IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 19, 2018, accepted October 1, 2018, date of publication November 1, 2018, date of current version December 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2879011

CPCA: A Feature Semantics Based Crowd Dimension Reduction Framework

YUANYUAN ZHANG[1,2], DAWEI GAO[1], JIE LUO[1], AND KE XU[1]
[1]State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[2]China Academy of Telecommunication Technology, Beijing 100191, China

Corresponding authors: Jie Luo (luojie@nlsde.buaa.edu.cn) and Ke Xu (kexu@nlsde.buaa.edu.cn)

This work was supported by NSFC under Grant 61822201 and Grant 61502021.

ABSTRACT Dimension reduction plays an important role in practical big data analysis and data mining applications. However, popular dimension reduction techniques, such as principal component analysis (PCA), are known to be computation-intensive and are considered as a computation bottleneck for data processing and mining. In this paper, we propose to reduce the computation of PCA via crowdsourcing, a paradigm that accomplishes hard-to-compute problems leveraging collective intelligence. We design CPCA, crowd principal component analysis, a novel crowd-based dimension reduction framework. The CPCA designs tasks for crowd workers to obtain the relations among features based on their semantics and formulates a weighted graph from the collected answers to derive the covariance matrix and the principal components. We prove the correctness of CPCA and conduct extensive evaluations on real datasets. Experimental results show that CPCA could achieve significantly reduction on the computational cost in terms of both time and memory, which lowers the bar for learning.

INDEX TERMS Dimensionality reduction, crowdsourcing, principal component analysis, machine learning.

I. INTRODUCTION

Dimension reduction reduces the computation and storage for data processing and can improve the performance of learning models by removing multi-collinearity [1]. It has been widely adopted in big data mining and knowledge processing applications [2]. Various dimension reduction techniques have been developed [1], [3], such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Singular Value Decomposition (SVD), kernel-PCA [4], kernel discriminant analysis, embedding [5], [6], clustering, low rank approximation [7]–[9] and auto-encoders [10].

However, the process of dimension reduction itself can be a computation bottleneck for big data applications. For example, the calculations of the covariance matrix in PCA takes $O(n^2m)$ time and $O(nm)$ space, where n is the number of dimension and m is the number of instances. Though in the cloud-based situation, the data can be fit into the cloud servers, it is too expensive for the individuals to rent cloud servers.

In this paper, we ask the question: *can we leverage humans to perform dimension reduction to reduce its computation?* We make two observations: *(i)* Mainstream linear dimension reduction techniques such as PCA operates by projecting *related* or *dependent* features into a new feature space with lower dimensions. *(ii)* Many real-world data are associated with semantic meanings and it is easy for humans to identify whether these semantics are correlated. For instance, it is easy for an ordinary person to identify that the features of 'weather' and 'temperature' are correlated. Based on these observations, it is viable to exploit human knowledge to identify related features according to their semantics and project them into a low-dimensional space.

It is non-trivial to conduct dimension reduction utilizing human knowledge. Humans make errors and it is common to collect and aggregate answers from a group of people to improve the accuracy, a paradigm called crowdsourcing. Crowdsourcing has been successfully applied to accomplish various works in database and data mining, such as ranking [11], entity resolution [12] and data cleaning [13]. There are two main challenges to design a crowdsourcing solution to dimension reduction: *(i) how to decompose the dimension reduction problem into multiple easy-to-answer tasks for individual crowd workers? (ii) how to construct a covariance matrix using answers to the crowd tasks and further project high-dimensional features into a low-dimensional space?*

We address the above challenges by proposing CPCA, a crowd-based principal component analysis framework for dimension reduction. CPCA designs tasks for crowd workers to obtain the relations among features based on their semantics, collects the answers and formulates a weighted graph to derive the covariance matrix and the principal components. We evaluate the performance of CPCA on a real-world application to reduce the feature space for taxi order prediction. Experimental results show that CPCA saves up to 30% time and storage for dimension reduction than the conventional PCA approach.

The main contributions of this work are as follows.

- We design a novel crowd-based solution to dimension reduction. To the best of our knowledge, it is the first effort to conduct dimension reduction with collective intelligence.
- We conduct extensive evaluations on real data. Experiments show that compared with PCA, our solution can accelerate dimension reduction by up to 30% while achieving comparable accuracy.

In the rest of this paper, we introduce the basics of dimension reduction in Sec. II, present the CPCA framework in Sec. III and evaluate its performance in Sec. IV. Finally we review related work in Sec. V and conclude in Sec. VI.

II. PRELIMINARIES

This section provides a brief introduction of dimension reduction and the motivations to reduce the computation overhead of dimension reduction.

In machine learning and statistics, dimension reduction refers to the process to reduce the random variables under some constraints to reduce noise and obtain a set of principal variables. Formally, the dimension reduction problem is defined as follows.

Definition 1 (Dimension Reduction): Given an N-dimensional dataset D with m instances, project D to a K-dimensional space (usually $K \ll N$) to reduce the noise in D and the computational cost of learning models.

The N-dimensional dataset D refers to a table with N features stored in the relational database, which may be joined from several data tables. We assume the dataset have been preprocessed *e.g.* missing value competition and anomaly detection, using previous studies [14]. The parameter K is defined by the specific applications and the available computation resources.

Principal component analysis (PCA) is an effective and widely adopted dimension reduction technique. Its main idea is to linearly project some features into a new dimension. In the transformation, the first principal component has the largest possible variance, and accounts for the largest variability in the data. Each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The common process is first compute the covariance matrix of the original data. Then it computes and returns the top-k eigenvalues and their corresponding eigenvectors of the covariance matrix. The major computation is the computation for covariances and eigenvalues. Despite its effectiveness, PCA is still considered computation-intensive especially for high-dimensional data, which are common in the big data era. We demonstrate the need to further reduce the computation overhead of PCA via the following example.

