Scalable Algorithms for CQA Post Voting Prediction

Yuan Yao, Hanghang Tong, Feng Xu, and Jian Lu

Abstract—Community Question Answering (CQA) sites, such as Stack Overflow and Yahoo! Answers, have become very popular in recent years. These sites contain rich crowdsourcing knowledge contributed by the site users in the form of questions and answers, and these questions and answers can satisfy the information needs of more users. In this article, we aim at predicting the voting scores of questions/answers shortly after they are posted in the CQA sites. To accomplish this task, we identify three key aspects that matter with the voting of a post, i.e., the non-linear relationships between features and output, the question and answer coupling, and the dynamic fashion of data arrivals. A family of algorithms are proposed to model the above three key aspects. Some approximations and extensions are also proposed to scale up the computation. We analyze the proposed algorithms in terms of optimality, correctness, and complexity. Extensive experimental evaluations conducted on two real data sets demonstrate the effectiveness and efficiency of our algorithms.

Index Terms—Question answering, voting prediction, non-linearity, coupling, dynamics

1 Introduction

Community Question Answering (CQA) sites, such as Stack Overflow[1] and Yahoo! Answers[2], have become very popular in recent years. These sites contain rich crowdsourcing knowledge contributed by the site users in the form of questions and answers, and these questions and answers can potentially satisfy the information needs of more users. For example, millions of programmers ask and answer questions on Stack Overflow, and even more users now use Stack Overflow to seek solutions for their programming problems [1].

In this article, we focus on the voting score prediction of questions/answers shortly after they are posted in the CQA sites. Such a task is essential for the prosperity and sustainability of the CQA ecosystem, and it may benefit all types of users, including the information producers and consumers [2]. For example, detecting potentially high-score answers can benefit the questioners as well as the people who have similar questions; it would also be helpful to identify high-score questions in the early stage and route them to expert answerers.

Generally speaking, there are three key aspects that matter with the voting prediction of a post, namely, (1) the *non-linearity* between features and output, (2) the *coupling* between questions and answers, and (3) the *dynamics* (of training data sets). First, both the contextual features (e.g., the reputation of the user who issues the question, etc.) and the content of the post (e.g., keywords, etc.) might affect its voting score, and the effect of each feature might be beyond the simple linear-relationship. Second, intuitively (which was also confirmed in our previous work [3]), the voting of a question might be correlated with that of its associated answers. Yet, the questions and answers may reside in different feature spaces. Third, CQA sites usually offer a large size of training data set, and the data may arrive in a dynamic (stream-like) way for the mining algorithms.

Due to the above three aspects, it is not an easy task to comprehensively and efficiently predict the voting scores of question/answer posts. The challenges are as follows. First, while each of the above three aspects might affect the voting scores of question/answer posts, they require different treatments in the data mining algorithms, making any off-the-shelf data mining algorithm sub-optimal for this problem. Second, each of the above three aspects will add the extra complexity into the mining process. For example, while many machine learning algorithms (e.g., kernel regression, support vector regression, etc.) are able to capture the non-linearity aspect, they typically require at least quadratic complexity in both time and space. Moreover, when the new training examples arrive in a stream, ever-growing fashion, even a linear algorithm might be too expensive. How can we build a *comprehensive* model to capture all the above three aspects to maximally boost the prediction accuracy? How can we make our prediction algorithms *scalable* to millions of CQA posts and *adaptive* to the newly arrived training examples over time? These are the main challenges that we aim to address in this article.

To address the challenges, we propose a family of algorithms for predicting the voting scores of question/answer posts. Our algorithms enjoy three key advantages. First, they are *comprehensive* in the sense that our model naturally captures all the above three key aspects (i.e., non-linearity, coupling, and dynamics) that matter with the voting score of a post. Second, they are *flexible* and *general*,

- *Yuan Yao, Feng Xu, and Jian Lu are with the State Key Laboratory for Novel Software Technology, Nanjing University, China. E-mail: {y.yao, xf, lj}@nju.edu.cn.*
- *Hanghang Tong is with Arizona State University, USA. E-mail: hanghang.tong@asu.edu.*

1. http://stackoverflow.com/
2. http://answers.yahoo.com/

being able to fade the effects of old examples, select a subset of features/examples, and handle some special cases (i.e., when one or two aspects are prominent). Third, they are *scalable* and *adaptive* to the newly arrived examples. For example, one of our algorithms (LIP-KIMAA) has a sub-linear complexity in both time and space. On the Stack Overflow data set with more than 3 million posts, this LIP-KIMAA algorithm can build and update our model in seconds, while straightforward approaches would take hours.

In summary, the main contributions of this paper include:

- *A family of novel algorithms* for the prediction of the voting scores of questions/answers in CQA sites. The algorithms can handle the three aspects of non-linearity, coupling, and dynamics. They can also fade the effects of old posts and learn a sparse set of features/examples. *Proofs and analysis* show the optimality, correctness, and computational efficiency, as well as the intrinsic relationships among different algorithms.
- *Extensive empirical evaluations*, demonstrating the effectiveness and efficiency of our algorithms. For example, compared with alternative choices (e.g., KRR [4]), one of our proposed algorithms (LIP-KIMAA) (1) leads up to 35.8% improvement in prediction accuracy; (2) and is up to 390x faster while enjoying *sub-linear* scalability.

The rest of the paper is organized as follows. Section 2 describes the problem definition. Section 3 presents the proposed algorithms. Section 4 further presents some variants and extensions. Section 5 presents the experimental results. Section 6 reviews related work, and Section 7 concludes the paper.

2 Problem Statement

In this section, we first define the target problem, and then present its solution space to illustrate the relationships between our proposed algorithms and the existing work.

2.1 Problem Definitions

Table 1 lists the main symbols we use throughout the paper. For convenience, we use bold capital letters for existing matrices/vectors at time t, and bold lower case letters for newly arrived matrices/vectors at time $t+1$. We use superscript (i.e., q or a) to distinguish questions and answers, and use subscript (i.e., t, t+1, etc.) to indicate time. For example, we use $\mathbf{F}_t^q$ to denote the feature matrix for questions at time t, and $\mathbf{f}_{t+1}^q$ to denote the feature matrix of newly arrived questions at time t+1. Each row of $\mathbf{F}_t^q$ and $\mathbf{f}_{t+1}^q$ contains the feature vector for the corresponding question. Similarly, we use $\mathbf{Y}_t^q$ to denote the vector of voting scores at time t, and $\mathbf{y}_{t+1}^q$ to denote the vector of voting scores from new questions at time $t+1$. Following conventions, we use calligraphic letter $\mathcal{K}_t^q$ and $\mathcal{K}_t^a$ to denote the kernel matrix for questions and answers at time t. We will omit the subscript when the meaning of matrices/vectors is clear in the context. We use the row-normalized $n_q \times n_a$ matrix $\mathbf{M}_t$ to denote the association between questions and answers at time t, where non-zero element $\mathbf{M}_t(i,j)$ indicates that the j^{th} answer belongs to the i^{th} question. Thus, the matrix $\mathbf{M}_t$ is sparse since it contains only n^a non-zero elements. We also use $\mathbf{F}_t^q(i,:)$ to represent the i^{th} row of matrix $\mathbf{F}_t^q$, $\mathbf{Y}_t^q(i)$ to represent the i^{th} element of vector $\mathbf{Y}_t^q$, and $(\mathbf{F}_t^q)'$ to represent the transpose of $\mathbf{F}_t^q$.

TABLE 1
Symbols.

Symbol	Definition and Description
$\mathbf{F}_t^q, \mathbf{F}_t^a$	the features for existing questions/answers at time t
$\mathbf{f}_{t+1}^q, \mathbf{f}_{t+1}^a$	the features of new questions/answers at time $t+1$
$\mathcal{K}_t^q, \mathcal{K}_t^a$	the kernel matrix of $\mathbf{F}_t^q$ and $\mathbf{F}_t^a$
$\mathbf{k}_{t+1}^q, \mathbf{h}_{t+1}^q$	the kernel matrix of new questions at time $t+1$
$\mathbf{k}_{t+1}^a, \mathbf{h}_{t+1}^a$	the kernel matrix of new answers at time $t+1$
$\mathbf{U}_t^q, \Lambda_t^q$	the low-rank matrices to approximate $\mathcal{K}_t^q$
$\mathbf{U}_t^a, \Lambda_t^a$	the low-rank matrices to approximate $\mathcal{K}_t^a$
$\mathbf{M}_t$	the row-normalized association matrix between existing questions and answers at time t
$\mathbf{m}_{t+1}$	the row-normalized association matrix between new questions and answers at time $t+1$
$\mathbf{Y}_t^q, \mathbf{Y}_t^a$	the voting score for existing questions/answers at time t
$\mathbf{y}_{t+1}^q, \mathbf{y}_{t+1}^a$	the voting score for new questions/answers at time $t+1$
n^q, n^a	the number of existing questions/answers at time t
i^q, i^a	the number of new questions/answers at time $t+1$
d	the feature dimension
r	the rank of $\mathbf{U}_t^q$, Λ_t^q, $\mathbf{U}_t^a$, and Λ_t^a
th	the threshold for the filtering step

In this work, our goal is to predict the voting scores (i.e., the number of upvotes minus the number of downvotes) of questions and answers shortly after they are created. Based on the above notations, we name the target problem as Long-term Impact Prediction (LIP) problem and define its static form as

Problem 1: Static LIP Problem

Given: the question/answer feature matrix $\mathbf{F}^q/\mathbf{F}^a$, the question/answer voting vector $\mathbf{Y}^q/\mathbf{Y}^a$, and the association matrix $\mathbf{M}$;

Output: the voting of new questions and their answers.

In real CQA sites where questions and answers continuously arrive, we need to update the model to keep it up-to-date. To this end, we define the following dynamic form of the problem

Problem 2: Dynamic LIP Problem

Given: the question/answer feature matrix $\mathbf{F}_t^q/\mathbf{F}_t^a$ and the newly arrived question/answer feature matrix $\mathbf{f}_{t+1}^q/\mathbf{f}_{t+1}^a$, the question/answer voting vector $\mathbf{Y}_t^q/\mathbf{Y}_t^a$ and the newly arrived question/answer voting vector $\mathbf{y}_{t+1}^q/\mathbf{y}_{t+1}^a$, as well as the association matrix $\mathbf{M}_t$ and $\mathbf{m}_{t+1}$;

Output: the voting of new questions and their answers.

2.2 Solution Space

Solution Space. Let us first define the solution space of the LIP problem, which is represented by a genealogy graph in Fig. 1. In this paper, we consider three key aspects that matter with the prediction performance, including (a) whether the predication models are linear or non-linear;

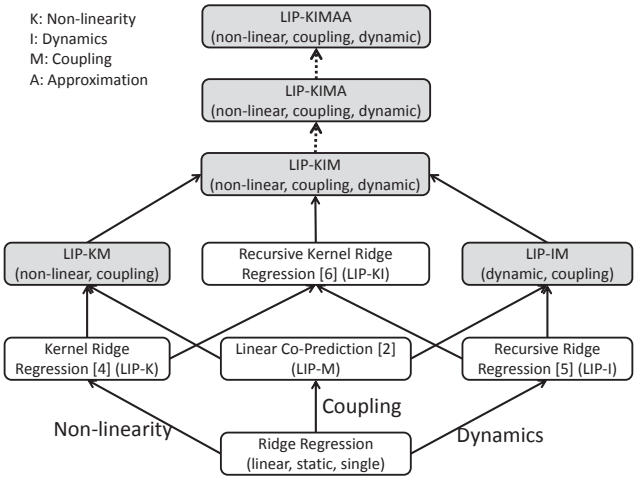


Fig. 1. The solution space of LIP problem. Shaded boxes are proposed algorithms; and white boxes are existing work.

(b) whether we treat the prediction of the questions and the answers separately (single) or jointly (coupling); and (c) whether the prediction is static or dynamic. Based on these three aspects, we could have different variants of the prediction algorithms for the LIP problem, whose intrinsic relationships are also summarized in Fig. 1. In the figure, we use the letter K, M, and I to denote non-linearity, coupling, and dynamics, respectively. For example, LIP-KIM means that our model is non-linear and dynamic, and it jointly predicts the voting of questions and answers; LIP-K means that our prediction model is non-linear and static, and it treats questions and answers separately, etc.

In Fig. 1, each upward solid arrow makes the model more comprehensive by modeling more aspects in the prediction algorithms, and each dashed arrow makes the algorithms more scalable. For example, starting with the ridge regression (RR) algorithm in the bottom layer, we have the kernel ridge regression (KRR) [4] by incorporating non-linearity, and we have the recursive ridge regression [5] by incorporating dynamics. If we incorporate both non-linearity and dynamics, we have the recursive kernel ridge regression algorithm (RKRR) [6] in the third layer.

Preliminaries. In Fig. 1, we use shaded boxes to indicate the algorithms proposed in this paper, and while boxes to indicate existing work. Before presenting the proposed algorithms in the next section, we first briefly review some existing work (i.e., white boxes), which serves as the building blocks of our proposed algorithms.

