^t$
  until *CONVERGENT*
  return J^t
end

Repeating the procedure illustrated above for different index sets, we reach a simple iterative algorithm:

1. *Randomly generate r index sets $I_1, \ldots, I_r$;*
2. *Project the matrix onto $C_1, \cdots, C_r$ cyclically;*
3. *Repeat steps (1) and (2) until convergence.*

In each iteration of the algorithm, we project a matrix onto C, which can be solved by Algorithm 1. The algorithm starts from an initial or random guess of the solution J^0 and performs the projection onto each subset $C_k = S_k \bigcap T$ cyclically by carrying out the updates shown in Eqs. (7) and (8).

The algorithm's major computation comes from computing $P_{C_k}(J)$, the projection of a matrix $J \in M_n$ onto each C_k. The projection can be obtained in two steps. Firstly, the principal submatrix indicated by I_k needs to satisfy the *positive semi-definiteness* and the *boundedness* constraints. This can be ensured via projecting an $|I_k| \times |I_k|$ principal submatrix onto the subset C_k by applying the basic Dykstra's calibration algorithm given in Section 3.1, which has a much smaller complexity than projecting an $n \times n$ matrix J.

Secondly, all other elements of $P_{C_k}(J)$ need to satisfy the *boundedness* constraint only, which can be trivially computed based on Fact 2 in Section 3.1. By merging the results from the two steps, the projection $P_{C_k}(J)$ is obtained.

3.5. A Parallel Projection Algorithm

In addition to the cyclic algorithm, a parallel matrix calibration algorithm can be implemented in three steps:

1. *Randomly generate r index sets $I_1, \ldots, I_r$;*
2. *Project the matrix onto $C_1, \cdots, C_k$ in parallel and merge the results;*
3. *Repeat steps (1) and (2) until convergence.*

The details of step (2) is given in Algorithm 2. Different from the cyclic update scheme, the major computation of projecting onto the r supersets $C_1, \cdots, C_r$ of C can be executed in parallel. Then these r projections $J_1^t, \cdots, J_r^t$ are averaged to form a new estimate J^t. It can be shown that the sequence $\left\{J^t\right\}_{t\geq 0}$ is *Fejér monotone with respect to* C (Butnariu et al., 2001; Escalante and Raydan, 2011), which satisfies $\left\|J^t - J\right\|_F^2 \leq \left\|J^{t-1} - J\right\|_F^2$, $(\forall t > 0, \forall J \in C)$. Considering that $J^* \in R$ and $R \subseteq C$, we have

$$\left\|J^t - J^*\right\|_F^2 \leq \cdots \leq \left\|J^0 - J^*\right\|_F^2, \forall t > 0. \tag{9}$$

Therefore, the sequence of J^t $(t > 0)$ improves the estimate of J^0 towards the true J^*.

3.6. Accelerations

It is worth mentioning that Dykstra's algorithm can be accelerated (López and Raydan, 2016). The key technique relies on the introduction of a suitable subspace in the product space $M_n \times M_n$. The acceleration intersects a conveniently defined line with this subspace after each iteration of projections. The acceleration keeps the optimality of Dykstra's algorithm, and guarantees termination at the solution. With the acceleration, the convergence speed of Dykstra's algorithm can often be significantly increased.

Based on the Dykstra's algorithm, our proposed approaches will potentially benefit from this acceleration scheme. A detailed discussion goes beyond the scope of this paper and is therefore omitted.

3.7. Complexity Analysis

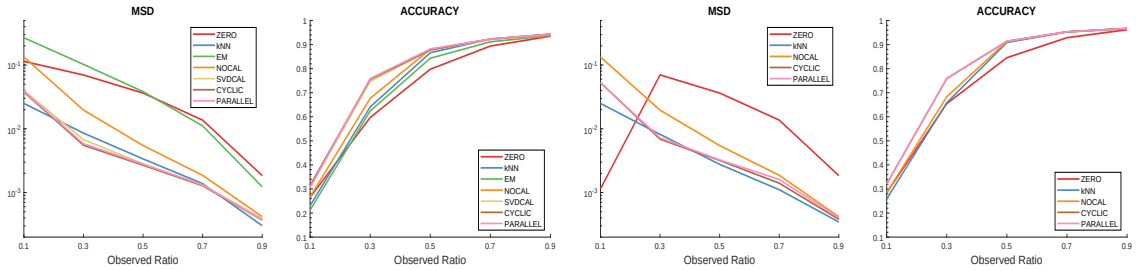
A key concern to the complexity of the calibration approaches is the computation required to project onto the feasible region. The work of (Li, 2015) projects an $n \times n$ matrix directly to S, the set of $n \times n$ positive semi-definite matrices. Fact 1 shows that the major computation comes from the singular value decomposition of the matrix. The complexity is $O\left(n^3\right)$ [1] (Golub and Van Loan, 1996; Cline and Dhillon, 2006).