Example 1: Consider a task to predict the number of taxi orders in a city using urban big data. Various data sources can be used as candidate features, such as weather, point of interest (POI), road network, etc. *A recent study [15] shows that more than 1,000 dimensions of initial features can be collected and the training of the prediction model is conducted on parameter servers. For developers with limited computation resources, it can be difficult to train a model with such high dimensional features and thus dimension reduction is compulsory. Nevertheless, the computation overhead to perform linear dimension reduction techniques such as PCA can still be large. On a big data with 1000 dimensions, the PCA method needs to compute a* 1000×1000 *covariance matrix and its corresponding eigenvalues.*

This example is not uncommon in the big data era. To reduce the computation of PCA and bring dimension reduction to everyone, we propose to involve humans in the loop and design a crowdsourced dimension reduction framework.

III. CROWD PRINCIPAL COMPONENT ANALYSIS

This section presents CPCA, a crowdsourced dimension reduction framework. We first give an overview of CPCA (Sec. III-A). Then we explain how to design the crowd tasks (Sec. III-B), construct the crowd covariance matrix (Sec. III-C) and calculate the principal components (Sec. III-D). Finally we analyze the correctness of CPCA and its efficiency (Sec. III-E).

A. OVERVIEW

CPCA is a linear dimension reduction algorithm that exploits the crowd to avoid unnecessary computation on convariance. Fig. 1 shows the process of CPCA. In the first step, the collected features are formalized as crowd tasks and distribute to the crowd to collect the answers on the independence between features. Then we gather the answers of the crowd and integrate them into a weighted undirected graph to describe the relations among the features. The vertices denote the features and the weights on the edges represent the independence between two features. Afterwards, we aggregate the vertices into clusters according to their independence values. The clusters tend to be independent with high probabilities. Finally we use these clusters to build the crowd covariance matrix and extract the principal components. We elaborate on the details of CPCA in the following subsections.

B. CROWD TASK DESIGN

We now explain how to design the crowd tasks.

Let $F = \{f_1, f_2, \cdots, f_N\}$ be the N dimensional features of D. In a crowd task, we present a pair of features from F and ask the crowd workers whether these features are independent or not. For example, Fig. 2 shows an example

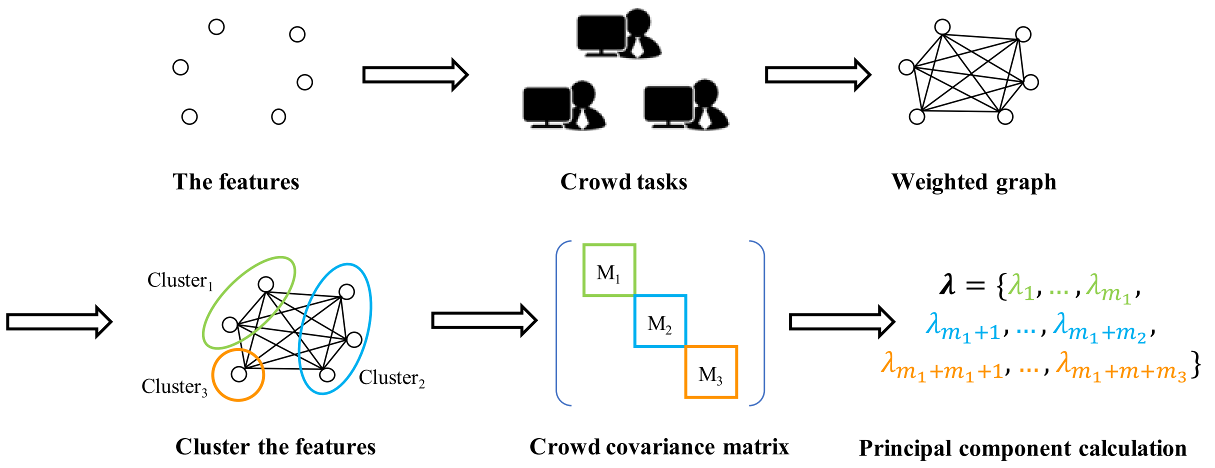
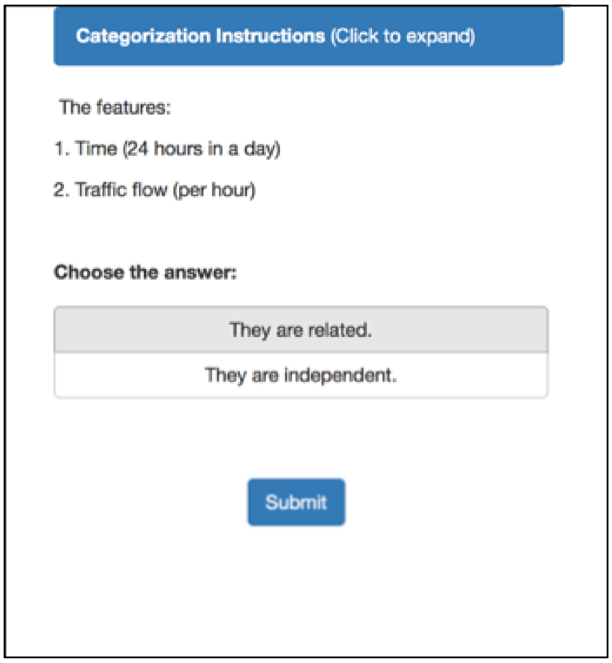

The features
Crowd tasks
Weighted graph
Cluster$_1$
Cluster$_2$
Cluster$_3$
Cluster the features
M$_1$
M$_2$
M$_3$
Crowd covariance matrix
$\boldsymbol{\lambda} = \{\lambda_1, \dots, \lambda_{m_1}, \lambda_{m_1+1}, \dots, \lambda_{m_1+m_2}, \lambda_{m_1+m_1+1}, \dots, \lambda_{m_1+m+m_3}\}$
Principal component calculation

FIGURE 1. An overview of CPCA.