(A) Linear Co-Prediction. In our tech report [3], we proposed a regularized optimization formulation to jointly predict the voting score of questions and answers

$$\begin{aligned}\min_{\boldsymbol{\alpha}^q,\boldsymbol{\alpha}^a} \quad & \textstyle\sum_{i=1}^{n^q}(\mathbf{F}^q(i,:)\boldsymbol{\alpha}^q - \mathbf{Y}^q(i))^2 + \sum_{i=1}^{n^a}(\mathbf{F}^a(i,:)\boldsymbol{\alpha}^a - \mathbf{Y}^a(i))^2 \\ & +\theta\textstyle\sum_{i=1}^{n^q}(\mathbf{F}^q(i,:)\boldsymbol{\alpha}^q - \mathbf{M}(i,:)\mathbf{F}^a\boldsymbol{\alpha}^a)^2 + \lambda(\|\boldsymbol{\alpha}^q\|_2^2 + \|\boldsymbol{\alpha}^a\|_2^2) \quad (1)\end{aligned}$$

where parameters λ and θ are used to control regularization and the importance of the coupling between questions and answers, respectively.

(B) Kernel Ridge Regression. In order to capture the non-linearity between the features and output, a natural choice is to use kernelized methods. Taking question voting prediction as an example, the so-called kernel ridge regression [4] aims to estimate a coefficient $\boldsymbol{\beta}^q$ as follows

$$\min_{\boldsymbol{\beta}^q} \quad \textstyle\sum_{i=1}^{n^q}(\mathcal{K}^q(i,:)\boldsymbol{\beta}^q - \mathbf{Y}^q(i))^2 + \lambda(\boldsymbol{\beta}^q)'\mathcal{K}^q\boldsymbol{\beta}^q \quad (2)$$

where $\mathcal{K}^q$ is the kernel matrix of $\mathbf{F}^q$.

3 The Proposed Algorithms

In this section, we propose our solutions for the LIP problem. We start with presenting two algorithms for Problem 1 (subsection 3.1) and Problem 2 (subsection 3.2), respectively; and then address the computational challenges (subsection 3.3-3.4).

3.1 LIP-KM Algorithm for Problem 1

Here, we address the static LIP problem (Problem 1). We propose an algorithm (LIP-KM) to capture both the non-linearity and the coupling aspects.

For the non-linear aspect, a natural choice is to kernelize a linear prediction model (e.g., linear ridge regression). Recall that kernel method aims to produce non-linear versions of linear learning algorithms by mapping the data points into a high-dimensional Hilbert space $\mathcal{H}$ with a non-linear function ϕ [7]. The key idea behind kernel methods is to use the kernel functions to replace the inner-product operations in the high-dimensional Hilbert space $\mathcal{H}$, and such replacement can be ensured by Mercer's Condition [8]. In other words, for two data points $\mathbf{F}(i,:)$ and $\mathbf{F}(j,:)$, the inner product of $\phi(\mathbf{F}(i,:))$ and $\phi(\mathbf{F}(j,:))$ in the Hilbert space $\mathcal{H}$ can be directly computed by a Mercer kernel $\kappa(\mathbf{F}(i,:),\mathbf{F}(j,:))$

$$\begin{aligned}\kappa(\mathbf{F}(i,:),\mathbf{F}(j,:)) &= <\phi(\mathbf{F}(i,:)),\phi(\mathbf{F}(j,:))> \\ &= \phi(\mathbf{F}(i,:))\phi(\mathbf{F}(j,:))' \quad (3)\end{aligned}$$

where $<\cdot,\cdot>$ indicates the inner product in $\mathcal{H}$. As we can see from Eq. (3), we can derive the non-linear models without any explicit knowledge of either ϕ or $\mathcal{H}$. Common kernel functions include Gaussian kernel, polynomial kernel, cosine kernel, etc.

For the coupling aspect, LIP-KM first transfers the features and then imposes a so-called *voting consistency* on the prediction space by requiring the predicted voting of a question to be close to that of its answer (see our tech report [3] for the detailed explanations about its rationality).

Putting the non-linearity and coupling aspects together, we have the following optimization formulation for Problem 1

$$\begin{aligned}\min_{\boldsymbol{\beta}^q,\boldsymbol{\beta}^a} \quad & \textstyle\sum_{i=1}^{n^q}(\mathcal{K}^q(i,:)\boldsymbol{\beta}^q - \mathbf{Y}^q(i))^2 + \sum_{i=1}^{n^a}(\mathcal{K}^a(i,:)\boldsymbol{\beta}^a - \mathbf{Y}^a(i))^2 \\ & +\theta\textstyle\sum_{i=1}^{n^q}(\mathcal{K}^q(i,:)\boldsymbol{\beta}^q - \mathbf{M}(i,:)\mathcal{K}^a\boldsymbol{\beta}^a)^2 \\ & +\lambda((\boldsymbol{\beta}^q)'\mathcal{K}^q\boldsymbol{\beta}^q + (\boldsymbol{\beta}^a)'\mathcal{K}^a\boldsymbol{\beta}^a) \quad (4)\end{aligned}$$

where θ is a weight parameter to control the importance of coupling, λ is a regularization parameter, and $\mathcal{K}^q$ and $\mathcal{K}^a$ are the kernel matrices of $\mathbf{F}^q$ and $\mathbf{F}^a$, respectively. $\mathcal{K}^q$ and $\mathcal{K}^a$ can computed as $\mathcal{K}^q(i,j) = \kappa(\mathbf{F}^q(i,:),\mathbf{F}^q(j,:))$, and $\mathcal{K}^a(i,j) = \kappa(\mathbf{F}^a(i,:),\mathbf{F}^a(j,:))$.

Eq. (4) can be solved by the closed-form solution

$$\begin{aligned}\boldsymbol{\beta} &= \arg\min_{\boldsymbol{\beta}^q,\boldsymbol{\beta}^a} \|\mathcal{K}^q\boldsymbol{\beta}^q - \mathbf{Y}^q\|_2^2 + \|\mathcal{K}^a\boldsymbol{\beta}^a - \mathbf{Y}^a\|_2^2 \\ &\quad +\theta\|\mathcal{K}^q\boldsymbol{\beta}^q - \mathbf{M}\mathcal{K}^a\boldsymbol{\beta}^a\|_2^2 + \lambda((\boldsymbol{\beta}^q)'\mathcal{K}^q\boldsymbol{\beta}^q + (\boldsymbol{\beta}^a)'\mathcal{K}^a\boldsymbol{\beta}^a) \\ &= \begin{bmatrix}(\theta+1)\mathcal{K}^q + \lambda\mathbf{I} & -\theta\mathbf{M}\mathcal{K}^a \\ -\theta\mathbf{M}'\mathcal{K}^q & \mathcal{K}^a + \theta\mathbf{M}'\mathbf{M}\mathcal{K}^a + \lambda\mathbf{I}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{Y}^q \\ \mathbf{Y}^a\end{bmatrix} \end{aligned} \quad (5)$$

where $\boldsymbol{\beta} = [\boldsymbol{\beta}^q; \boldsymbol{\beta}^a]$. Once the coefficient vectors $\boldsymbol{\beta}^q$ and $\boldsymbol{\beta}^a$ are inferred from the above equation, the voting of questions/answers can then be predicted as

$$\hat{\mathbf{Y}}^q_{test} = \mathcal{K}^q_{test}\boldsymbol{\beta}^q, \quad \hat{\mathbf{Y}}^a_{test} = \mathcal{K}^a_{test}\boldsymbol{\beta}^a \quad (6)$$

where the kernel matrices on the test set $\mathbf{F}^q_{test}$ and $\mathbf{F}^a_{test}$ can be computed as $\mathcal{K}^q_{test}(i,j) = \kappa(\mathbf{F}^q_{test}(i,:), \mathbf{F}^q(j,:))$, $\mathcal{K}^a_{test}(i,j) = \kappa(\mathbf{F}^a_{test}(i,:), \mathbf{F}^a(j,:))$.

Algorithm Analysis. Let us analyze the effectiveness and efficiency of the LIP-KM algorithm (i.e., Eq. (5)). We first summarize the optimality of LIP-KM in the following lemma, which states that LIP-KM finds (at least) a local minimum for the static LIP problem.

Lemma 1: **Optimality of LIP-KM**. Eq. (5) finds a local minimum for Eq. (4).

Proof: See the supplementary material. □

Next, we summarize the time complexity and space complexity of LIP-KM in the following lemma, which basically states that LIP-KM requires $O((n^q + n^a)^3)$ time and $O((n^q + n^a)^2)$ space.

Lemma 2: **Complexity of LIP-KM**. The time complexity of Eq. (5) is $O((n^q + n^a)^3 + (n^q)^2 d + (n^a)^2 d)$; the space complexity of Eq. (5) is $O((n^q + n^a)^2 + (n^q + n^a)d)$.

Proof: See the supplementary material. □

3.2 LIP-KIM Algorithm for Problem 2

Here, we address the dynamic LIP problem (Problem 2), which can be formulated as

$$\begin{aligned}\min_{\boldsymbol{\beta}^q_{t+1},\boldsymbol{\beta}^a_{t+1}} \quad & \|\mathcal{K}^q_{t+1}\boldsymbol{\beta}^q_{t+1} - \mathbf{Y}^q_{t+1}\|_2^2 + \|\mathcal{K}^a_{t+1}\boldsymbol{\beta}^a_{t+1} - \mathbf{Y}^a_{t+1}\|_2^2 \\ & +\theta\|\mathcal{K}^q_{t+1}\boldsymbol{\beta}^q_{t+1} - \mathbf{M}_{t+1}\mathcal{K}^a_{t+1}\boldsymbol{\beta}^a_{t+1}\|_2^2 \\ & +\lambda((\boldsymbol{\beta}^q_{t+1})'\mathcal{K}^q_{t+1}\boldsymbol{\beta}^q_{t+1} + (\boldsymbol{\beta}^a_{t+1})'\mathcal{K}^a_{t+1}\boldsymbol{\beta}^a_{t+1}) \end{aligned} \quad (7)$$

where the new kernel matrices $\mathcal{K}^q_{t+1}$ and $\mathcal{K}^a_{t+1}$ are computed as

$$\mathcal{K}^q_{t+1} = \begin{bmatrix}\mathcal{K}^q_t & (\mathbf{k}^q_{t+1})' \\ \mathbf{k}^q_{t+1} & \mathbf{h}^q_{t+1}\end{bmatrix}, \quad \mathcal{K}^a_{t+1} = \begin{bmatrix}\mathcal{K}^a_t & (\mathbf{k}^a_{t+1})' \\ \mathbf{k}^a_{t+1} & \mathbf{h}^a_{t+1}\end{bmatrix} \quad (8)$$

The kernel matrices involving the newly arrived examples can be computed as $\mathbf{k}^q_{t+1}(i,j) = \kappa(\mathbf{f}^q_{t+1}(i,:), \mathbf{F}^q_t(j,:))$, $\mathbf{h}^q_{t+1}(i,j) = \kappa(\mathbf{f}^q_{t+1}(i,:), \mathbf{f}^q_{t+1}(j,:))$, $\mathbf{k}^a_{t+1}(i,j) = \kappa(\mathbf{f}^a_{t+1}(i,:), \mathbf{F}^a_t(j,:))$, and $\mathbf{h}^a_{t+1}(i,j) = \kappa(\mathbf{f}^a_{t+1}(i,:), \mathbf{f}^a_{t+1}(j,:))$.

Next, we present the LIP-KIM algorithm for Eq. (7). The basic idea of LIP-KIM is to incorporate the dynamic aspect into LIP-KM, and therefore it is adaptive to newly arrived training examples. In other words, our goal is to incrementally update the model in Eq. (5) from time t to time $t+1$ with some intermediate results (e.g., $\boldsymbol{\beta}^q_t$ and $\boldsymbol{\beta}^a_t$) at time t.

To simplify the algorithm description, let us introduce the following matrices

$$\mathbf{S}_t = \begin{bmatrix}(\theta+1)\mathcal{K}^q_t + \lambda\mathbf{I} & -\theta\mathbf{M}_t\mathcal{K}^a_t \\ -\theta\mathbf{M}'_t\mathcal{K}^q_t & \mathcal{K}^a_t + \theta\mathbf{M}'_t\mathbf{M}_t\mathcal{K}^a_t + \lambda\mathbf{I}\end{bmatrix} \quad (9)$$

Algorithm 1 The LIP-KIM Algorithm.