Our work proposes to project the matrix to S_k $(1 \leq k \leq r)$ cyclically, which requires the principal submatrix indicated by the index set I_k to be positive semi-definite. When setting the cardinality of the index set I_k to $O\left(\frac{n}{r}\right)$, the complexity of decomposing this submatrix is $O\left(\frac{n^3}{r^3}\right)$. In each iteration, we need to repeat the projection r times, and the complexity becomes $O\left(\frac{n^3}{r^2}\right)$ in total, which brings a significant improvement.

Another concern is about the number of iterations needed to converge. Theoretically the convergence rate of Dykstra's algorithm for polyhedral sets is linear (Deutsch, 2001), which coincides with the convergence rate of von Neumann's alternating projection method (von Neumman, 1955). Empirically our proposed approaches used around 30 to 40 iterations (generations of different index sets) to converge on a problem with $10K$ samples [2], and used 50 to 100 iterations on a problem with $70K$ samples.

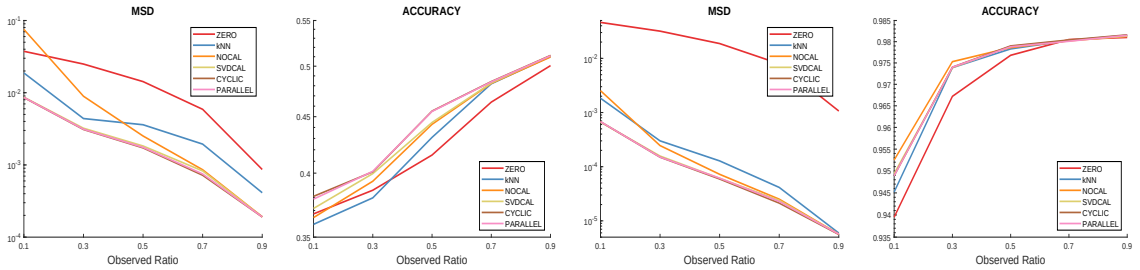
For the memory requirement, the proposed approaches need to store the affinity matrix with the complexity of $O\left(n^2\right)$, which is the same as the method proposed in (Li, 2015).

1. The complexity of projecting onto T is trivial and omitted.
2. For short, "K" is used to denote one thousand in this paper.



(*a*) MNIST(10K) (*b*) MNIST(10K) (*c*) MNIST(70K) (*d*) MNIST(70K)

Figure 1: **MSD and Accuracy on MNIST dataset (10K and 70K samples respectively) with cosine similarity model**. X-axis: observed ratio; Y-axis: *MSD* or classification accuracy. Under all settings with the observed ratio varying from 0.1 to 0.9, *CYCLIC* and *PARALLEL* provided comparable results to *SVDCAL*, while significantly improving the results of the other methods. The results of *EM* and *SVDCAL* were not available on *MNIST* (70K) for computation restrictions.



(*a*) PROTEIN(10K) (*b*) PROTEIN(10K) (*c*) WEBSPAM(10K) (*d*) WEBSPAM(10K)

Figure 2: **MSD and Accuracy on PROTEIN and WEBSPAM datasets (10K samples each) with Jaccard index model**. X-axis: observed ratio; Y-axis: *MSD* or classification accuracy. Under all settings with the observed ratio varying from 0.1 to 0.9, *CYCLIC* and *PARALLEL* provided comparable results to *SVDCAL*, while significantly improving the results of the other methods. The results of *EM* were not available for computation restrictions.