Categorization Instructions (Click to expand)

The features:

1. Time (24 hours in a day)
2. Traffic flow (per hour)

Choose the answer:

They are related.

They are independent.

Submit

FIGURE 2. An example of crowdsourced task.

of a crowdsourced task. We only ask whether two features are independent in a task for the following reasons.

- It is easy for humans to determine whether a pair of features are independent or not based on their semantics using common sense. For instance, the correlation between the features of 'time' and 'traffic flow'. Note that professional knowledge may be required for some tasks. If necessary, we can filter workers and recruit those with specific background knowledge using crowdsourcing platforms such as Amazon Mechanical Turk.
- It is in general difficult for the crowd to further distinguish linear and non-linear relations purely from semantic meanings if two features are dependent. Note that in principle we need answers about whether two features are *linearly* dependent since CPCA is a linear dimension reduction algorithm. We will show in Sec. III-C that such information suffices to conduct linear dimension reduction with high accuracy.

C. CROWD COVARIANCE MATRIX CONSTRUCTION

The core of CPCA is to construct a covariance matrix from the collected answers about independence between features. In this subsection, we first explain how to aggregate the answers into a weighted graph, and then cluster the features and finally derive the covariance matrix.

1) ANSWER AGGREGATION

Assume for a pair of features (f_a, f_b), we collect β answers from the crowd. That is, we collect answers about the independence between f_a and f_b) as $F_i(f_a, f_b)$, $i = 1, \cdots, \beta$, where the function $F_i(f_a, f_b)$ equals 1 if the answer of (f_a, f_b) is dependent, and zero otherwise. We apply weighted voting to aggregate these answers.

Specifically, we use weighted voting to integrate the answers as follows.

$$c(f_a, f_b) = \frac{\sum_{i=1}^{\beta} w_i F_i(f_a, f_b)}{\sum_{i=1}^{\beta} w_i}, \tag{1}$$

where w_i is the weight of the ith answer, which can be obtained by analyzing the crowd worker's abilities or the accuracy of his/her past answers. When there are not enough historical data to analyze the quality of each worker, we apply "golden tasks" to roughly estimate the quality of workers. Golden tasks are a small portion of tasks with ground truth labels to be used to test the quality of workers via qualification tests or hidden tests [16]. Existing research has shown their effectiveness to control the quality of crowd workers [16].

To obtain the aggregated answers $c(f_a, f_b)$ for every pair of features (f_a, f_b) in F, we need to distribute $O(N^2)$ different tasks, where each task needs to be performed multiple times.

To reduce the number of different tasks needed to be performed, we leverage the transitivity in the dependence among features. Particularly, we make the following observation.

- Dependence among features are transitive if the dependence is linear. Note that we do not differ linear and non-linear dependence in the crowd tasks. However, we may regard all the dependent features as transitive. This way, the only impact would be to leave some independent features in the principal component calculation (Sec. III-D).

Based on the observation, we can reduce the number of different tasks needed to be distributed to the crowd. For example, there are three features $\{f_a, f_b, f_c\}$, and we have already know that the pair of (f_a, f_b) and (f_b, f_c) are dependent. Then we do not need to distribute the task of pair (f_a, f_c) to the crowd since these three features are dependent based on the transitive property. When submitting the tasks, we could first submit a part of them, collect the answers and decide whether to release certain tasks. Using the aggregated answers $c(f_a, f_b)$ for every pair of features in F, we can construct a weighted graph as in Fig. 1, where the weight between f_a and f_b is $c(f_a, f_b)$.

2) FEATURE CLUSTERING

After obtaining the results from the crowd by Eq.1 S, we divide the features with high possibility of linear correlation into the same clusters. To measure the linear correlation, we use a threshold value $\theta \in [0, 1]$ to divide the features. When θ is small, the features will be divided into more clusters, and it will reduce more computation in the principal component calculation. But a θ that is too small will ignore weak linear correlation, which may affect the accuracy of the features. We evaluate the impact of θ on the performance of dimension reduction in Sec. IV.

Alg. 1 shows the process of feature clustering. First, we initialize C as an empty set (Line 1). For each feature $f \in F$, we find all the features with a weight larger than θ and put them into a cluster C (Lines 3-11). Finally we obtain the collection of feature sets $C = \{C_1, C_2, \cdots, C_{|C|}\}$ (Line 13).

3) COVARIANCE MATRIX CALCULATION

Finally we build a matrix M according to the collection $C = \{C_1, C_2, \cdots, C_{|C|}\}$, where $C_i = \{f_1^i, f_2^i, \cdots, f_{|C_i|}^i\}$.

$$M = \begin{bmatrix} M_1 & & & \\ & M_2 & & \\ & & \ddots & \\ & & & M_{|C|} \end{bmatrix}, \quad (2)$$

where $M_i =$

$$\begin{bmatrix} cov(\delta f_1^i, \delta f_1^i) & cov(\delta f_1^i, \delta f_2^i) & \cdots & cov(\delta f_1^i, \delta f_{|C_i|}^i) \\ cov(\delta f_2^i, \delta f_1^i) & cov(\delta f_2^i, \delta f_2^i) & \cdots & cov(\delta f_2^i, \delta f_{|C_i|}^i) \\ \vdots & \vdots & \ddots & \vdots \\ cov(\delta f_{|C_i|}^i, \delta f_1^i) & cov(\delta f_{|C_i|}^i, \delta f_1^i) & \cdots & cov(\delta f_{|C_i|}^i, \delta f_{|C_i|}^i) \end{bmatrix}, \quad (3)$$