Input: $\mathbf{S}_t^{-1}$, $\boldsymbol{\beta}_t$, $\mathbf{F}^q_t$, $\mathbf{f}^q_{t+1}$, $\mathbf{F}^a_t$, $\mathbf{f}^a_{t+1}$, $\mathbf{y}^q_{t+1}$, $\mathbf{y}^a_{t+1}$, $\mathbf{M}_t$ and $\mathbf{m}_{t+1}$
Output: $\boldsymbol{\beta}_{t+1}$, $\mathbf{S}_{t+1}^{-1}$

1: compute the new kernels $\mathbf{k}^q_{t+1}$, $\mathbf{h}^q_{t+1}$, $\mathbf{k}^a_{t+1}$ and $\mathbf{h}^a_{t+1}$ in Eq. (8);
2: compute $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$, $\mathbf{D}$, and $\mathbf{E}_1$ as Eq. (10) - (14);
3: update $\mathbf{S}_{t+1}^{-1}$ as $\mathbf{E}_1\begin{bmatrix}\mathbf{S}_t^{-1} + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}\mathbf{S}_2\mathbf{S}_t^{-1} & -\mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D} \\ -\mathbf{S}_3^{-1}\mathbf{S}_2\mathbf{S}_t^{-1}(\mathbf{I} + \mathbf{S}_1\mathbf{D}\mathbf{S}_2\mathbf{S}_t^{-1}) & \mathbf{D}\end{bmatrix}\mathbf{E}_1$;
4: update $\boldsymbol{\beta}_{t+1}$ as $\mathbf{E}_1\begin{bmatrix}\boldsymbol{\beta}_t + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}(\mathbf{S}_2\boldsymbol{\beta}_t - [\mathbf{y}^q_{t+1}; \mathbf{y}^a_{t+1}]) \\ -\mathbf{S}_3^{-1}\mathbf{S}_2(\boldsymbol{\beta}_t + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}\mathbf{S}_2\boldsymbol{\beta}_t) + \mathbf{D}[\mathbf{y}^q_{t+1}; \mathbf{y}^a_{t+1}]\end{bmatrix}$;
5: **return** $\boldsymbol{\beta}_{t+1}$, $\mathbf{S}_{t+1}^{-1}$;

$$\mathbf{S}_1 = \begin{bmatrix}(\theta+1)(\mathbf{k}^q_{t+1})' & -\theta\mathbf{M}_t(\mathbf{k}^a_{t+1})' \\ -\theta\mathbf{M}'_t(\mathbf{k}^q_{t+1})' & (\mathbf{k}^a_{t+1})' + \theta\mathbf{M}'_t\mathbf{M}_t(\mathbf{k}^a_{t+1})'\end{bmatrix} \quad (10)$$

$$\mathbf{S}_2 = \begin{bmatrix}(\theta+1)\mathbf{k}^q_{t+1} & -\theta\mathbf{m}_{t+1}\mathbf{k}^a_{t+1} \\ -\theta\mathbf{m}'_{t+1}\mathbf{k}^q_{t+1} & \mathbf{k}^a_{t+1} + \theta\mathbf{m}'_{t+1}\mathbf{m}_{t+1}\mathbf{k}^a_{t+1}\end{bmatrix} \quad (11)$$

$$\mathbf{S}_3 = \begin{bmatrix}(\theta+1)\mathbf{h}^q_{t+1} + \lambda\mathbf{I} & -\theta\mathbf{m}_{t+1}\mathbf{h}^a_{t+1} \\ -\theta\mathbf{m}'_{t+1}\mathbf{h}^q_{t+1} & \mathbf{h}^a_{t+1} + \theta\mathbf{m}'_{t+1}\mathbf{m}_{t+1}\mathbf{h}^a_{t+1} + \lambda\mathbf{I}\end{bmatrix} \quad (12)$$

$$\mathbf{D} = (\mathbf{S}_3 - \mathbf{S}_2\mathbf{S}_t^{-1}\mathbf{S}_1)^{-1} \quad (13)$$

With these extra notations, we present our LIP-KIM algorithm for solving Problem 2 in Alg. 1. As we can see from the algorithm, we re-use $\mathbf{S}_t^{-1}$ and $\boldsymbol{\beta}_t$ from previous computations. Therefore, we also need to update $\mathbf{S}_{t+1}^{-1}$ and $\boldsymbol{\beta}_{t+1}$ for future iterations. After we compute the new kernels as well as the matrices (i.e., $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$ and $\mathbf{D}$) that are based on the new kernels, we can update the model in Steps 3-4. In these two steps, $\mathbf{E}_1$ stands for a permutation matrix to exchange the corresponding rows/columns

$$\mathbf{E}_1 = \begin{bmatrix}\mathbf{I}_{n^q\times n^q} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{i^q\times i^q} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n^a\times n^a} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{i^a\times i^a}\end{bmatrix} \quad (14)$$

where i^q and i^a denote the number of questions and answers arrived at time $t+1$, respectively.

Algorithm Analysis. The correctness of LIP-KIM is summarized in the following theorem, which states that LIP-KIM can find the same coefficients (i.e., $\boldsymbol{\beta}_{t+1}$) as if we apply the LIP-KM algorithm whenever we have new training examples.

Theorem 1: **Correctness of LIP-KIM**. Let $\boldsymbol{\beta}^*_{t+1}$ be the output of Eq. (5) at time $t+1$, and $\boldsymbol{\beta}_{t+1}$ be the output of Alg. 1 updated from time t to $t+1$, we have that $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}^*_{t+1}$.

Proof: See the supplementary material. □

The time complexity and space complexity of our LIP-KIM are summarized in the following lemma.

Lemma 3: **Complexity of LIP-KIM**. The time complexity of Alg. 1 is $O((n^q+n^a)^2(i^q+i^a) + (n^q+n^a)(i^q+i^a)^2 + (i^q+i^a)^3 + (n^q i^q + n^a i^a + (i^q)^2 + (i^a)^2)d)$; the space complexity of Alg. 1 is $O((n^q + n^a + i^q + i^a)^2 + (n^q + n^a + i^q + i^a)d)$.

Proof: See the supplementary material. □

Since i^q and i^a are usually much smaller than n^q and n^a, and the feature dimension d is a fixed constant, the time and space complexity of LIP-KIM can be re-written as $O((n^q+n^a)^2)$. Compared with LIP-KM which is cubic in time, LIP-KIM is much more efficient. However, it is still

quadratic wrt the total number of the training examples. In the next two subsections, we propose two approximate algorithms to further speed-up the computation.

3.3 LIP-KIMA Algorithm

The quadratic complexity of LIP-KIM comes from the fact that we need to maintain and manipulate two kernel matrices of the size $n_q \times n_q$ and $n_a \times n_a$, respectively. In order to avoid quadratic cost for both time and space, we need an efficient way to approximate/compress the full kernel matrices and update them over time.

Take the kernel matrix for questions as an example. Notice that $\mathcal{K}_t^q$ is symmetric and semi-positive definite; therefore, we can approximate it by eigen-decomposition: $\mathcal{K}_t^q \approx \mathbf{U}_t^q \Lambda_t^q (\mathbf{U}_t^q)'$, where $\mathbf{U}_t^q$ is an $n^q \times r$ orthogonal matrix, and Λ_t^q is an $r \times r$ diagonal matrix whose entries are the largest r eigenvalues of $\mathcal{K}_t^q$. By doing so, we can reduce the space cost from $O(n_q^2)$ to $O(n_q r)$.

When new questions arrive at time $t+1$, we have the new kernel matrix $\mathcal{K}_{t+1}^q$ as shown in Eq. (8). We approximate $\mathcal{K}_{t+1}^q$ by Nyström method [9] as

$$\begin{aligned}
\mathcal{K}_{t+1}^q &= \begin{bmatrix} \mathcal{K}_t^q & (\mathbf{k}_{t+1}^q)' \\ \mathbf{k}_{t+1}^q & \mathbf{h}_{t+1}^q \end{bmatrix} \\
&\approx \begin{bmatrix} \mathcal{K}_t^q \\ \mathbf{k}_{t+1}^q \end{bmatrix} (\mathcal{K}_t^q)^{-1} \begin{bmatrix} \mathcal{K}_t^q \\ \mathbf{k}_{t+1}^q \end{bmatrix}' \\
&\approx \begin{bmatrix} \mathcal{K}_t^q \\ \mathbf{k}_{t+1}^q \end{bmatrix} \mathbf{U}_t^q (\Lambda_t^q)^{-1} \Lambda_t^q (\Lambda_t^q)^{-1} (\mathbf{U}_t^q)' \begin{bmatrix} \mathcal{K}_t^q \\ \mathbf{k}_{t+1}^q \end{bmatrix}' \\
&= \begin{bmatrix} \mathbf{U}_t^q \\ \mathbf{k}_{t+1}^q \mathbf{U}_t^q (\Lambda_t^q)^{-1} \end{bmatrix} \Lambda_t^q \begin{bmatrix} \mathbf{U}_t^q \\ \mathbf{k}_{t+1}^q \mathbf{U}_t^q (\Lambda_t^q)^{-1} \end{bmatrix}' \\
&= \mathbf{X}_1 \Lambda_t^q \mathbf{X}_1' \qquad (15)
\end{aligned}$$

where we define $\mathbf{X}_1$ as the $(n^q + i^q) \times r$ matrix $[\mathbf{U}_t^q; \mathbf{k}_{t+1}^q \mathbf{U}_t^q (\Lambda_t^q)^{-1}]$.

To make the decomposition of $\mathcal{K}_{t+1}^q$ reusable for future updates, we need to find the eigen-decomposition form of $\mathcal{K}_{t+1}^q$. To this end, we first perform the Singular Value Decomposition (SVD) on $\mathbf{X}_1$ as $\mathbf{X}_1 = \mathbf{P} \Lambda_1^q \mathbf{Q}'$, where both $\mathbf{P}$ and $\mathbf{Q}$ are orthogonal. Next, we perform eigen-decomposition on an $r \times r$ matrix $\mathbf{X}_2 = \Lambda_1^q \mathbf{Q}' \Lambda_t^q \mathbf{Q} \Lambda_1^q$, that is, $\mathbf{X}_2 = \mathbf{V} \Lambda^q \mathbf{V}'$.

Based on the above two steps, we have the approximate eigen-decomposition of the new kernel matrix $\mathcal{K}_{t+1}^q$ as

$$\begin{aligned}
\mathcal{K}_{t+1}^q &\approx \mathbf{X}_1 \Lambda_t^q \mathbf{X}_1' \\
&= \mathbf{P} (\Lambda_1^q \mathbf{Q}' \Lambda_t^q \mathbf{Q} \Lambda_1^q) \mathbf{P}' \\
&= \mathbf{P} (\mathbf{V} \Lambda^q \mathbf{V}') \mathbf{P}' \\
&= \mathbf{U}_{t+1}^q \Lambda_{t+1}^q (\mathbf{U}_{t+1}^q)' \qquad (16)
\end{aligned}$$

where we define $\mathbf{U}_{t+1}^q = \mathbf{P}\mathbf{V}$ and $\Lambda_{t+1}^q = \Lambda^q$. Notice that $\mathbf{U}_{t+1}^q$ is orthogonal because both $\mathbf{P}$ and $\mathbf{V}$ are orthogonal.

We use the same approach to approximate and update the kernel matrix for answers: $\mathcal{K}_{t+1}^a \approx \mathbf{U}_{t+1}^a \Lambda_{t+1}^a (\mathbf{U}_{t+1}^a)'$. We further define the following notations to simplify the algorithm description

$$\begin{aligned}
\mathbf{U} &= [\mathbf{U}_{t+1}^q, \mathbf{0}; \mathbf{0}, \mathbf{U}_{t+1}^a] \\
\Lambda &= [\Lambda_{t+1}^q, \mathbf{0}; \mathbf{0}, \Lambda_{t+1}^a] \\
\mathbf{G} &= [\mathbf{I}, -\mathbf{M}_{t+1}; -\mathbf{M}_{t+1}', \mathbf{M}_{t+1}' \mathbf{M}_{t+1}] \qquad (17)
\end{aligned}$$

Algorithm 2 The LIP-KIMA Algorithm.

Input: $\mathbf{U}_t^q$, Λ_t^q, $\mathbf{U}_t^a$, Λ_t^a, $\mathbf{F}_t^q$, $\mathbf{f}_{t+1}^q$, $\mathbf{F}_t^a$, $\mathbf{f}_{t+1}^a$, $\mathbf{Y}_t^q$, $\mathbf{y}_{t+1}^q$, $\mathbf{Y}_t^a$, $\mathbf{y}_{t+1}^a$, $\mathbf{M}_t$ and $\mathbf{m}_{t+1}$

Output: $\boldsymbol{\beta}_{t+1}$, $\mathbf{U}_{t+1}^q$, Λ_{t+1}^q, $\mathbf{U}_{t+1}^a$ and Λ_{t+1}^a

1: compute the new kernels $\mathbf{k}_{t+1}^q$ and $\mathbf{k}_{t+1}^a$ in Eq. (8);
2: update $\mathbf{U}_{t+1}^q$, Λ_{t+1}^q, $\mathbf{U}_{t+1}^a$ and Λ_{t+1}^a as Eq. (16);
3: define $\mathbf{U}$, Λ and $\mathbf{G}$ as Eq. (17);
4: define $\mathbf{A}$ and $\mathbf{B}$ as $[\mathbf{U}, \theta\mathbf{G}\mathbf{U}]$ and $[\Lambda\mathbf{U}'; \Lambda\mathbf{U}']$;
5: update $\boldsymbol{\beta}_{t+1}$ as $\frac{1}{\lambda}(\mathbf{I} - \mathbf{A}(\lambda\mathbf{I} + \mathbf{B}\mathbf{A})^{-1}\mathbf{B}) \begin{bmatrix} \mathbf{Y}_{t+1}^q \\ \mathbf{Y}_{t+1}^a \end{bmatrix}$;
6: **return** $\boldsymbol{\beta}_{t+1}$, $\mathbf{U}_{t+1}^q$, Λ_{t+1}^q, $\mathbf{U}_{t+1}^a$ and Λ_{t+1}^a;

Then, we have the following approximation for the $\mathbf{S}_{t+1}$ matrix defined in Eq. (9)

$$\begin{aligned}
\mathbf{S}_{t+1} &= \begin{bmatrix} (\theta+1)\mathcal{K}_{t+1}^q + \lambda\mathbf{I} & -\theta\mathbf{M}_{t+1}\mathcal{K}_{t+1}^a \\ -\theta\mathbf{M}_{t+1}'\mathcal{K}_{t+1}^q & \mathcal{K}_{t+1}^a + \theta\mathbf{M}_{t+1}'\mathbf{M}_{t+1}\mathcal{K}_{t+1}^a + \lambda\mathbf{I} \end{bmatrix} \\
&= \lambda\mathbf{I} + (\mathbf{I} + \theta\mathbf{G}) \begin{bmatrix} \mathcal{K}_{t+1}^q & \mathbf{0} \\ \mathbf{0} & \mathcal{K}_{t+1}^a \end{bmatrix} \\
&\approx \lambda\mathbf{I} + \mathbf{U}\Lambda\mathbf{U}' + \theta\mathbf{G}\mathbf{U}\Lambda\mathbf{U}' \\
&= \lambda\mathbf{I} + \mathbf{A}\mathbf{B} \qquad (18)
\end{aligned}$$

where we define $\mathbf{A} = [\mathbf{U}, \theta\mathbf{G}\mathbf{U}]$ and $\mathbf{B} = [\Lambda\mathbf{U}'; \Lambda\mathbf{U}']$.