4. Evaluation

We carried out a series of experiments to verify the performance of the proposed methods, with three objectives:

- To evaluate the effectiveness in calibrating the affinity matrix, measured by the mean square deviation of the calibrated matrix from the ground-truth.
- To evaluate the effectiveness of the calibrated matrix in machine learning applications, measured by the classification accuracy.

- To evaluate the scalability to large-scale problems, measured by the actual and the expected running time on problems of different sizes.

The same as (Li, 2015), benchmark datasets from three application domains were included in the experiments.

- MNIST (LeCun et al., 1998): an image database of $70K$ handwritten digits ("0" to "9"). After preprocessing, each image is represented as a 784-dimensional binary vector.
- PROTEIN (Chang and Lin, 2011): a bioinformatics database with three classes of instances. Each instance is represented as a 357-dimensional sparse binary vector.
- WEBSPAM (Wang et al., 2012): a dataset with spam and non-spam web pages. Each page is represented as a binary vector. The data are highly sparse. On average one vector has about $4K$ non-zero values out of more than 16 million features.

Our two proposed algorithms, denoted by *CYCLIC* (the cyclic projection algorithm) and *PARALLEL* (the parallel projection algorithm) respectively, were compared with a number of imputation methods that are popularly used in practice. We applied these imputation methods to replace missing observations with substituted values and then calculated the affinity matrix. Besides, we also included the results of *NOCAL* and *SVDCAL* as the baselines.

- ZERO: Replace all missing elements by zero.
- kNN: Replace each missing element of a sample by the median over k nearest neighboring samples based on the observed features. The value of k was iterated from 1 to 5 and the best result was reported, which actually overestimated the performance of this approach.
- EM: The missing elements are imputed by the classical expectation maximization algorithm (Dempster et al., 1977). An implementation from (Ghahramani and Jordan, 1994) were used.
- NOCAL: The affinity matrix is best calculated from the observed data. That is, for two samples, the similarity is calculated based on their commonly observed features. The matrix J^0 was obtained with this approach.
- SVDCAL: The calibration approach proposed in (Li, 2015) that applies singular value decomposition on the whole affinity matrix.

In addition to the algorithms above, we also tried to apply the low-rank matrix recovery algorithm (Wright et al., 2009) as an imputation method [3]. But unfortunately the experiment failed. A thorough inspection found that the low-rank assumption does not hold on these datasets, which made an unfair comparison to the matrix recovery algorithm. So the results from the algorithm are omitted here.

3. Code downloaded from https://people.eecs.berkeley.edu/~yima/matrix-rank/sample_code.html.

Although in theory our proposed algorithms can be applied on all affinity models with the characteristics of *positive semi-definiteness* and *boundedness*, our experiments specifically investigated two affinity models that are widely used in machine learning and information retrieval tasks (Duda and Hart, 2000; Leskovec et al., 2019).

- Cosine Similarity: a model that measures the cosine of the angle between two non-zero vectors of an inner product space, with each value being within $[-1, 1]$.
- Jaccard Index: a model that measures the size of the intersection of any two samples' features divided by the size of the union of their features, with each value being within $[0, 1]$.

All the computations were carried out on a mainstream 8-way server with 224 CPU cores sharing 1.5TB memory. All the codes were implemented on *MATLAB* platform with *intel MKL* as the maths library.

4.1. Calibration

On *MNIST* dataset, we randomly selected $10K$ samples. For each sample, different portions (from 10% to 90%) of randomly chosen feature values were assumed to be observed.

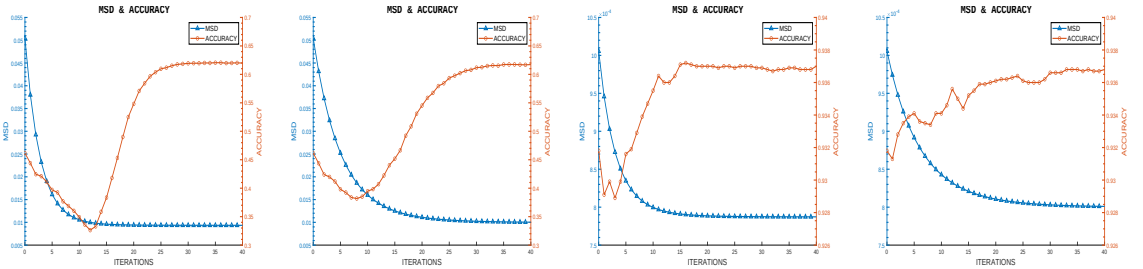
We firstly built an approximate *cosine similarity* matrix J^0 based on the incomplete data with the *NOCAL* approach. Then we applied the calibration approaches on the matrix to meet the *positive semi-definiteness* and the *boundedness* constraints. For the imputation approaches, the *cosine similarity* matrix was calculated directly from the imputed data.