Algorithm 1: Divide Features Into Independent Clusters

```
Input: The feature set $F$, score matrix $S$ and threshold $\theta$.
Output: Cluster set $C$.
1  $C \leftarrow \emptyset$;
2  while $F \neq \emptyset$ do
3      sample a feature $f^c$ from $F$;
4      $C' \leftarrow \{f^c\}$;
5      foreach $f \in F - \{f^c\}$ do
6          if $S[f^c][f] \geq \theta$ then
7              $C' \leftarrow f$;
8              $F \leftarrow F - \{f\}$;
9      $C \leftarrow C + C'$;
10 return $C$;
```

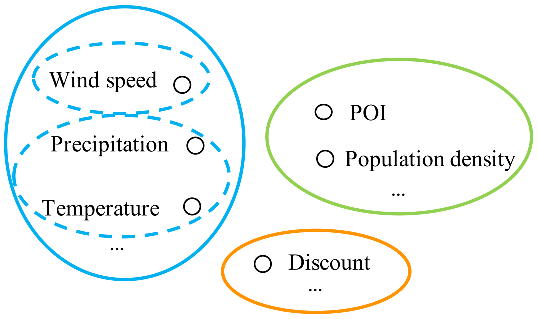


FIGURE 3. An example of clusters of features collected from answers from the crowd.

where $\delta f_j^i = f_j^i - \overline{f_j^i}, \overline{f_i^j}$ is the average of the feature f_j^i, and the function $cov(\cdot, \cdot)$ is to compute the covariance of two specific features.

M consists of many square matrices $\{M_1, M_2, \cdots, M_{|C|}\}$. M_i is the covariance matrix of the feature set $C_i = \{f_1^i, f_2^i, \cdots, f_{|C_i|}^i\}$. Note that the order of the covariance matrices in M does not affect the performance of CPCA. M can be seen as a simplified covariance matrix of F. Since we know features in different C_i are independent and their covariances should be equal or close to zero, we can safely omit them and only compute the covariance of the linearly dependent features, *i.e.* covariances in $M_1, \cdots, M_{|C|}$.

Example 2: Back to the example of taxi order prediction. We first distribute tasks about the independence of each pair of features to crowd workers. Then we collect the answers and compute the weights for these features and construct a weighted graph. Finally the features are divided into different clusters. Fig. 3 shows the answers from the crowd. The features are divided into three primary clusters, which include the weather condition (wind speed, precipitation and temperature), the geographic information (POI and population density) and promotion information (discount). It is natural for the crowd to think that the POI is independent of the temperature and promotion information. For wind speed and the other two features in weather condition, the dependence is weak such that they can be divided into different clusters

$$
\begin{array}{c|cccccc|}
 & \text{Wind speed} & \text{Precipitation} & \text{Temperature} & \text{POI} & \text{Population} & \text{Discount} \\
 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \\
\hline
f_1 & cov(f_1,f_1) & cov(f_1,f_2) & cov(f_1,f_3) & & & \\
f_2 & cov(f_2,f_1) & cov(f_2,f_2) & cov(f_2,f_3) & & & \\
f_3 & cov(f_3,f_1) & cov(f_3,f_2) & cov(f_3,f_3) & & & \\
f_4 & & & & cov(f_4,f_4) & cov(f_4,f_5) & \\
f_5 & & & & cov(f_5,f_4) & cov(f_5,f_5) & \\
f_6 & & & & & & cov(f_6,f_6) \\
 & & & & & & \cdots
\end{array}
$$

FIGURE 4. **The covariance matrix in the example.**

through tuning the threshold. But we can still put them in a cluster and verify their independence through principal component calculation. For promotion information and the geographic information, in general the crowd think they are independent.

D. PRINCIPAL COMPONENT CALCULATION

After obtaining the matrix M, we can derive the principal components for F and conduct dimension reduction similar to the traditional PCA.

As shown in Fig. 1, we compute the eigenvalues of each M_i as λ_i. Note that a large eigenvalue means that this projection has a greater variance and can be distinguished from other classes more easily [17]. We sort all the eigenvalues in the descending order, and extract the K largest eigenvalues $\{\lambda_1, \lambda_2, \cdots, \lambda_K\}$ and their corresponding eigenvectors $V = \{\vec{v_1}, \vec{v_2}, \cdots, \vec{v_K}\}$. Then the final projection of the dataset D is computed as follows.

$$P = DV, D \in R^{m\times N}. \tag{4}$$

P is a vector in $R^{m\times K}$ and we project the original dataset D into a new space with low dimension and large variance.

E. ALGORITHM ANALYSIS

In this subsection we prove the correctness of the principal component calculation and analyze the complexity of CPCA.

1) CORRECTNESS

To prove the process of extracting principal features is correct, we first present and prove Lemma. 1, and then apply it to our scenario.

Lemma 1: If a matrix $M \in R^{n\times n}$ is in the form of

$$M = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}, \tag{5}$$

where $A \in R^{n_1\times n_1}$, $B \in R^{n_2\times n_2}$ and $n_1 + n_2 = n$, then the eigenvalues of M equal the sum of the eigenvalues of A and B.