Finally, by applying Matrix Inversion Lemma [10], [11] to Eq. (18), we can learn the coefficients $\boldsymbol{\beta}_{t+1}$ as

$$\begin{aligned}
\boldsymbol{\beta}_{t+1} &= \mathbf{S}_{t+1}^{-1} \begin{bmatrix} \mathbf{Y}_{t+1}^q \\ \mathbf{Y}_{t+1}^a \end{bmatrix} \\
&\approx (\lambda\mathbf{I} + \mathbf{A}\mathbf{B})^{-1} \begin{bmatrix} \mathbf{Y}_{t+1}^q \\ \mathbf{Y}_{t+1}^a \end{bmatrix} \\
&= \frac{1}{\lambda}(\mathbf{I} - \mathbf{A}(\lambda\mathbf{I} + \mathbf{B}\mathbf{A})^{-1}\mathbf{B}) \begin{bmatrix} \mathbf{Y}_{t+1}^q \\ \mathbf{Y}_{t+1}^a \end{bmatrix} \qquad (19)
\end{aligned}$$

The complete algorithm of LIP-KIMA is summarized in Alg. 2. As we can see, in addition to $\boldsymbol{\beta}_{t+1}$, the only variables we need to update are $\mathbf{U}_t^q$, Λ_t^q, $\mathbf{U}_t^a$ and Λ_t^a. Compared to Alg. 1, we do not need to store $\boldsymbol{\beta}_t$; instead we need to store $\mathbf{Y}_t^q$ and $\mathbf{Y}_t^a$. More importantly, we do not need to store $\mathbf{S}_t$; instead, we only need to store the much smaller matrices of $\mathbf{U}_t^q$, Λ_t^q, $\mathbf{U}_t^a$ and Λ_t^a.

Algorithm Analysis. The effectiveness of LIP-KIMA is summarized in Lemma 4. According to Lemma 4, there are two possible places where we could introduce the approximation error in the LIP-KIMA algorithm, including (a) eigen-decomposition for the $\mathcal{K}_t$ and (b) the Nyström method for $\mathcal{K}_{t+1}$. Notice that if we only do eigen-decomposition at $t = 1$, such approximation error might be accumulated and amplified over time. In practice, we could 're-start' the algorithm every few time ticks, that is, to re-compute (as opposed to approximate) the eigen-decomposition for the current kernel matrix.

Lemma 4: **Effectiveness of LIP-KIMA**. Let $\boldsymbol{\beta}_{t+1}^*$ be the output of Eq. (5) at time $t+1$, and $\boldsymbol{\beta}_{t+1}$ be the output of Alg. 2 updated from time t to $t+1$, we have $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_{t+1}^*$ if $\mathcal{K}_t = \mathbf{U}_t \Lambda_t (\mathbf{U}_t)'$ and $\mathbf{h}_{t+1} = \mathbf{k}_{t+1} (\mathcal{K}_t)^{-1} (\mathbf{k}_{t+1})'$ hold for both questions and answers.

Proof: See the supplementary material. □

The time complexity and space complexity of Alg. 2 is summarized in the following lemma. It basically says that

the LIP-KIMA algorithm requires *linear* time and space wrt the total number of questions and answers.

Lemma 5: **Complexity of LIP-KIMA**. The time complexity of Alg. 2 is $O((n^q+n^a+i^q+i^a)r^2+r^3+(n^q i^q+n^a i^a)d)$; the space complexity of Alg. 2 is $O((n^q+n^a+i^q+i^a)d+(n^q+n^a+i^q+i^a)r+r^2)$.

Proof: See the supplementary material. □

Since i^q and i^a are usually much smaller than n^q and n^a, and the low rank r and feature dimension d are fixed constants, the time complexity and space complexity of Alg. 2 can be re-written as $O(n^q+n^a)$, which is linear in terms of the total number of questions and answers.

3.4 LIP-KIMAA Algorithm

Compared with LIP-KIM, LIP-KIMA is much more scalable, being *linear* in terms of both time and space complexity. However, if the new training examples arrive in a stream-like, ever-growing fashion, a linear algorithm might be still too expensive. To address this issue, we further present the LIP-KIMAA algorithm to reduce the complexity to be *sub-linear*.

Our LIP-KIMAA is built upon LIP-KIMA. The main difference between LIP-KIMAA and LIP-KIMA is that we add an additional filtering step between Step 1 and Step 2 in Alg. 2. That is, when new questions and answers arrive at time $t+1$, we first treat them as test set and apply the existing model at time t on this test set. Based on the prediction results, we only add the questions and answers whose prediction error is larger than a given threshold *th*. Notice that the complexity of LIP-KIMA is linear wrt the number of questions and answers; as a result, our LIP-KIMAA scales linearly wrt the number of *remaining* questions and answers after the filtering steps. Therefore, LIP-KIMAA scales sub-linearly wrt to the total number of questions and answers in both time and space. We omit the detailed algorithm for brevity.

4 Variants and Extensions

The proposed LIP-KIM and its two approximate algorithms (LIP-KIMA and LIP-LIMAA) are *comprehensive*. In terms of the modeling power, they capture all the three aspects (non-linearity, coupling, and dynamics). In this section, we show that our algorithms are also *flexible*. For example, if only a subset of these three aspects matter with the prediction performance for some applications, our algorithms can be naturally adapted to these special cases (e.g., see LIP-IM and LIP-KIA in the supplementary material). Here, we further show the flexibility of our models in terms of temporal forgetting and feature/example selection.

4.1 LIP-IMF: LIP-IM with Forgetting

Based on LIP-IM which considers the coupling and dynamic aspects, we can further add a forgetting factor η ($0 \leqslant \eta \leqslant 1$) to fade the relative weights of old examples. Taking question prediction as an example, we can add the forgetting factor as

$$\mathbf{F}^q_{t+1} = \left[\eta^{0.5}\mathbf{F}^q_t; \mathbf{f}^q_{t+1}\right], \qquad \mathbf{Y}^q_{t+1} = \left[\eta^{0.5}\mathbf{Y}^q_t; \mathbf{y}^q_{t+1}\right]$$

Algorithm 3 The LIP-IMF Algorithm.

Input: $\mathbf{S}_t^{-1}$, $\boldsymbol{\alpha}_t$, $\mathbf{f}^q_{t+1}$, $\mathbf{f}^a_{t+1}$, $\mathbf{y}^q_{t+1}$, $\mathbf{y}^a_{t+1}$, and $\mathbf{m}_{t+1}$
Output: $\boldsymbol{\alpha}_{t+1}$, $\mathbf{S}_{t+1}^{-1}$

1: define $\mathbf{L}$, $\mathbf{R}$, and $\mathbf{C}_f$ as Eq. (22), Eq. (23), and Eq. (24);
2: update $\mathbf{S}_{t+1}^{-1}$ as $\frac{1}{\eta}(\mathbf{I}-\mathbf{C}_f)\mathbf{S}_t^{-1}$;
3: update $\boldsymbol{\alpha}_{t+1}$ as $(\mathbf{I}-\mathbf{C}_f)\left(\boldsymbol{\alpha}_t + \frac{1}{\eta}\mathbf{S}_t^{-1}\begin{bmatrix}(\mathbf{f}^q_{t+1})'\mathbf{y}^q_{t+1}\\ (\mathbf{f}^a_{t+1})'\mathbf{y}^a_{t+1}\end{bmatrix}\right)$;
4: **return** $\boldsymbol{\alpha}_{t+1}$, $\mathbf{S}_{t+1}^{-1}$;

where we fade the old examples by factor $\eta^{0.5}$. Here, η controls the degree of fading effects. For example, if we set $\eta=1$, it degenerates to LIP-IM which treats old examples and new examples equivalently; if we set $\eta=0$, the model ignores the old examples and uses the new examples only.

We can obtain the static solution for LIP-IM with forgetting as

$$\begin{aligned}\boldsymbol{\alpha}_{t+1} = &\left(\eta\mathbf{S}_t + \begin{bmatrix}(\theta+1)(\mathbf{f}^q_{t+1})'\mathbf{f}^q_{t+1} & -\theta(\mathbf{f}^q)'_{t+1}\mathbf{m}_{t+1}\mathbf{f}^a_{t+1}\\ -\theta(\mathbf{f}^a_{t+1})'\mathbf{m}'_{t+1}\mathbf{f}^q_{t+1} & (\mathbf{f}^a_{t+1})'\mathbf{f}^a_{t+1}+\theta(\mathbf{f}^a_{t+1})'\mathbf{m}'_{t+1}\mathbf{m}_{t+1}\mathbf{f}^a_{t+1}\end{bmatrix}\right)^{-1}\\ &\left(\eta\begin{bmatrix}(\mathbf{F}^q_t)'\mathbf{Y}^q_t\\ (\mathbf{F}^a_t)'\mathbf{Y}^a_t\end{bmatrix}+\begin{bmatrix}(\mathbf{f}^q_{t+1})'\mathbf{y}^q_{t+1}\\ (\mathbf{f}^a_{t+1})'\mathbf{y}^a_{t+1}\end{bmatrix}\right)\end{aligned} \tag{20}$$

where $\mathbf{S}_t$ is defined in Eq. (21).

$$\mathbf{S}_t = \begin{bmatrix}(\theta+1)(\mathbf{F}^q_t)'\mathbf{F}^q_t+\lambda\mathbf{I} & -\theta(\mathbf{F}^q_t)'\mathbf{M}_t\mathbf{F}^a_t\\ -\theta(\mathbf{F}^a_t)'\mathbf{M}'_t\mathbf{F}^q_t & (\mathbf{F}^a_t)'\mathbf{F}^a_t+\theta(\mathbf{F}^a_t)'\mathbf{M}'_t\mathbf{M}_t\mathbf{F}^a_t+\lambda\mathbf{I}\end{bmatrix} \tag{21}$$

Next, by using the matrices $\mathbf{L}$ and $\mathbf{R}$ as defined in Eq. (22) and Eq. (23),

$$\begin{aligned}\mathbf{L} = &[(\theta+1)(\mathbf{f}^q_{t+1})', -\theta(\mathbf{f}^q_{t+1})', \mathbf{0}, \mathbf{0};\\ &\mathbf{0}, \mathbf{0}, -\theta(\mathbf{f}^a_{t+1})'\mathbf{m}'_{t+1}, [(\mathbf{f}^a_{t+1})', \theta(\mathbf{f}^a_{t+1})'\mathbf{m}'_{t+1}]]\end{aligned} \tag{22}$$

$$\mathbf{R} = [\mathbf{f}^q_{t+1}, \mathbf{0}; \mathbf{0}, \mathbf{m}_{t+1}\mathbf{f}^a_{t+1}; \mathbf{f}^q_{t+1}, \mathbf{0}; \mathbf{0}, [\mathbf{f}^a_{t+1}; \mathbf{m}_{t+1}\mathbf{f}^a_{t+1}]] \tag{23}$$

and further defining the following matrix,

$$\mathbf{C}_f = \mathbf{S}_t^{-1}\mathbf{L}(\eta\mathbf{I}+\mathbf{R}\mathbf{S}_t^{-1}\mathbf{L})^{-1}\mathbf{R} \tag{24}$$

we summarize the LIP-IMF algorithm for incrementally updating Eq. (20) in Alg. 3. In the algorithm, we re-use $\mathbf{S}_t^{-1}$ and $\boldsymbol{\alpha}_t$ for each iteration. $\mathbf{L}$ and $\mathbf{R}$ are computed based on new observations only (i.e., $\mathbf{f}^q_{t+1}$, $\mathbf{f}^a_{t+1}$, and $\mathbf{m}_{t+1}$), and $\mathbf{C}_f$ is computed from $\mathbf{L}$, $\mathbf{R}$, and $\mathbf{S}_t^{-1}$.

Algorithm Analysis. Here, we briefly analyze Alg. 3. The following theorem shows the correctness of Alg. 3.

Theorem 2: **Correctness of LIP-IMF**. Let $\boldsymbol{\alpha}^*_{t+1}$ be the output of Eq. (20) at time $t+1$, and $\boldsymbol{\alpha}_{t+1}$ be the output of Alg. 3 updated from time t to $t+1$, we have that $\boldsymbol{\alpha}_{t+1}=\boldsymbol{\alpha}^*_{t+1}$.

Proof: See the supplementary material. □

The complexity of Alg. 3 is summarized in the following lemma. Basically, by ignoring the small terms (i^q and i^a), the time complexity of LIP-IMF can be written as $O(d^2)$.

Lemma 6: **Complexity of LIP-IMF**. The time complexity of Alg. 3 is $O((i^q+i^a)d^2+(i^q+i^a)^2d+(i^q+i^a)^3)$; the space complexity of Alg. 3 is $O(d^2+(i^q+i^a)d)$.