The results are presented through the comparison of mean square deviations (*MSD*) from the true *cosine similarity* matrix J^* which was computed from the fully observed data. For any $n \times n$ matrix J, its *MSD* from J^* is defined as the square Frobenius distance between the two matrices, divided by the number of elements, ie., $\frac{\sum_{ij=1}^{n}\left(J_{ij}-J^*_{ij}\right)^2}{n^2}$.

The results are shown in Figure 1. Comparing with *NOCAL*, consistently improved results were achieved by the calibration approaches. The improvement is especially significant when the ratio of observed features is low. When the ratio is below 0.5, the calibration brought about 2 to more than 10 times smaller *MSD* values from the un-calibrated matrices. As an example, the *MSD* value was reduced from 6.989×10^{-2} to 5.525×10^{-3} when the observed ratio is 30%. Comparing with the imputation approaches (*ZERO*, *kNN*, *EM*), evidently smaller *MSD* values were observed in most experiments. When comparing the three calibration approaches, *CYCLIC* and *PARALLEL* provided highly comparable, if not better than, results to *SVDCAL*.

In the experiment with the full *MNIST* dataset of $70K$ samples, the results of *EM* and *SVDCAL* were not available due to the prohibitive computation required. Similarly improved results were observed under all settings of observed ratios, which verified the effectiveness of the two proposed calibration approaches.

In addition to the *cosine similarity* model, we further evaluated the performance of the proposed approaches on *Jaccard index* model with *PROTEIN* and *WEBSPAM* datasets of $10K$ samples each. The results are shown in Figure 2, where our proposed approaches reported similar improvements on reducing the *MSD* from the true affinity matrix over the uncalibrated matrix and over the results from the imputation algorithms.



(*a*) CYCLIC(20%) (*b*) PARALLEL(20%) (*c*) CYCLIC(80%) (*d*) PARALLEL(80%)

Figure 3: **MSD and Accuracy on MNIST dataset (10K samples) with cosine similarity model**. X-axis: iterations; Y-axis: (left) *MSD*, (right) classification accuracies. The *CYCLIC* and *PARALLEL* methods were recorded with 20% and 80% observed features. On all settings, the two proposed algorithms showed fast convergence and their results evidently improved the results of *NOCAL* (shown at iteration = 0).

4.2. Classification

The second experiment investigated whether the reduced *MSD* could benefit practical applications. Specifically, we applied the calibrated affinity matrices in nearest neighbor classification tasks. Given a training set of labeled samples, we tried to predict the labels of the samples in the testing set. For each testing sample, its label was determined by the label of the sample in the training set that had the largest affinity value with it.

The experiment was carried out with $10K$ and $70K$ samples and different portions of observed values from 10% to 90% respectively. In each run, 10-fold cross validation was used and the mean accuracies were reported. Similarly from the classification results shown in Figures 1 and 2, the calibration methods reported the best results in most experiments and consistently improved the results of *NOCAL*. As an example, on *MNIST* (10K) with 30% observed features, the classification accuracy sharply increased around 10% through calibration.

Similar improvements were observed on *PROTEIN* and *WEBSPAM* datasets under different experimental settings. All these results successfully verified the benefits brought by the reduced deviation from the true affinity matrix, and justified the usefulness of the calibration methods in machine learning tasks.

4.3. Convergence and Scalability

Figure 3 shows the convergence of the two proposed approaches on *MNIST* dataset with $10K$ samples and 20% and 80% observed feature values. In the experiment, r was set to 8 and each index set I_k had around $2.5K$ elements [4]. The results clearly demonstrated that, in around 30 to 40 iterations, the two calibration algorithms converged in both *MSD* and classification accuracy.