Proof: To compute the eigenvalues of matrix M in the form of Eq.5, we have

$$\begin{aligned}(M-\lambda I)X &= \begin{bmatrix} A-\lambda I & 0 \\ 0 & B-\lambda I \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \\ &= \begin{bmatrix} (A-\lambda I)X_1 \\ (B-\lambda I)X_2 \end{bmatrix} = 0,\end{aligned} \tag{6}$$

where $X = (X_1^{\mathrm{T}}, X_2^{\mathrm{T}})^{\mathrm{T}}$, $X_1 \in R^{n_1}$ and $X_2 \in R^{n_2}$. Let $\lambda_{A/B}$ and $X_{A/B}$ denote the eigenvalue and eigenvector of A and B. Then we have $(A - \lambda_A I)X_A = 0$ and $(B - \lambda_B I)X_B = 0$. Suppose $\lambda = \lambda_A$, $X_1 = X_A$ and $X_2 = 0$ in Eq.6. Then λ_A and $(X_A^{\mathrm{T}}, 0, \cdots, 0)^{\mathrm{T}} \in R^n$ are the eigenvalue and eigenvector of M. the same proof holds for the eigenvalue of B. Therefore, the eigenvalues of M equal the sum of the eigenvalues of A and B. ■

Now we can easily generalize Lemma. 1 and apply it to the matrix in the form of Eq.2. Consequently, CPCA can obtain the accurate principal features of the crowd covariance matrix.

2) COMPLEXION ANALYSIS

The main computational challenge of CPCA is to compute the covariances and eigenvalues. For the crowd covariance matrix construction, we take the crowdsourcing process as a crowd function like [11] and [12] that can obtain result in $1-2$ days. Otherwise, there are many works studied on stimulation and latency control [18], which is not our focus. In this paper we reduce the computational complexity for both covariances and eigenvalues. First, the time complexity of the original computation is $O(mN^2)$. As for CPCA, assume that there are $|C|$ sets in collection C and the best performance can reach $O(m(\frac{N}{|C|})^2)$. Second, for the eigenvalues it takes $O(n^2)$ for a $n \times n$ matrix, and CPCA reduce the complexity from $O(N^2)$ to $O(\frac{N^2}{|C|})$. Note that the above analysis is under the best condition, because the both the value and the distribution of features in collection C depend on the distribution of dataset. But we would conduct experiments on real dataset in Sec. IV to prove the efficiency of CPCA.

Example 3: Back to our previous example in Fig. 3. Suppose the features are divided into three clusters {Wind speed, Precipitation, Temperature}, {POI, Population density}, {Discount, $\cdots$}. Therefore, we can assume that the features in different clusters are independent. Next we build the covariance matrix like Fig. 4, where we use $\{f_1, f_2, \cdots\}$ to represent the features. In this matrix, we only compute the covariance for the dependent features, while the others are treated as zero. For example, we compute the covariance between the wind speed and precipitation rather than the pair of temperature and POI. The answers of the crowd may be uncertain, but we can decrease the threshold θ to left the uncertain part for the computation of covariance matrix.

IV. EVALUATION

This section presents the evaluations of CPCA.

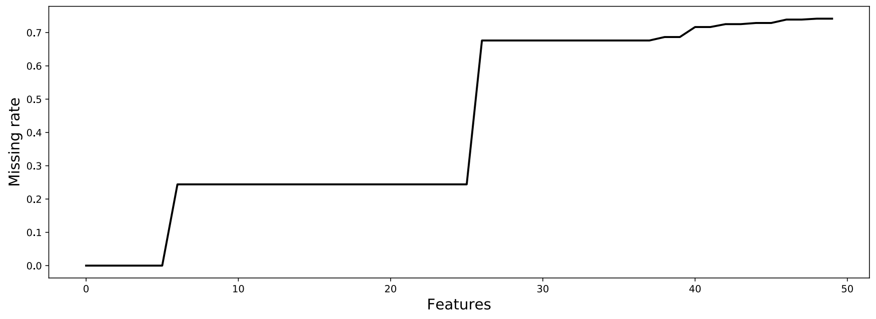


FIGURE 5. The missing rate of the dataset.

TABLE 1. The detail of the dataset.

Category	Number of Features	Feature Details
Weather (Specific time & 24-hour)	13	Temperature Hummidity Wind conditions Precipitation Air regime ...
POI	14	Category of POI Level of POI Location of POI District ...
Historical Order Status	29	The origin The destination Generation time of the order Payment of the order Price per kilometer ...

A. EXPERIMENT SETUP

We first introduce the experimental settings.

1) DATASETS

We use a dataset for taxi order prediction collected by a taxi-calling platform in China. The dataset contains taxi orders in Beijing, China during the period of June 1st, 2016 to June 5th, 2016, as well as various urban data as initial features for taxi order prediction. The initial features are obtained from different data sources and can be roughly divided into three categories: weather conditions, POIs and historical order status.

- Features about weather conditions: temperature, humidity, wind conditions, precipitation, air regime, *et al.* for a specific time and the around 24 hours.
- Features about POIs: name, category, level, location *et al.* of POIs. The category is also the name of the POIs, *e.g.* hospital, university and park. The level is used to describe the scale of POI.5
- Features about historical order status: origin/destination, start/end time, normal driving distance, estimated price, payment, discount *et al.*

Table 1 shows some example features. The semantics of these features are easy to understand through the schema. In total, the dataset has about 56 dimensions and 17,078,533 instances.

2) PREPROCESSING

Before performing dimension reduction on the dataset, we first preprocess the dataset to fill missing values. As shown in Fig. 5, there are many missing values in the dataset. The missing rates of most features are around 24% and 70%. Since we do not want to affect the distribution of these features and Eq.3 only uses the bias of these features, we fill these empty values with the mean of the other available values. Note that filling missing values is a well studied topic [19], [20] and is not the main focus of our paper.