Proof: See the supplementary material. □

4.2 LIP-MS: LIP-M with Feature Selection

In addition to the three aspects that matter with voting prediction, it would be useful to identify a subset of features for both efficiency and interpretability reasons. For LIP-M (i.e., Eq (1)), we can further encode L_1 norm ($\|\boldsymbol{\alpha}^q\|_1$ and $\|\boldsymbol{\alpha}^a\|_1$) to identify a subset of useful features for our prediction problem. This results in the so-called elastic net [12]. In LIP-M, we have constrained the consistency on the prediction space (i.e., the predicted voting of a question is close to that of its answer). We can also add the constraint on the parameter space. For example, if questions and answers use the same features, we can add $\|\boldsymbol{\alpha}^q - \boldsymbol{\alpha}^q\|_2^2$ into the formulation to constrain their parameter distance. The assumption is that the same feature should play similar roles in both question prediction and answer prediction. Putting the above ideas together, we have

$$\begin{aligned}\min_{\boldsymbol{\alpha}^q,\boldsymbol{\alpha}^a} \quad & \textstyle\sum_{i=1}^{n^q}(\mathbf{F}^q(i,:)\boldsymbol{\alpha}^q - \mathbf{Y}^q(i))^2 + \sum_{i=1}^{n^a}(\mathbf{F}^a(i,:)\boldsymbol{\alpha}^a - \mathbf{Y}^a(i))^2 \\ & +\theta\textstyle\sum_{i=1}^{n^q}(\mathbf{F}^q(i,:)\boldsymbol{\alpha}^q - \mathbf{M}(i,:)\mathbf{F}^a\boldsymbol{\alpha}^a)^2 \\ & +\lambda_1(1-\rho_1)(\|\boldsymbol{\alpha}^q\|_2^2 + \|\boldsymbol{\alpha}^a\|_2^2) + \lambda_1\rho_1(\|\boldsymbol{\alpha}^q\|_1 + \|\boldsymbol{\alpha}^a\|_1) \\ & +\lambda_2(1-\rho_2)\|\boldsymbol{\alpha}^q - \boldsymbol{\alpha}^a\|_2^2 + \lambda_2\rho_2\|\boldsymbol{\alpha}^q - \boldsymbol{\alpha}^a\|_1 \end{aligned} \tag{25}$$

where λ_1 and λ_2 are used to control the importance of the regularization and parameter consistency, respectively; ρ_1 and ρ_2 are used to control the relative importance of L_2 norm and L_1 norm. In addition to the separate feature selection encoded in $\|\boldsymbol{\alpha}^q\|_1$ and $\|\boldsymbol{\alpha}^a\|_1$, the $\|\boldsymbol{\alpha}^q - \boldsymbol{\alpha}^a\|_1$ term in the above formulation is used to find commonly useful features for both question prediction and answer prediction. This kind of term is also known as fused lasso [13]. For the above formulation, we need to normalize the features so that $\sum_i \mathbf{F}(i,j) = 0$ and $\sum_j \mathbf{F}(i,j)^2 = 1$.

Note that there are several special cases of Eq. (25). For example, if we set $\lambda_2 = 0$ and $\rho_1 = 0$, it degenerates to LIP-M; if we set $\lambda_2 = 0$ and $\rho_1 = 1$, it becomes coupled lasso for our problem; if we set $\lambda_2 = 0$ and $\rho_1 \in (0,1)$, it becomes the coupled elastic net. Overall, the proposed formulation in Eq. (25) considers the consistency in both prediction space and parameter space; it also incorporates separate and common feature selection.

To solve Eq. (25), we adopt the alternating strategy. That is, we first fix $\boldsymbol{\alpha}^a$ and update $\boldsymbol{\alpha}^q$ using the proximal gradient descent method [14], and then fix $\boldsymbol{\alpha}^q$ and update $\boldsymbol{\alpha}^a$. We next show how we update $\boldsymbol{\alpha}^q$.

When $\boldsymbol{\alpha}^a$ is fixed, the optimization problem in Eq. (25) becomes

$$\begin{aligned}\min_{\boldsymbol{\alpha}^q} \quad & \|\mathbf{F}^q\boldsymbol{\alpha}^q - \mathbf{Y}^q\|_2^2 + \theta\|\mathbf{F}^q\boldsymbol{\alpha}^q - \mathbf{M}\mathbf{F}^a\boldsymbol{\alpha}^a\|_2^2 \\ & +\lambda_1(1-\rho_1)\|\boldsymbol{\alpha}^q\|_2^2 + \lambda_1\rho_1\|\boldsymbol{\alpha}^q\|_1 \\ & +\lambda_2(1-\rho_2)\|\boldsymbol{\alpha}^q - \boldsymbol{\alpha}^a\|_2^2 + \lambda_2\rho_2\|\boldsymbol{\alpha}^q - \boldsymbol{\alpha}^a\|_1 \end{aligned} \tag{26}$$

Based on Theorem 1 in [15], we can first solve the L_1 norm $\|\boldsymbol{\alpha}^q - \boldsymbol{\alpha}^a\|_1$ while ignoring $\|\boldsymbol{\alpha}^q\|_1$ for the above equation. Therefore, by defining $\boldsymbol{\gamma}^q = \boldsymbol{\alpha}^q - \boldsymbol{\alpha}^a$, we have

$$\begin{aligned}\min_{\boldsymbol{\alpha}^q} \quad & \|\mathbf{F}^q(\boldsymbol{\gamma}^q + \boldsymbol{\alpha}^a) - \mathbf{Y}^q\|_2^2 + \theta\|\mathbf{F}^q(\boldsymbol{\gamma}^q + \boldsymbol{\alpha}^a) - \mathbf{M}\mathbf{F}^a\boldsymbol{\alpha}^a\|_2^2 \\ & +\lambda_1(1-\rho_1)\|\boldsymbol{\gamma}^q + \boldsymbol{\alpha}^a\|_2^2 + \lambda_2(1-\rho_2)\|\boldsymbol{\gamma}^q\|_2^2 + \lambda_2\rho_2\|\boldsymbol{\gamma}^q\|_1 \end{aligned} \tag{27}$$

Using the proximal gradient descent method, we can solve Eq. (27) as

$$\begin{aligned}\boldsymbol{\gamma}^q &= \boldsymbol{\gamma}^q - s*((\mathbf{F}^q)'(\mathbf{F}^q\boldsymbol{\alpha}^q - \mathbf{Y}^q) + \theta(\mathbf{F}^q)'(\mathbf{F}^q\boldsymbol{\alpha}^q - \mathbf{M}\mathbf{F}^a\boldsymbol{\alpha}^a) \\ &\quad +\lambda_1(1-\rho_1)\boldsymbol{\alpha}^q + \lambda_2(1-\rho_2)\boldsymbol{\gamma}^q) \\ \boldsymbol{\gamma}^q &= sgn(\boldsymbol{\gamma}^q)\odot max(|\boldsymbol{\gamma}^q| - \lambda_2\rho_2, 0) \end{aligned} \tag{28}$$

Algorithm 4 The LIP-MS Algorithm.

Input: $\mathbf{F}^q$, $\mathbf{F}^a$, $\mathbf{Y}^q$, $\mathbf{Y}^a$, and $\mathbf{M}$
Output: $\boldsymbol{\alpha}^q$ and $\boldsymbol{\alpha}^a$

```
initialize α^q and α^a;
while not convergent do
    update α^q as Eq. (28) - (29);
    update α^a as Eq. (30);
end while
return α^q and α^a;
```

where s is the learning step size, $\odot$ is the Hadamard product, and sgn is defined element-wisely with $sgn(t) = 1$ if $t > 0$, $sgn(t) = 0$ if $t = 0$, $sgn(t) = -1$ if $t < 0$.

Next, based on Theorem 1 in [15], we have the updating rule for $\boldsymbol{\alpha}^q$

$$\boldsymbol{\alpha}^q = sgn(\boldsymbol{\gamma}^q + \boldsymbol{\alpha}^a)\odot max(|\boldsymbol{\gamma}^q + \boldsymbol{\alpha}^a| - \lambda_1\rho_1, 0) \tag{29}$$

Analogously, we can derive the update rule for $\boldsymbol{\alpha}^a$

$$\begin{aligned}\boldsymbol{\gamma}^a &= \boldsymbol{\gamma}^a - s*((\mathbf{F}^a)'(\mathbf{F}^a\boldsymbol{\alpha}^a - \mathbf{Y}^a) + \theta(\mathbf{F}^a)'(\mathbf{M}'\mathbf{M}\mathbf{F}^a\boldsymbol{\alpha}^a \\ &\quad -\mathbf{M}'\mathbf{F}^q\boldsymbol{\alpha}^q) + \lambda_1(1-\rho_1)\boldsymbol{\alpha}^a + \lambda_2(1-\rho_2)\boldsymbol{\gamma}^a) \\ \boldsymbol{\gamma}^a &= sgn(\boldsymbol{\gamma}^a)\odot max(|\boldsymbol{\gamma}^a| - \lambda_2\rho_2, 0) \\ \boldsymbol{\alpha}^a &= sgn(\boldsymbol{\gamma}^a + \boldsymbol{\alpha}^q)\odot max(|\boldsymbol{\gamma}^a + \boldsymbol{\alpha}^q| - \lambda_1\rho_1, 0) \end{aligned} \tag{30}$$

where $\boldsymbol{\gamma}^a = \boldsymbol{\alpha}^a - \boldsymbol{\alpha}^q$. The overall algorithm for solving Eq. (25) is summarized in Alg. 4.

For the special case of Eq. (25) with $\rho_1 = \rho_2 = 0$, we have the closed-form solution as follows

$$\begin{aligned}\boldsymbol{\alpha} &= \arg\min_{\boldsymbol{\alpha}^q,\boldsymbol{\alpha}^a} \|\mathbf{X}^q\boldsymbol{\beta}^q - \mathbf{Y}^q\|_2^2 + \|\mathbf{X}^a\boldsymbol{\beta}^a - \mathbf{Y}^a\|_2^2 \\ &\quad +\theta\|\mathbf{X}^q\boldsymbol{\alpha}^q - \mathbf{M}\mathbf{X}^a\boldsymbol{\alpha}^a\|_2^2 + \lambda_1(\|\boldsymbol{\alpha}^q\|_2^2 + \|\boldsymbol{\alpha}^a\|_2^2) + \lambda_2\|\boldsymbol{\alpha}^q - \boldsymbol{\alpha}^q\|_2^2 \\ &= \left(\begin{bmatrix} (\theta+1)(\mathbf{X}^q)'\mathbf{X}^q + (\lambda_1+\lambda_2)\mathbf{I} & -\theta(\mathbf{X}^q)'\mathbf{M}'\mathbf{X}^a - \lambda_2\mathbf{I} \\ -\theta(\mathbf{X}^a)'\mathbf{M}'\mathbf{X}^q - \lambda_2\mathbf{I} & -\theta(\mathbf{X}^q)'\mathbf{M}\mathbf{X}^a - \lambda_2\mathbf{I} \\ & (\mathbf{X}^a)'\mathbf{X}^a + \theta(\mathbf{X}^a)'\mathbf{M}'\mathbf{M}\mathbf{X}^a + (\lambda_1+\lambda_2)\mathbf{I} \end{bmatrix}\right)^{-1}\begin{bmatrix}(\mathbf{X}^q)'\mathbf{Y}^q \\ (\mathbf{X}^a)'\mathbf{Y}^a\end{bmatrix} \end{aligned} \tag{31}$$

Algorithm Analysis. Here, we briefly analyze Alg. 4. The following lemma shows the effectiveness of Alg. 4.

Lemma 7: **Optimality of LIP-MS**. Alg. 4 finds a local minimum for Eq. (25).

Proof: See the supplementary material. □

The complexity of Alg. 4 is summarized in the following lemma. It basically says that the LIP-MS requires linear time and space wrt the number of questions and answers.

Lemma 8: **Complexity of LIP-MS**. The time complexity of Alg. 4 is $O((n^q + n^a)dl)$ where l is the maximum iteration number; the space complexity of Alg. 4 is $O((n^q + n^a)d)$.

Proof: See the supplementary material. □

4.3 LIP-KMS: LIP-KM with Example Selection

Similar to LIP-MS, we can also add the L_1 norm for the kernelized case. L_1 norm in the kernelized case plays the role of example selection, i.e., identifying a subset of useful questions/answers for the prediction problem. By encoding the L_1 norm into Eq. (4), we have the LIP-KMS optimization formulation as follows

$$\begin{aligned}\min_{\boldsymbol{\beta}^q,\boldsymbol{\beta}^a} \quad & \textstyle\sum_{i=1}^{n^q}(\mathcal{K}^q(i,:)\boldsymbol{\beta}^q - \mathbf{Y}^q(i))^2 + \sum_{i=1}^{n^a}(\mathcal{K}^a(i,:)\boldsymbol{\beta}^a - \mathbf{Y}^a(i))^2 \\ & +\theta\textstyle\sum_{i=1}^{n^q}(\mathcal{K}^q(i,:)\boldsymbol{\beta}^q - \mathbf{M}(i,:)\mathcal{K}^a\boldsymbol{\beta}^a)^2 \\ & +\lambda(1-\rho)((\boldsymbol{\beta}^q)'\mathcal{K}^q\boldsymbol{\beta}^q + (\boldsymbol{\beta}^a)'\mathcal{K}^a\boldsymbol{\beta}^a) + \lambda\rho(\|\boldsymbol{\beta}^q\|_1 + \|\boldsymbol{\beta}^a\|_1) \end{aligned} \tag{32}$$

Algorithm 5 The LIP-KMS Algorithm.