4. These index sets were overlapping and each index number roughly appeared in two index sets.

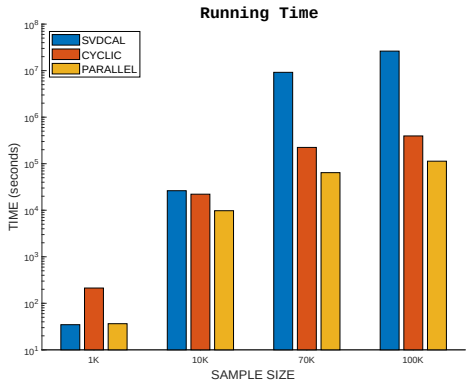


Figure 4: **Real and Estimated Running Time by different calibration methods**. X-axis: number of samples (n); Y-axis: time in seconds (logscale). The estimations of *SVDCAL*-$70K$/$100K$, *CYCLIC*-$100K$ and *PARALLEL*-$100K$ were based on 50 iterations. For $n = 1K/10K$, $r = 8$; for $n = 70K/100K$, $r = 80$. $|I_k| \approx \frac{2n}{r}$ $(1 \leq k \leq r)$. The proposed algorithms exhibited evident improvement in running time when the sample size becomes large.

Next, we compared the running time by the three calibration methods. For *CYCLIC* and *PARALLEL*, we recorded their real running time with $1K$, $10K$ and $70K$ samples and estimated their time with $100K$ samples based on the complexity analysis in Section 3.7 and the real performance on smaller datasets. For *SVDCAL*, we recorded its running time with $1K$ and $10K$ samples, and estimated its time with $70K$ and $100K$ samples. Here, the execution was regarded as converged if the relative improvement on *MSD* was smaller than 0.1% from the MSD of last iteration.

In the experiment, both *SVDCAL* and *CYCLIC* were allowed to use at most 50 parallel threads. *PARALLEL* run on 8 CPU nodes, and each node was allowed to use at most 50 parallel threads.

As shown in Figure 4, *CYCLIC* and *PARALLEL* did not have significantly improved running efficiency on small problems. With $1K$ samples, *CYCLIC* was even several times slower than *SVDCAL*, and *PARALLEL* was comparable to *SVDCAL*. With $10K$ samples, the running time of *CYCLIC* and *SVDCAL* became comparable. The improvement from parallel execution was still not significant.

With $70K$ samples, the two proposed approaches exhibited their full advantages. They were expected to be fifty to more than a hundred time faster than *SVDCAL*. It was estimated that over 100 days would be needed to run *SVDCAL* to calibrate a $70K \times 70K$ matrix in 50 iterations. Comparatively, *CYCLIC* spent around 62 hours to converge. More significantly, *PARALLEL* spent less than 20 hours. Comparing with *SVDCAL*'s fast growing $O\left(n^3\right)$ complexity, the time needed by the proposed approaches grows much smoother when the sample size increases, which provides a scalable solution for large-scale data processing applications.

5. Conclusion

Estimating pairwise affinity between data samples is a fundamental problem in data science and machine learning. In practice, it is hard to perform the estimation when there are incompletely observed data samples. Instead of trying to impute the missing values, our work followed the idea of matrix calibration and proposed two simple yet efficient methods for large-scale calibration problems. Specifically, we addressed the scalability challenge to the existing matrix calibration method that resides on decomposing a full affinity matrix. It was shown that, by dividing a matrix and calibrating the smaller submatrices one by one, we can also reach an improved estimate with guarantee. In our experiments, the calibration approaches reported superior results over the classical imputation approaches in estimating affinity matrices from incomplete observations.

Our work provided another successful example that the idea of "divide-and-conquer" works empirically well in machine learning and data analysis applications (Li et al., 2007). The proposed algorithms reported high-quality calibration results as the state-of-the-art calibration method. At the same time, the new algorithms brought significant improvement in scalability. Problems that cannot be calibrated by the existing approach can now be tackled with the proposed algorithms.

To calibrate a smaller submatrix, our work resorts to the classical singular value decomposition method. For future development of our work, it may be possible to seek modern alternatives to solve these sub-problems, such as (Cheng and Higham, 1998; Sun et al., 2020). The combination of these modern solvers with our proposed framework deserves further investigation.