3) IMPLEMENTATION

We submit 1,540 tasks on the AMT to collect answers about the independence between features from the crowd. Considering that our task does not require any professional knowledge,

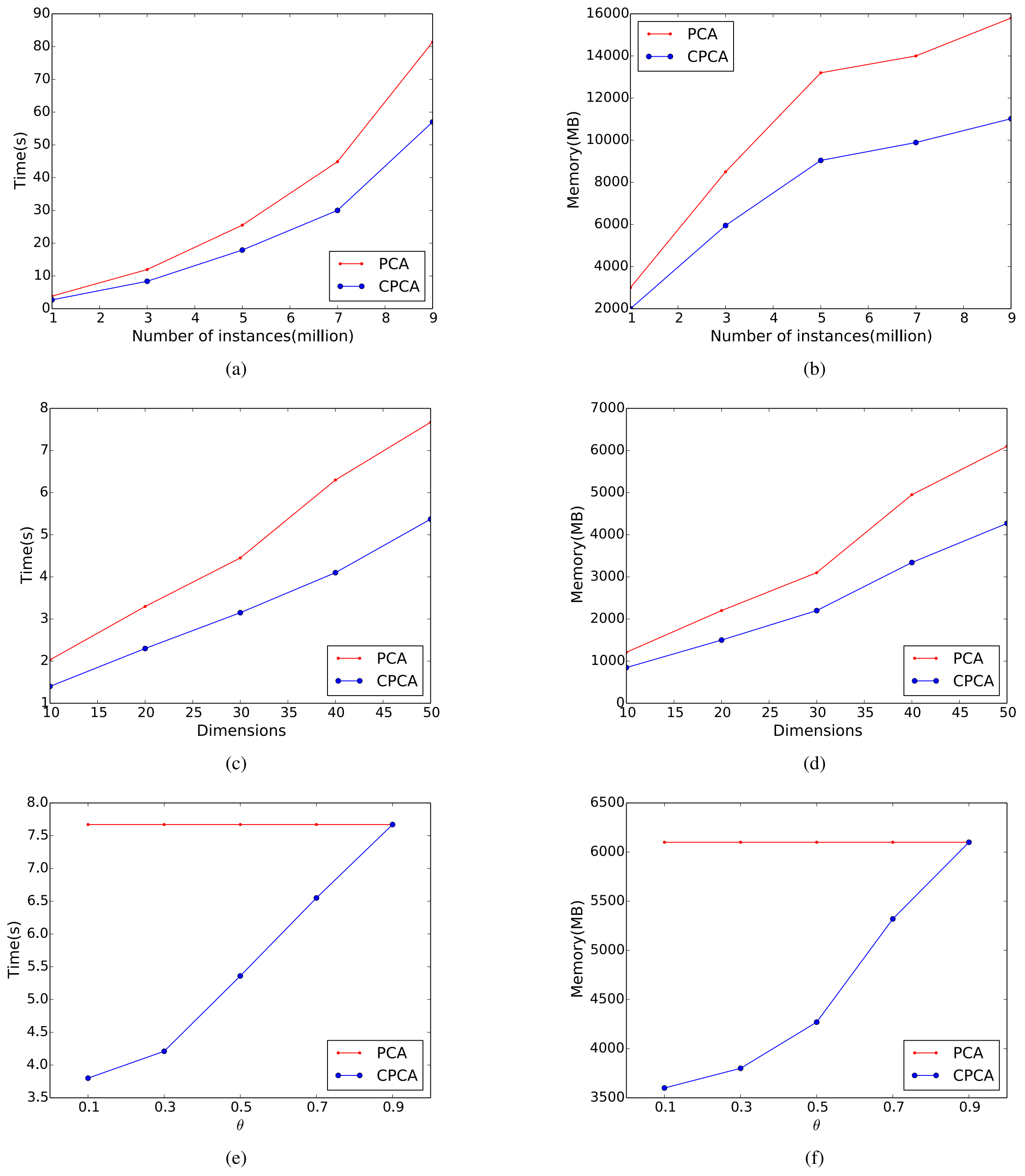


FIGURE 6. **Performance on the dataset. (a) Time of varying the number of instances. (b) Memory of varying the number of instances. (b) Time of varying dimensions. (c) Memory of varying dimensions. (d) Time of varying θ. (e) Memory of varying θ.**

we do not set constraints on the workers' skills or knowledge. The tasks are submitted for three days and in total about 1000 results are collected.

We implement the dimension reduction algorithms in python and conduct all the experiments on a computer with Intel(R) Core(TM) i7 3.4GHz CPU and 16GB main memory.

4) BASELINES AND METRICS

We compare CPCA with the ordinary PCA as the baseline. We mainly evaluate the time and storage of the two dimension reduction algorithms by projecting the original dataset into a 10-dimensional space.

B. PERFORMANCE

We first evaluate the impact of θ on the performance of CPCA. Fig. 6(e) and Fig. 6(f) show the running time and the memory cost by varying θ from 0.1 to 0.9. As is shown, when θ increases, both the running time and the memory cost of CPCA increase. This is because when θ increases, the features are grouped into fewer clusters, and CPCA needs more time and memory to compute the covariances among the features. When θ is so large that all features are grouped into a single cluster, CPCA takes nearly the same time and memory as PCA. In the rest of the experiments, we set θ to 0.5.