Input: $\mathbf{U}^q$, Λ^q, $\mathbf{U}^a$, Λ^a, $\mathbf{Y}^q$, $\mathbf{Y}^a$, and $\mathbf{M}$
Output: $\boldsymbol{\beta}^q$ and $\boldsymbol{\beta}^a$

```
initialize β^q and β^a;
while not convergent do
    update β^q as Eq. (33);
    update β^a as Eq. (34);
end while
return β^q and β^a;
```

where ρ is used to control the relative importance of L_1 norm and L_2 norm. Since the question prediction and answer prediction do not share the same feature space, we do not consider the $\|\boldsymbol{\beta}^q - \boldsymbol{\beta}^a\|_1$ term in the above equation.

A straightforward approach to solving the above formulation would cost cubic time. Hence, we adopt the approximation strategy in LIP-KIMA. That is, we can approximate the kernel matrices $\mathcal{K}^q$ and $\mathcal{K}^a$ by $\mathbf{U}^q\Lambda^q(\mathbf{U}^q)'$ and $\mathbf{U}^a\Lambda^a(\mathbf{U}^a)'$ for a given time point. Here, we assume that we have already obtained the decomposed matrices, which can be directly decomposed or updated from time t to time $t+1$.

Next, similar to the alternating strategy in LIP-MS, we have

$$\begin{aligned}\boldsymbol{\beta}^q &= \boldsymbol{\beta}^q - s*(\mathbf{U}^q(\Lambda^q)^2(\mathbf{U}^q)'\boldsymbol{\beta}^q - \mathbf{U}^q\Lambda^q(\mathbf{U}^q)'\mathbf{Y}^q \\ &\quad +\theta(\mathbf{U}^q(\Lambda^q)^2(\mathbf{U}^q)'\boldsymbol{\beta}^q - \mathbf{U}^q\Lambda^q(\mathbf{U}^q)'\mathbf{M}\mathbf{U}^a\Lambda^a(\mathbf{U}^a)'\boldsymbol{\beta}^a) \\ &\quad +\lambda(1-\rho)\mathbf{U}^q\Lambda^q(\mathbf{U}^q)'\boldsymbol{\beta}^q) \\ \boldsymbol{\beta}^q &= sgn(\boldsymbol{\beta}^q)\odot max(|\boldsymbol{\beta}^q| - \lambda\rho, 0) \end{aligned} \quad (33)$$

and

$$\begin{aligned}\boldsymbol{\beta}^a &= \boldsymbol{\beta}^a - s*(\mathbf{U}^a(\Lambda^a)^2(\mathbf{U}^a)'\boldsymbol{\beta}^a - \mathbf{U}^a\Lambda^a(\mathbf{U}^a)'\mathbf{Y}^a + \theta(\mathbf{U}^a\Lambda^a(\mathbf{U}^a)' \\ &\quad \mathbf{M}'\mathbf{M}\mathbf{U}^a\Lambda^a(\mathbf{U}^a)'\boldsymbol{\beta}^a - \mathbf{U}^a\Lambda^a(\mathbf{U}^a)'\mathbf{M}'\mathbf{U}^q\Lambda^q(\mathbf{U}^q)'\boldsymbol{\beta}^q) \\ &\quad +\lambda(1-\rho)\mathbf{U}^a\Lambda^a(\mathbf{U}^a)'\boldsymbol{\beta}^a) \\ \boldsymbol{\beta}^a &= sgn(\boldsymbol{\beta}^a)\odot max(|\boldsymbol{\beta}^a| - \lambda\rho, 0) \end{aligned} \quad (34)$$

for updating $\boldsymbol{\beta}^q$ and $\boldsymbol{\beta}^a$, respectively. The overall algorithm is summarized in Alg. 5.

Similar to the filtering step in LIP-KIMAA, LIP-KMS can also be used to filter out less useful training examples for the prediction problem. We will experimentally evaluate this in the next section.

Algorithm Analysis. Here, we briefly analyze Alg. 5. The following lemma shows the effectiveness of Alg. 5.

Lemma 9: **Optimality of LIP-KMS**. Ignoring the approximation error of kernel matrix decomposition, Alg. 5 finds a local minimum for Eq. (32).

Proof: See the supplementary material. □

The complexity of Alg. 5 is summarized in the following lemma. It basically says that the algorithm requires linear time and space wrt the number of questions and answers.

Lemma 10: **Complexity of LIP-KMS**. The time complexity of Alg. 5 is $O((n^q + n^a)rl)$ where l is the maximum iteration number; the space complexity of Alg. 5 is $O((n^q + n^a)r)$.

Proof: See the supplementary material. □

TABLE 3
The statistics of SO and Math data sets.

Data	Questions	Answers	Users	Votes
SO	1,966,272	4,282,570	756,695	14,056,000
Math	16,638	32,876	12,526	202,932

4.4 Summarization

Finally, we make a comparison of different algorithms in Fig. 1 in terms of their modeling power and efficiency. The results are summarized in Table 2. In this table, we first check whether a given algorithm captures each of the three desired aspects (non-linearity, coupling, and dynamics). As we can see, only our LIP-KIM, LIP-KIMA, and LIP-KIMAA algorithms meet this requirement. We also summarize the time complexity of the algorithm for updating the model at each time tick, where the smaller terms (e.g., the feature dimensionality d, the number of new training examples i^q, i^a, etc) are omitted for clarity. For non-linear algorithms like KRR [4], support vector regression (SVR) [16], [17], RKRR [6], LIP-KM, and LIP-KIM, they need at least quadratic time for the model training because they typically need to maintain and manipulate the kernel matrix. Such a complexity is unaffordable in many CQA sites. On the other hand, the time complexity of the proposed LIP-KIMA and LIP-KIMAA algorithms is *linear* and *sub-linear* wrt the total number of questions and answers, respectively.

5 Experiments

In this section, we present the experimental evaluations.

5.1 Experiment Setup

We use the data from two real CQA sites, i.e., Stack Overflow (SO) and Mathematics Stack Exchange (Math). SO and Math are two popular CQA sites for programming and math, respectively. Both data sets are officially published and publicly available[3]. The statistics of the two data sets are summarized in Table 3.

For SO and Math data, we use both content and contextual features. For content features, we adopt the "bag of words" model to extract content features after removing the infrequent words. This model is widely used in natural language processing where the frequency of each word is used as a feature for training. In addition, we adopt some commonly used contextual features in the literature, including the questioner's reputation at question creation time, the answerer's reputation at answer creation time, the number of questioner's previous questions at question creation time, and the number of answerer's previous answers at answer creation time. For these features, we can extract them at the moment when the question/answer is posted. For other contextual features, we need to choose a short time window by the end of which the voting score is predicted. With such a fixed time window, we can include some time-related features such as the number of comments/answers received during the fixed time window. In this work, we

3. http://blog.stackoverflow.com/category/cc-wiki-dump/

TABLE 2
The summarization of the algorithms from the genealogy graph in Fig. 1. The complexity is the time complexity for updating the model at each time tick, where n is the total number of the training examples and those smaller terms are omitted for clarity. The right six columns are the proposed algorithms in this paper.

	SVR	KRR	RKRR	CoPs	LIP-IM	LIP-KIA	LIP-KM	LIP-KIM	LIP-KIMA	LIP-KIMAA
Non-linearity	✓	✓	✓	x	x	✓	✓	✓	✓	✓
Coupling	x	x	x	✓	✓	x	✓	✓	✓	✓
Dynamics	x	x	✓	x	✓	✓	x	✓	✓	✓
Complexity	$\geqslant O(n^2)$	$O(n^3)$	$O(n^2)$	$O(n)$	$O(1)$	$O(n)$	$O(n^3)$	$O(n^2)$	$O(n)$	$< O(n)$

fix this time window as 24 hours, and use the number of answers received in 24 hours and the number of comments received in 24 hours as features. Overall, we have 4,180 and 4,444 features for Math data and SO data, respectively (i.e., $d = 4,180$ for Math data and $d = 4,444$ for SO data). For the kernel matrix, we adopt the cosine kernel function due to the sparsity of our feature matrix.

For the prediction target, we restrict our attention to predict the voting score (i.e., the number of upvotes minus the number of downvotes) of a question/answer after it is posted for at least six months. We normalize the voting scores into the range of $[0, 1]$.

To evaluate the dynamic aspect of our algorithms, we start with a small initial training set, and gradually add new examples into the training set in chronological order. For Math data, we start with 5% initial data, add 5% data for each update, and use the latest 10% data as the test set. For SO data whose size is much larger, we start with 0.1% initial data, add 0.1% data for each update, and use the latest 0.1% data as the test set.

For evaluation metrics, we adopt the root mean square error (RMSE) between the real voting score and the estimated score for effectiveness, and the wall-clock time for efficiency. All the efficiency experiments were run on a machine with eight 3.4GHz Intel Cores and 24GB memory.

For the parameters, we tune them by standard cross validation. Unless otherwise specified, we set $\lambda = \theta = 0.1$ for SO data, and $\lambda = \theta = 1$ for Math data. For the low rank r in LIP-KIMA, LIP-KIMAA, and LIP-KIA, we set it as 10. For LIP-KIMAA, we set $th = 0.22$ for Math data and $th = 0.18$ for SO data.

5.2 Effectiveness Results

(A) The Effectiveness Comparisons. We first compare the effectiveness of the proposed algorithms (i.e., LIP-KIM, LIP-KIMA, and LIP-KIMAA) with two state-of-the-art non-linear regression methods, i.e., kernel ridge regression (KRR) [4] and support vector regression (SVR) [17]. The prediction results of questions and answers on the SO and Math data sets are shown in Fig. 2. On SO data, we only report the first few points because some of the algorithms (e.g., KRR) cannot finish training within 1 hour. We do not report the results by linear models (e.g., linear ridge regression) since their performance (RMSE) is much worse than non-linear models.

We make several observations from Fig. 2. First, the proposed LIP-KIM algorithm performs the best in most of the cases. For example, when the size of training set increases to 90% on the Math data, LIP-KIM improves the SVR method by 5.7% for questions and 6.0% for answers. On SO data, LIP-KIM improves the KRR method by up to 35.8% for questions and 3.6% for answers. This indicates that the coupling aspect indeed helps in voting prediction. Second, the performance of the proposed LIP-KIMA algorithm is close to that of LIP-KIM. This result indicates that while LIP-KIMA reduces the time complexity from quadratic to linear, the approximation introduces little performance loss. Finally, although not as good as LIP-KIM and LIP-KIMA, the LIP-KIMAA algorithm is still better than the compared methods for most of the cases.

TABLE 4
Performance gain analysis. Smaller is better. All three aspects of non-linearity, coupling, and dynamics are helpful.

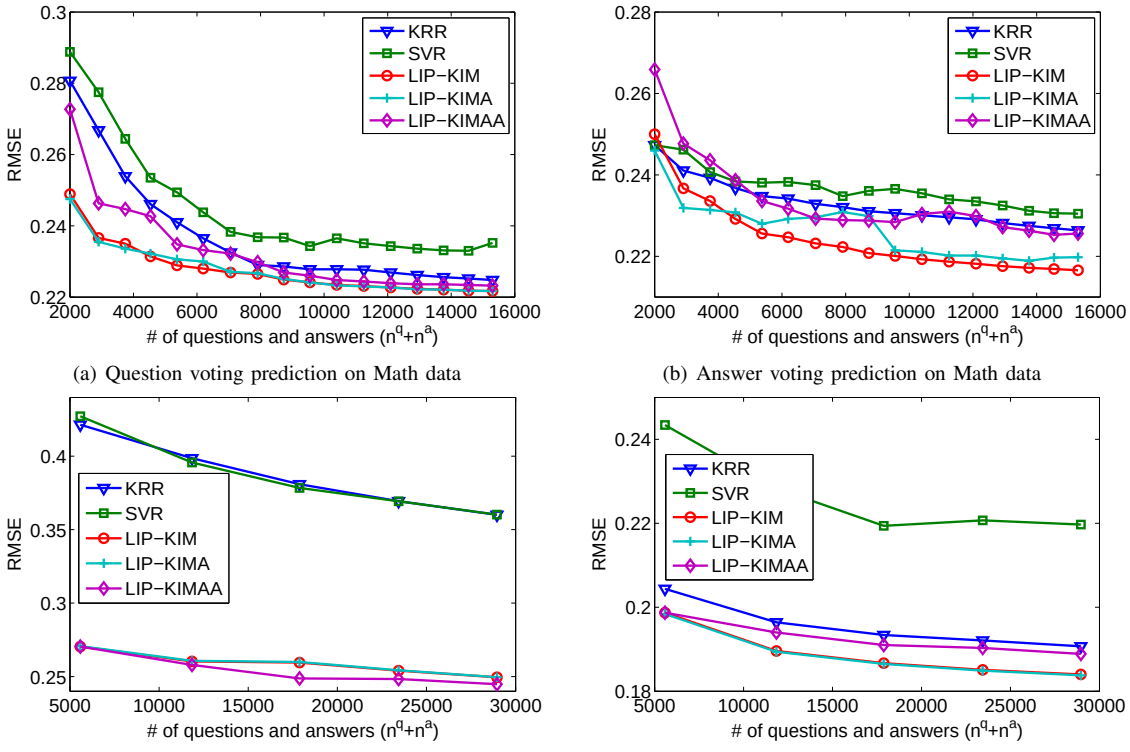
Questions/Answers	SO	Math
RR	0.4920/0.4409	0.2799/0.3860
LIP-K	0.4214/0.2044	0.2461/0.2368
LIP-KM	0.2704/0.1987	0.2314/0.2292
LIP-KIM	0.2595/0.1867	0.2249/0.2208

(B) Performance Gain Analysis. To further show the effects of all the three aspects (i.e., non-linearity, coupling, and dynamics), we analyze the performance gain in Table 4. In the table, LIP-K incorporates non-linearity into RR (ridge regression), LIP-KM incorporates coupling into LIP-K, and LIP-KIM incorporates dynamics into LIP-KM. As we can see, all three aspects are helpful to improve the prediction performance.