Considering the improvement and the benefit brought by the proposed algorithms, we strongly believe that our work provides a highly practical tool, which potentially benefit various tasks such as in kernel approximation problems (Gisbrecht and Schleif, 2015), in recommender systems (Resnick and Varian, 1997) and in large-scale genome analysis (Mount, 2001). Future applications along these lines are highly expected.

Acknowledgments

We thank all the anonymous reviewers for their helpful comments and suggestions. The work is partially supported by Shenzhen Fundamental Research Fund (JCYJ20170306141038939, KQJSCX20170728162302784).

References

M. Berry, D. Mezher, B. Philippe, and A. Sameh. Parallel algorithms for the singular value decomposition. In *Handbook of Parallel Computing and Statistics*, pages 133–180. Chapman and Hall/CRC, 2005.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

D. Butnariu, S. Reich, and Y. Censor. *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, volume 8. Elsevier, 2001.

C. Chang and C. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.

S. Cheng and N. Higham. A modified cholesky algorithm based on a symmetric indefinite factorization. *SIAM Journal on Matrix Analysis and Applications*, 19(4):1097–1110, 1998.

A. Cline and I. Dhillon. Computation of the singular value decomposition. In *Handbook of Linear Algebra*, pages 45–1. Chapman and Hall/CRC, 2006.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

F. Deutsch. *Best Approximation in Inner Product Spaces*. Springer, 2001.

R. Duda and P. Hart. *Pattern Classification*. John Wiley and Sons, 2000.

R. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

C. Enders. *Applied Missing Data Analysis*. Guilford Press, 2010.

R. Escalante and M. Raydan. *Alternating Projection Methods*. SIAM, 2011.

Z. Ghahramani and M. Jordan. Supervised learning from incomplete data via an EM approach. In *Advances in Neural Information Processing Systems*, volume 6, pages 120–127. Morgan Kaufmann, 1994.

A. Gisbrecht and F. Schleif. Metric and non-metric proximity transformations at linear costs. *Neurocomputing*, 167:643–657, 2015.

G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

N. Higham. Computing the nearest correlation matrix - a problem from finance. *IMA Journal of Numerical Analysis*, 22:329–343, 2002.

R. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, 2012.

P. Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912.

A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31 (3):264–323, 1999.

D. Knol and J. ten Berge. Least-squares approximation of an improper correlation matrix by a proper one. *Psychometrika*, 54(1):53–61, 1989.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

J. Leskovec, A. Rajaraman, and J. Ullman. *Mining of Massive Data Sets*. Cambridge University Press, 2019.

W. Li. Estimating Jaccard index with missing observations: a matrix calibration approach. In *Advances in Neural Information Processing Systems*, pages 2620–2628, 2015.

W. Li, K.H. Lee, and K.S. Leung. Large-scale RLSC learning without agony. In *Proceedings of the 24th International Conference on Machine Learning*, pages 529–536, 2007.

R. Little and D. Rubin. *Statistical Analysis with Missing Data*, volume 793. John Wiley & Sons, 2019.

W. López and M. Raydan. An acceleration scheme for Dykstra's algorithm. *Computational Optimization and Applications*, 63(1):29–44, 2016.

D. Mount. *Bioinformatics: Sequence and Genome Analysis*, volume 1. Cold Spring Harbor Laboratory Press, 2001.

P. Resnick and H. Varian. Recommender systems. *Communications of the ACM*, 40(3): 56–58, 1997.

D. Rogers and T. Tanimoto. A computer program for classifying plants. *Science*, 132(3434): 1115–1118, 1960.

G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

B. Scholköpf and A. Smola. *Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001.

D. Sun, K. Toh, Y. Yuan, and X. Zhao. Sdpnal+: A matlab software for semidefinite programming with bound constraints (version 1.0). *Optimization Methods and Software*, 35(1):87–115, 2020.

J. von Neumman. *Mathematical Foundations of Quantum Mechanics*. Princeton University Press, 1955.

D. Wang, D. Irani, and C. Pu. Evolutionary study of web spam: Webb spam corpus 2011 versus webb spam corpus 2006. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 40–49. IEEE, 2012.

J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in Neural Information Processing Systems*, pages 2080–2088, 2009.