We then compare the running time and the memory cost of the two algorithms to process different number of instances. Fig. 6(a) and Fig. 6(b) show the result of varying the number of instances from 1 million to 9 million. In Fig. 6(a), the running time of both PCA and CPCA increases as the number of instances increases. CPCA run nearly 30% faster than PCA. In Fig. 6(b), the memory of both methods also increases. For PCA, it requires nearly 15GB memory when there are 9 million instances. It almost takes all the available memory and cannot run for a larger dataset in the machine with 16GB main memory. In contrast, CPCA takes about 10GB memory when there are 9 million instances and reduces about 37.5% of the memory cost. When further increasing the number of instances for CPCA, it could handle nearly 14 million instances.

Next we evaluate the running time and the memory cost of the two algorithms to process dataset of different number of dimensions. Fig. 6(c) and Fig. 6(d) show the result of varying the dimension of the dataset from 10 to 50. The results are obtained by processing two million instances. Fig. 6(c) shows that the running time of CPCA is less than that of PCA by 28%. In addition, as shown in Fig. 6(d), PCA spends up 30.3% to more memory than CPCA.

Afterwards, a linear classification model is trained on the processed datasets of both methods to predict the taxi orders. The two datasets both have ten dimensions, and the loss function of the linear model is the error of mean square. We train the model for the same number of iterations and use F1-measure to evaluate the effect of the dimensionality reduction methods. The F1-measure of PCA is around 0.78, and that of CPCA is about 0.69. Through the F1-measure of CPCA is lower than PCA, it achieves a large reduction on the computational cost.

SUMMARY OF RESULTS

From the experiments, we observe that CPCA outperforms PCA in terms of both the running time and memory cost. Although CPCA requires a few days to obtain the results from the crowd, it reduces the computational cost and allows the individual developers to run the dimension reduction methods on their own personal computers.

V. RELATED WORK

In this section, we review related work from two categories, dimension reduction and crowdsourcing.

A. DIMENSION REDUCTION

Dimensionality reduction is an important topic in machine learning and big data. Traditional linear dimensional reduction methods, such as PCA [17] and LDA [21], project the high-dimensional data into a lower-dimensional space. The following studies include the incremental and kernel-based methods [22], [23]. In recent years, there are also proposals utilizing various neural network for dimension reduction. For example, Krizhevsky *et al.* [24] use convolutional neural network (CNN) to extract feature sequences from the word images. Wu *et al.* [25] propose a deep neural network (DNN) for text recognition from the natural scene images. However, these methods still have huge computational cost, especially when computing the covariance matrices.

B. CROWDSOURCING

Crowdsourcing is a computing paradigm that aims to solve the hard-to-compute problem with the crowd [26]. Crowdsourcing has been applied to various applications. Yi *et al.* [11] mine users' preferences through the crowd tasks under the low-rank assumption. In [13], a data cleaning system KATARA is designed to identify and improve the data quality. Some other works apply crowdsourcing in applications such as annotating the training dataset for machine learning [27], [28] and data integration [29]. To support these applications, technical issues such as task allocation [30], [31], quality control [16], [32] and incentive mechanism design [33], [34] have also been extensively studied. To the best of our knowledge, there is no prior work to apply crowdsourcing in dimension reduction.

VI. CONCLUSION

In this paper, we propose a crowd-based dimension reduction framework called Crowd Principal Component Analysis (CPCA). The aim is to reduce the computation overhead of traditional principle component analysis leveraging collective intelligence so as to bring dimension reduction to individuals who have limited computation resources. Specifically, CPCA designs crowd tasks and asks crowd workers to estimate the independence among the features based on their semantics. After aggregating the answers from the crowd, CPCA eliminates the unnecessary computation to calculate the original covariance matrix and eigenvalues. We prove the correctness of CPCA and also analyze its complexity. Extensive experiments on the real dataset validate the effectiveness of our method.

REFERENCES

[1] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *J. Mach. Learn. Res.*, vol. 16, pp. 2859–2900, Dec. 2015.

[2] M. Nilashi, O. Ibrahim, and K. Bagherifard, "A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques," *Expert Syst. Appl.*, vol. 92, pp. 507–520, Feb. 2018.

[3] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Elect. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014.

[4] M. E. Tipping, "Sparse kernel principal component analysis," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Denver, CO, USA, 2000, pp. 633–639.

[5] X. Shi, Z. Guo, Z. Lai, Y. Yang, Z. Bao, and D. Zhang, "A framework of joint graph embedding and sparse regression for dimensionality reduction," *IEEE Trans. Image Process.*, vol. 24, no. 4, pp. 1341–1355, Apr. 2015.

[6] Z. Lai, W. K. Wong, Y. Xu, J. Yang, and D. Zhang, "Approximate orthogonal sparse embedding for dimensionality reduction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 723–735, Apr. 2016.

[7] C. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas, "Randomized dimensionality reduction for k-means clustering," *IEEE Trans. Inf. Theory*, vol. 61, no. 2, pp. 1045–1062, Feb. 2015.

[8] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu, "Dimensionality reduction for k-means clustering and low rank approximation," in *Proc. 47th Annu. ACM Symp. Theory Comput. (STOC)*, Portland, OR, USA, Jun. 2015, pp. 163–172.

[9] C. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, and D. P. Mandic, "Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions," *Found. Trends Mach. Learn.*, vol. 9, nos. 4–5, pp. 249–429, 2016.

[10] S. Wang, Z. Ding, and Y. Fu, "Feature selection guided auto-encoder," in *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, Feb. 2017, pp. 2725–2731.

[11] J. Yi, R. Jin, S. Jain, and A. Jain, "Inferring users' preferences from crowdsourced pairwise comparisons: A matrix completion approach," in *Proc. 1st AAAI Conf. Hum. Comput. Crowdsourcing (HCOMP)*, Palm Springs, CA, USA, Nov. 2013.