(C) Sensitivity Evaluations. For the two parameters (i.e., θ and λ) in our LIP-KIM algorithm, we conduct a parametric study and show the results on the SO data in Fig. 3. We can observe from the figure that, in general, the RMSE results stay stable wrt both parameters in a large range. Similar results are observed for the LIP-KIMA and LIP-KIMAA algorithms, and we omit them for brevity. Overall, we conclude that our methods are robust wrt the two parameters in a large range.

(D) Effectiveness of Forgetting. Next, we evaluate the usefulness of the forgetting factor in LIP-IMF, and report the results for questions and answers on the SO data in Fig. 4. As we can see, introducing the forgetting factor (i.e., fading the effects of old examples) can help to decrease the RMSE results. Additionally, we can observe that the better results can be obtained when η is set around 0.6.

(E) Feature Selection Results. Next, we present some



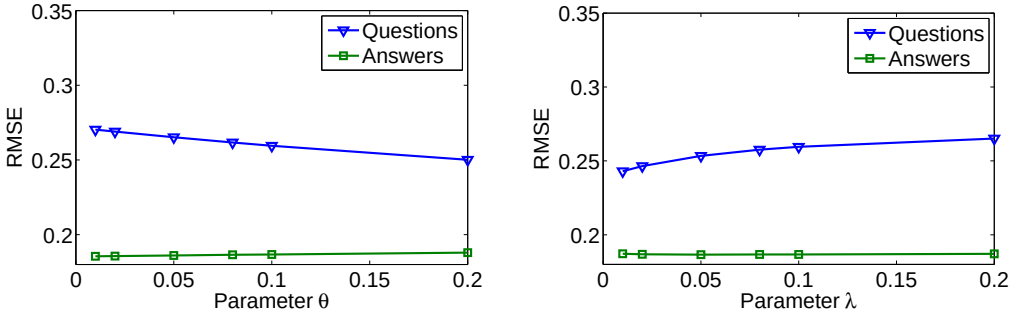
(a) Question voting prediction on Math data

(b) Answer voting prediction on Math data

(c) Question voting prediction on SO data

(d) Answer voting prediction on SO data

Fig. 2. The effectiveness comparisons. Lower is better. The proposed algorithms outperform both SVR and KRR.



(a) RMSE results when we set $\lambda = 0.1$ and vary parameter θ on SO data

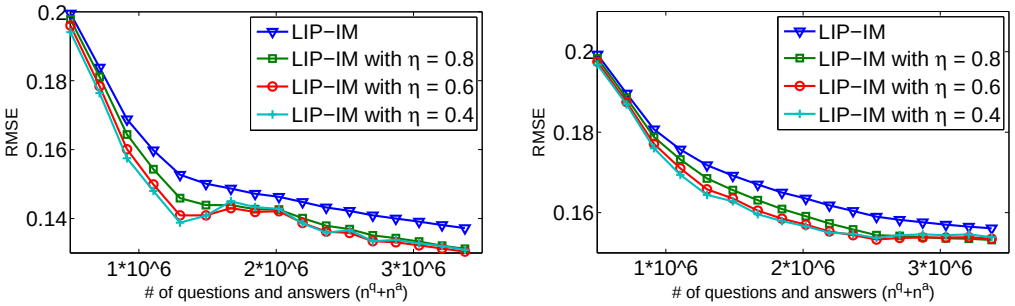
(b) RMSE results when we set $\eta = 0.1$ and vary parameter λ on SO data

Fig. 3. The sensitivity evaluations. Our algorithms are robust wrt both parameters.

TABLE 5
Feature selection results of LIP-MS.

Feature	Question Coefficient	Answer Coefficient
Questioner's reputation at question creation time	13.3	9.9
# of questioner's previous questions at question creation time	-31.0	-37.5
The length of the question body	-15.9	-34.3
The length of the question title	3.1	2.0
# of answers received in 24 hours after question creation	46.1	67.5
# of comments received in 24 hours after question creation	-8.4	-8.4
Answerer's reputation at answer creation time	0.0	19.7
# of answerer's previous answers at answer creation time	-62.5	-62.5
The length of the answer	40.0	47.2
# of comments received in 24 hours after question creation	-13.3	7.3

qualitative analysis about the feature selection results of LIP-MS. Since our goal here is to check the selected features, we set $\theta = \lambda_1 = \lambda_2 = 0.1$ and the elastic net parameters to 0.2 (i.e., $\rho_1 = \rho_2 = 0.2$) for simplicity. The results of the contextual features on the SO data are shown in Table 5.



(a) Question voting prediction on SO data

(b) Answer voting prediction on SO data

Fig. 4. The effectiveness results of LIP-IMF. Lower is better. Fading the effects of old examples can help to improve the accuracy of voting prediction.

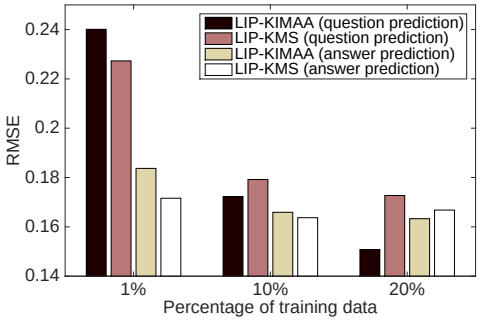


Fig. 5. The effectiveness results of applying the example selection by LIP-KMS. Lower is better. Selecting a small subset of questions and answers by LIP-KMS can achieve comparable results with LIP-KIMAA.

There are several interesting observations from the table. First, the reputations of questioners and answerers are indicative features; however, answerer's reputation has little impact on question voting prediction. Second, the number of previous questions/answers plays negative roles. The possible reason is that questioners/answerers tend to pay little attention to the quality of their questions/answers if they ask/answer a lot of questions/answers. Third, the length of question is a negative feature while the length of answer is a positive feature. This means that people may be impatient for long questions, while they may be more patient on the answers if they encounter the same questions. Fourth, the number of answers received in 24 hours is an important feature for both question and answer prediction. This feature indicates the popularity/impact of the question-answer thread. Finally, the number of comments is a negative feature for question prediction, and a positive feature for answer prediction. The possible reason is that a question with many comments may be of low-quality (e.g., the expression is unclear), while comments for answers may indicate agreement.

In addition to the contextual features in Table 5, we also have many content features derived from the "bag of words" model. We found that the coefficients of over 85% words are zeros. This result indicates that we may use only a very small set of content features to speedup our algorithms.

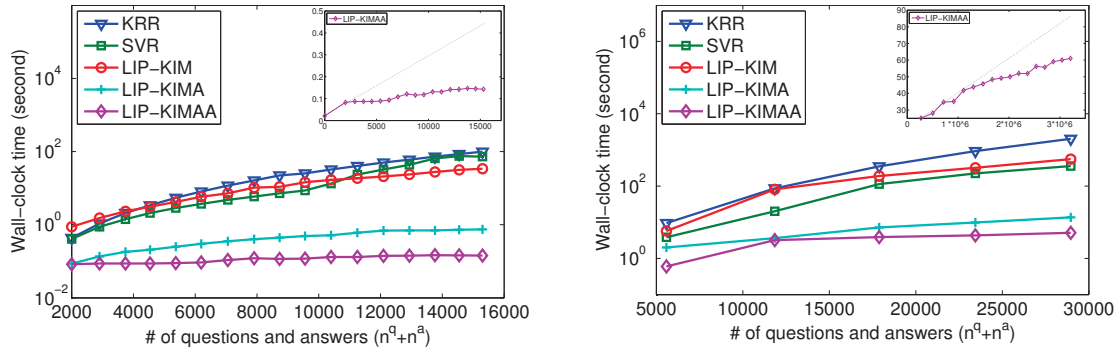
(F) Effectiveness Results by Example Selection. Next, we evaluate the effects of example selection by the LIP-KMS algorithm. The results on the SO data are shown in Fig. 5 where we keep 1%, 10%, and 20% training data, respectively. Here, we first apply LIK-KMS to select a subset of questions/answers and then run LIP-KIMA on the selected examples (referred to as 'LIP-KMS' in the figure). This experimental setting is similar to LIP-KIMAA: instead of filtering out some newly arrived examples (LIP-KIMAA), LIP-KMS keeps only a small subset of existing questions/answers. Therefore, we also plot the results of LIP-KIMAA for comparison. For simplicity, we set the elastic net parameter of LIP-KMS to 0.3 (i.e., $\rho = 0.3$[4]). As we can see from the figure, selecting a small subset of questions and answers by LIP-KMS can achieve even better results than LIP-KIMAA when the data size is small. This result indicates that LIP-KMS can identify useful examples for the prediction task. When the data size becomes large, LIP-KIMAA performs better. This is due to the fact that LIP-KMS always selects a small subset of questions and answers (e.g., less than two thousands), while LIP-KIMAA may select much more examples. More examples can be kept by tuning the elastic net parameter in LIP-KMS. However, keeping a small set of examples may potentially improve the efficiency results as we will later show in Fig. 7.

5.3 Efficiency Results

(G) The Speed Comparisons. For efficiency, we first compare the proposed algorithms (i.e., LIP-KIM, LIP-KIMA, and LIP-KIMAA) with KRR and SVR on SO and Math data in Fig. 6. Notice that the y-axis is in log scale. In Fig. 6, we also plot the results of LIP-KIMAA with y-axis in linear scale in the upper-right corner. We only report the results by LIP-KIMAA there because it is the only algorithm that can handle the entire SO data set.

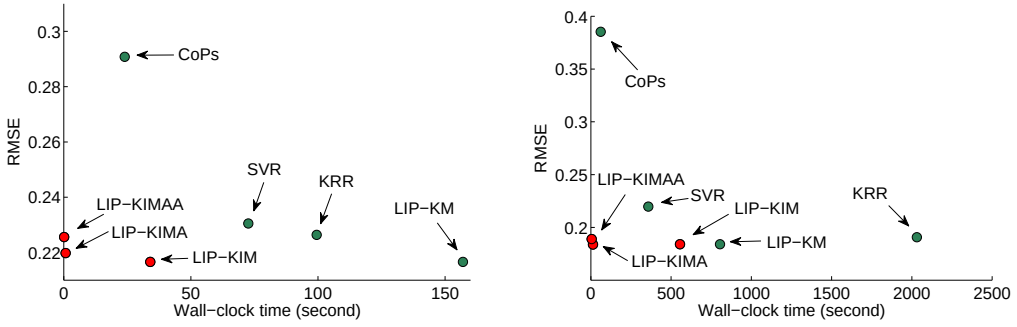
As we can see from the figure, our LIP-KIMA and LIP-KIMAA are much faster than the other algorithms. In the upper-right corner, we can observe that the LIP-KIMAA scales *sub-linearly* wrt the total number of questions and

4. This parameter can be tuned to determine the size of selected questions and answers.



(a) Math data (b) SO data

Fig. 6. The speed comparisons. The proposed LIP-KIMA and LIP-KIMAA are much faster. Furthermore, LIP-KIMAA scales sub-linearly (in the upper-right corner).



(a) Answer voting prediction on Math data (b) Answer voting prediction on SO data

Fig. 8. The quality-speed balance-off. The proposed LIP-KIMA and LIP-KIMAA achieve a good balance between the prediction quality and the efficiency (in the left-bottom corner). Best viewed in color.

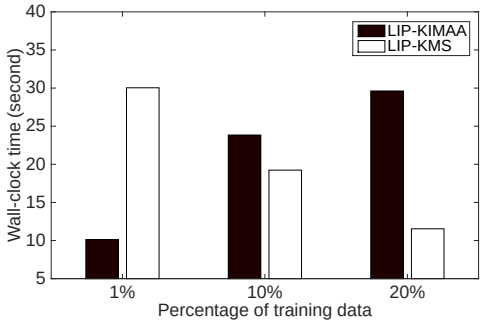


Fig. 7. The efficiency results of applying the example selection by LIP-KMS. Using LIP-KMS for example selection may be even faster than LIP-KIMAA.

answers. For instance, it only requires about 60 seconds when there are more than 3,000,000 questions and answers. In contrast, KRR requires more than 2,000 seconds when the size of the training set is about 30,000.

(H) Efficiency Results by Example Selection. Next, we evaluate the efficiency results of applying LIK-KMS before running LIP-KIMA. The results on the SO data are shown in Fig. 7, where we also plot LIP-KIMAA for comparison. As we can see from the figure, using LIP-KMS for example selection gets faster than LIP-KIMAA when the data size becomes larger. The reason is that the computation time is proportional to the number of selected questions and answers, and LIP-KMS selects a smaller number of questions and answers than LIP-KIMAA.

(I) Quality-Speed Balance-off. Finally, we study the quality-speed balance-off of different algorithms in Fig. 8. In Fig. 8, we show the answer prediction results only. Similar results are observed in question prediction, and we omit the results for brevity. In Fig. 8, we plot the RMSE on the y-axis and the wall-clock time on the x-axis. We also plot the results of the linear co-prediction method CoPs [3] and LIP-KM. Ideally, we want an algorithm sitting in the left-bottom corner. As we can see, both our LIP-KIMA and LIP-KIMAA are in the left-bottom corner. For example, for answer voting prediction on the SO data, compared with SVR, LIP-KIMAA is 70x faster in wall-clock time and 14.0% better in RMSE. Overall, we recommend LIP-KIMAA in practice.

6 Related Work

In this section, we briefly review related work including mining CQA sites and mining stream data.

Mining CQA Sites: There is a large body of existing work on mining CQA sites. For example, Li et al. [18] aim to predict question quality, which is defined as the combination of user attention, answer attempts and the arrival speed of the best answer. Ravi et al. [19] also study

the question quality which is combined by voting score and pageviews. Jeon et al. [20] and Suryanto et al. [21] evaluate the usefulness of answer quality and incorporate it to improve retrieval performance. To predict the quality of both questions and answers, Agichtein et al. [22] develop a graph-based model to catch the relationships among users, Li et al. [23] adopt the co-training approach to employ both question features and answer features, and Bian et al. [24] propose to propagate the labels through user-question-answer graph so as to tackle the sparsity problem where only a small number of questions/answers are labeled. Recently, Anderson et al. [25] propose to predict the long-lasting value (i.e., the pageviews) of a question and its answers. Dror et al. [26] aim to predict whether a question will be answered or not. How to predict the answer that the questioner will probably choose as the accepted answer is also well studied [27], [28], [29], [30]. Overall, our work differs from these existing work at the methodology level. While most of the existing work treats the prediction problem as a single, and/or linear, and/or static problem, we view the problem from a comprehensive perspective and propose to incorporate all these important aspects into the prediction models.

Mining Stream Data: From the dynamic aspect, our LIP problem is related to stream mining [31] and time-series mining [32]. The main focus of existing stream/time-series mining work is on pattern discovery, clustering, and classification tasks. Chen et al. [33] and Ikonomovska et al. [34] study the regression problem in data streams; however, they still focus on a single and linear prediction problem. Several researchers also consider the non-linear and dynamic aspects in regression problem [6], [35]. Different from these existing work, we consider the coupling between questions and answers, and propose approximation methods to speed-up and scale-up the computation.

Other Related Work: There are several pieces of interesting work that are remotely related to our work. Liu et al. [36] propose the problem of CQA site searcher satisfaction, i.e., whether or not the answer in a CQA site satisfies the information searcher using the search engines. Shtok et al. [37] attempt to answer certain new questions by existing answers. Zhou et al. [38] aim to find similar questions for a given query. Zhou et al. [39] propose to identify whether a user is asking a subjective question or not. Wei et al. [40] discard user biases to re-rank the voting scores of answers. Omari et al. [41] propose to provide a set of diverse and novel answers for questions. Sung et al. [42] aim to detect the potentially contributive users from recently-joined users. The question routing problem (e.g., how to route the right question to the right answerer) is also an active research area [43], [44], [45].

7 Conclusions

In this aricle, we have proposed a family of algorithms to comprehensively and efficiently predict the voting scores of questions/answers in CQA sites. In particular, some of the proposed algorithms (LIP-KIM, LIP-KIMA, and LIP-KIMAA) can capture three key aspects (non-linearity, coupling, and dynamics) that matter with the voting score of a post, while others can handle the special cases when only a fraction of the three aspects are prominent. In terms of computation efficiency, some algorithms (LIP-IM, LIP-IMF, LIP-KIA, LIP-KIMAA, and LIP-KIMAA) enjoy linear, sub-linear, or even constant scalability. The proposed algorithms are also able to fade the effects of old examples (LIP-IMF), and select a subset of features/examples (LIP-MS and LIP-KMS). We analyze our algorithms in terms of optimality, correctness, and complexity, and reveal the intrinsic relationships among different algorithms. We conduct extensive experimental evaluations on two real data sets to demonstrate the effectiveness and efficiency of our approaches.

Acknowledgments

We would like to thank the reviewers for their constructive comments.

This work is supported by the National Key Research and Development Program of China (No. 2016YFB1000802), National 863 Program of China (No. 2015AA01A203), National Natural Science Foundation of China (No. 61672274, 61690204), the Collaborative Innovation Center of Novel Software Technology and Industrialization. Hanghang Tong is partially supported by NSF (IIS-1651203), DTRA (HDTRA1-16-0017), ARO (W911NF-16-1-0168), NIH (R01LM011986), RUTC2 (49997-33 25) and a Baidu gift.

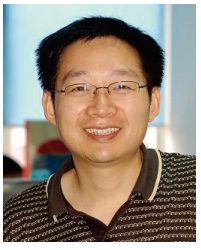
References

[1] T. Osbourn, "Getting the most out of the web," *Software, IEEE*, vol. 28, no. 1, pp. 96–96, 2011.
[2] Y. Yao, H. Tong, T. Xie, L. Akoglu, F. Xu, and J. Lu, "Joint voting prediction for questions and answers in cqa," in *ASONAM*, 2014.
[3] ——, "Want a good answer? ask a good question first!" *arXiv preprint arXiv:1311.6876*, 2013.
[4] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *ICML*, 1998, pp. 515–521.
[5] S. S. Haykin, *Adaptive filter theory*, 2005.
[6] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.
[7] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, vol. 68, no. 3, pp. 337–404, 1950.
[8] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.
[9] P. Drineas and M. W. Mahoney, "On the nyström method for approximating a gram matrix for improved kernel-based learning," *The Journal of Machine Learning Research*, vol. 6, pp. 2153–2175, 2005.
[10] G. Golub and C. Van Loan, "Matrix computations," 1996.
[11] W. W. Piegorsch and G. Casella, "Inverting a sum of matrices," *SIAM Review*, vol. 32, no. 3, pp. 470–470, 1990.
[12] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.
[13] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 1, pp. 91–108, 2005.
[14] X. Chen, Q. Lin, S. Kim, J. G. Carbonell, and E. P. Xing, "Smoothing proximal gradient method for general structured sparse regression," *The Annals of Applied Statistics*, pp. 719–752, 2012.
[15] J. Liu, L. Yuan, and J. Ye, "An efficient algorithm for a class of fused lasso problems," in *KDD*. ACM, 2010, pp. 323–332.

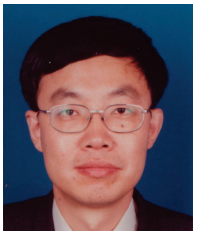
[16] R. Collobert and S. Bengio, "Svmtorch: Support vector machines for large-scale regression problems," *The Journal of Machine Learning Research*, vol. 1, pp. 143–160, 2001.
[17] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, p. 27, 2011.
[18] B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak, "Analyzing and predicting question quality in community question answering services," in *WWW*. ACM, 2012, pp. 775–782.
[19] S. Ravi, B. Pang, V. Rastogi, and R. Kumar, "Great question! question quality in community q&a," in *ICWSM*, 2014, pp. 426–435.
[20] J. Jeon, W. Croft, J. Lee, and S. Park, "A framework to predict the quality of answers with non-textual features," in *SIGIR*. ACM, 2006, pp. 228–235.
[21] M. Suryanto, E. Lim, A. Sun, and R. Chiang, "Quality-aware collaborative question answering: methods and evaluation," in *WSDM*. ACM, 2009, pp. 142–151.
[22] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne, "Finding high-quality content in social media," in *WSDM*. ACM, 2008, pp. 183–194.
[23] B. Li, Y. Liu, and E. Agichtein, "Cocqa: co-training over questions and answers with an application to predicting question subjectivity orientation," in *EMNLP*, 2008, pp. 937–946.
[24] J. Bian, Y. Liu, D. Zhou, E. Agichtein, and H. Zha, "Learning to recognize reliable users and content in social media with coupled mutual reinforcement," in *WWW*. ACM, 2009, pp. 51–60.
[25] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Discovering value from community activity on focused question answering sites: a case study of stack overflow," in *KDD*. ACM, 2012, pp. 850–858.
[26] G. Dror, Y. Maarek, and I. Szpektor, "Will my question be answered? predicting question answerability in community question-answering sites," in *ECML/PKDD*, 2013, pp. 499–514.
[27] Y. Liu, J. Bian, and E. Agichtein, "Predicting information seeker satisfaction in community question answering," in *SIGIR*. ACM, 2008, pp. 483–490.
[28] C. Shah and J. Pomerantz, "Evaluating and predicting answer quality in community qa," in *SIGIR*, 2010, pp. 411–418.
[29] L. Adamic, J. Zhang, E. Bakshy, and M. Ackerman, "Knowledge sharing and yahoo answers: everyone knows something," in *WWW*. ACM, 2008, pp. 665–674.
[30] Q. Tian, P. Zhang, and B. Li, "Towards predicting the best answers in community-based question-answering services," in *ICWSM*, 2013.
[31] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *ACM Sigmod Record*, vol. 34, no. 2, pp. 18–26, 2005.
[32] T.-C. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.
[33] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, "Multi-dimensional regression analysis of time-series data streams," in *VLDB*, 2002, pp. 323–334.
[34] E. Ikonomovska, J. Gama, and S. Džeroski, "Learning model trees from evolving data streams," *Data mining and knowledge discovery*, vol. 23, no. 1, pp. 128–168, 2011.
[35] B. Pan, J. J. Xia, P. Yuan, J. Gateno, H. H. Ip, Q. He, P. K. Lee, B. Chow, and X. Zhou, "Incremental kernel ridge regression for the prediction of soft tissue deformations," in *Medical Image Computing and Computer-Assisted Intervention*. Springer, 2012, pp. 99–106.
[36] Q. Liu, E. Agichtein, G. Dror, E. Gabrilovich, Y. Maarek, D. Pelleg, and I. Szpektor, "Predicting web searcher satisfaction with existing community-based answers," in *SIGIR*, 2011, pp. 415–424.
[37] A. Shtok, G. Dror, Y. Maarek, and I. Szpektor, "Learning from the past: answering new questions with past answers," in *WWW*, 2012, pp. 759–768.
[38] G. Zhou, T. He, J. Zhao, and P. Hu, "Learning continuous word embedding with metadata for question retrieval in community question answering," in *ACL*, 2015, pp. 250–259.
[39] T. Zhou, X. Si, E. Chang, I. King, and M. Lyu, "A data-driven approach to question subjectivity identification in community question answering," in *AAAI*, 2012.
[40] X. Wei, H. Huang, C.-Y. Lin, X. Xin, X. Mao, and S. Wang, "Re-ranking voting-based answers by discarding user behavior biases," in *AAAI*, 2015, pp. 2380–2386.
[41] A. Omari, D. Carmel, O. Rokhlenko, and I. Szpektor, "Novelty based ranking of human answers for community questions," in *SIGIR*, 2016, pp. 215–224.
[42] J. Sung, J.-G. Lee, and U. Lee, "Booming up the long tails: Discovering potentially contributive users in community-based question answering services," in *ICWSM*, 2013.
[43] Y. Zhou, G. Cong, B. Cui, C. Jensen, and J. Yao, "Routing questions to the right users in online communities," in *ICDE*. IEEE, 2009, pp. 700–711.
[44] D. Horowitz and S. Kamvar, "The anatomy of a large-scale social search engine," in *WWW*. ACM, 2010, pp. 431–440.
[45] I. Szpektor, Y. Maarek, and D. Pelleg, "When relevance is not enough: promoting diversity and freshness in personalized question recommendation," in *WWW*, 2013, pp. 1249–1260.

Yuan Yao is currently an assistant researcher at the Department of Computer Science and Technology, Nanjing University, China. He received the B.Sc. and Ph.D. degree in computer science from Nanjing University, China, in 2009 and 2015. His research interest includes social media mining and software repository mining.

Hanghang Tong is currently an assistant professor at School of Computing, Informatics, and Decision Systems Engineering, Arizona State University. Before that, he was a research staff member at IBM T.J. Watson Research Center and a Post-doctoral fellow in Carnegie Mellon University. He received his M.Sc and Ph.D. degree from Carnegie Mellon University in 2008 and 2009, both majored in Machine Learning. His research interest is in large scale data mining for graphs and multimedia. His research has been funded by NSF, DARPA and ARL. He has received several awards, including best paper award in CIKM 2012, SDM 2008 and ICDM 2006. He has published over 60 referred articles and more than 20 patents. He has served as a program committee member in top data mining, databases and artificial intelligence venues.



Feng Xu received the B.S. and M.S. degrees from Hohai University in 1997 and 2000, respectively. He received his Ph.D. degree from Nanjing University in 2003. He is a professor in the Department of Computer Science and Technology at Nanjing University. His research interests include trust management, trusted computing, and software reliability.



Jian Lu received his B.Sc., M.Sc. and Ph.D. degrees in Computer Science from Nanjing University, P.R. China. He is currently a professor in the Department of Computer Science and Technology and the Director of the State Key Laboratory for Novel Software Technology at Nanjing University. He serves on the Board of the International Institute for Software Technology of the United Nations University (UNU-IIST). He also serves as the director of the Software Engineering Technical Committee of the China Computer Federation. His research interests include software methodologies, software automation, software agents, and middleware systems.