[12] S. Das *et al.*, "Falcon: Scaling up hands-off crowdsourced entity matching to build cloud services," in *Proc. ACM Int. Conf. Manage. Data (SIGMOD)*, Chicago, IL, USA, May 2017, pp. 1431–1446.

[13] X. Chu *et al.*, "Katara: A data cleaning system powered by knowledge bases and crowdsourcing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Melbourne, VIC, Australia, May/Jun. 2015, pp. 1247–1261.

[14] O. Elezaj, S. Yildirim, and E. Kalemi, "Data-driven machine learning approach for predicting missing values in large data sets: A comparison study," in *Proc. Int. Workshop Mach. Learn., Optim., Big Data*, Volterra, Italy, Sep. 2017, pp. 268–285.

[15] Y. Tong *et al.*, "The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1653–1662.

[16] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng, "Truth inference in crowdsourcing: Is the problem solved?" *Proc. VLDB Endowment*, vol. 10, no. 5, pp. 541–552, 2017.

[17] I. Jolliffe, "Principal component analysis," in *International Encyclopedia of Statistical Science*. New York, NY, USA: Springer-Verlag, 2011, pp. 1094–1096.

[18] Y. Zeng, Y. Tong, L. Chen, and Z. Zhou, "Latency-oriented task completion via spatial crowdsourcing," in *Proc. 34th IEEE Int. Conf. Data Eng. (ICDE)*, Paris, France, Apr. 2018, pp. 317–328.

[19] M. Kolar and E. P. Xing, "Consistent covariance selection from data with missing values," in *Proc. 29th Int. Conf. Mach. Learn. (ICML)*, Edinburgh, U.K., Jun./Jul. 2012, pp. 551–558.

[20] X. Yi, Y. Zheng, J. Zhang, and T. Li, "ST-MVL: Filling missing values in Geo-sensory time series data," in *Proc. 25th Int. Joint Conf. Artif. Intell. (IJCAI)*, New York, NY, USA, Jul. 2016, pp. 2704–2710.

[21] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Müllers, "Fisher discriminant analysis with kernels," in *Proc. 9th Neural Netw. Signal Process. Soc. Workshop.*, Aug. 1999, pp. 41–48.

[22] A. Balsubramani, S. Dasgupta, and Y. Freund, "The fast convergence of incremental PCA," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2013, pp. 3174–3182.

[23] L. Zhang, T. Yang, J. Yi, R. Jin, and Z.-H. Zhou, "Stochastic optimization for kernel PCA," in *Proc. 13th AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, Feb. 2016, pp. 2315–2322.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 26th Annu. Conf. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2012, pp. 1097–1105.

[25] R. Wu, S. Yang, D. Leng, Z. Luo, and Y. Wang, "Random projected convolutional feature for scene text recognition," in *Proc. 15th Int. Conf. Frontiers Handwriting Recognit. (ICFHR)*, Shenzhen, China, Oct. 2016, pp. 132–137.

[26] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," in *Proc. 33rd IEEE Int. Conf. Data Eng. (ICDE)*, San Diego, CA, USA, Apr. 2017, pp. 39–40.

[27] D. Haas, J. Wang, E. Wu, and M. J. Franklin, "Clamshell: Speeding up crowds for low-latency data labeling," *Proc. VLDB Endowment*, vol. 9, no. 4, pp. 372–383, 2015.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Miami, FL, USA, Jun. 2009, pp. 248–255.

[29] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux, "Large-scale linked data integration using probabilistic reasoning and crowdsourcing," *J. Very Large Data Bases*, vol. 22, no. 5, pp. 665–687, 2013.

[30] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *Proc. 32nd IEEE Int. Conf. Data Eng. (ICDE)*, Helsinki, Finland, May 2016, pp. 49–60.

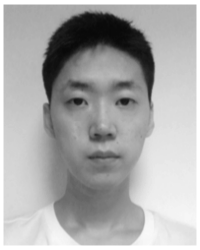
[31] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, "Online minimum matching in real-time spatial data: Experiments and analysis," *Proc. VLDB Endowment*, vol. 9, no. 12, pp. 1053–1064, 2016.

[32] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng, "QASCA: A quality-aware task assignment system for crowdsourcing applications," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Melbourne, VIC, Australia, May/Jun. 2015, pp. 1031–1046.

[33] Y. Tong, L. Wang, Z. Zhou, L. Chen, B. Du, and J. Ye, "Dynamic pricing in spatial crowdsourcing: A matching-based approach," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, Houston, TX, USA, Jun. 2018, pp. 773–788.

[34] Y. Tong, L. Chen, Z. Zhou, H. V. Jagadish, L. Shou, and W. Lv, "SLADE: A smart large-scale task decomposer in crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 8, pp. 1588–1601, Aug. 2018.

YUANYUAN ZHANG received the master's degree from the School of Software Engineering, Beihang University, Beijing, China, in 2014. She is currently a Senior Engineer with the China Academy of Telecommunication Technology, Beijing. Her major research interests include crowdsourcing, databases, and data mining.



DAWEI GAO received the bachelor's degree from the School of Computer Science and Engineering, Beihang University, Beijing, China, in 2016, where he is currently pursuing the Ph.D. degree. His major research interests include crowdsourcing, databases, data mining, and machine learning.

JIE LUO received the Ph.D. degree from Beihang University, Beijing, China. He is currently a Lecturer with the School of Computer Science and Engineering, Beihang University. His research interests include mathematical logic, knowledge reasoning, algorithms, crowd intelligence, and formal methods.

KE XU received the B.E., M.E., and Ph.D. degrees from Beihang University, China, in 1993, 1996, and 2000, respectively. He is currently a Professor with Beihang University. His research interests include algorithm and complexity, data mining, crowdsourcing, and complex networks.