Predicting Long-Term Impact of CQA Posts: A Comprehensive Viewpoint

Yuan Yao

Joint work with
Hanghang Tong, Feng Xu, and Jian Lu

Aug 24-27, KDD 2014
Roadmap

- Background and Motivations
- Modeling Multi-aspect
- Computation Speedup
- Empirical Evaluations
- Conclusions
Roadmap

- Background and Motivations
- Modeling Multi-aspect
- Computation Speedup
- Empirical Evaluations
- Conclusions
Long-Term Impact

What is the difference between `#include <filename>` and `#include "filename"`?

In the C and C++ programming languages, what is the difference between using angle brackets and using quotes in an `include` statement, as follows?

1. `#include <filename>`
2. `#include "filename"`

Q: How many users will find it beneficial?

Which one is faster and why? Array vs Link List [closed]

Which one is faster and why? (in C/C++)

1. Array
2. Link List.

If we just want to iterate in for loop and print it.
Challenges

- Q: Why not off-the-shell data mining algorithms?

- Challenge 1: Multi-aspect
  - C1.1. Coupling between questions and answers
  - C1.2. Feature non-linearity
  - C1.3. Posts dynamically arrive

- Challenge 2: Efficiency
C1.1 Coupling

Strong positive correlation!

[Yao+@ASONAM’14]


C1.1 Coupling

**LIP-M**: [Yao+@ASONAM’14]

1. **Question prediction**
   \[
   \min_{\alpha^q, \alpha^a} \sum_{i=1}^{n^q} (F^q(i,:\alpha^q - Y^q(i))^2 + \sum_{i=1}^{n^a} (F^a(i,:\alpha^a - Y^a(i))^2 + \theta \sum_{i=1}^{n^q} (F^q(i,:)\alpha^q - M(i,:F^a\alpha^a))^2
   \]

2. **Answer prediction**
   \[
   \sum_{i=1}^{n^a} (F^a(i,:)\alpha^a - Y^a(i))^2 + \lambda(\|\alpha^q\|_2^2 + \|\alpha^a\|_2^2)
   \]

3. **Voting Consistency**

4. **Regularization**
C1.2 Non-linearity

The kernel trick (e.g., SVM)

- Mercer kernel
  \[ \kappa(F(i,:), F(j,:)) = \langle \phi(F(i,:)), \phi(F(j,:)) \rangle \]

- Kernel matrix as new feature matrix
  \[ K^q(i, j) = \kappa(F^q(i,:), F^q(j,:)) \]
  \[ K^a(i, j) = \kappa(F^a(i,:), F^a(j,:)) \]
C1.3 Dynamics

Solution: recursive least squares regression

This Paper

- **Q1:** how to comprehensively capture the multi-aspect in one algorithm?
  - Coupling, non-linearity, and dynamics

- **Q2:** how to make the long-term impact prediction algorithm efficient?
Roadmap

- Background and Motivations
- Modeling Multi-aspect
- Computation Speedup
- Empirical Evaluations
- Conclusions
Modeling Non-linearity

- Basic Idea: kernelize LIP-M
- Details - LIP-KM:

  1. **Question prediction**

  $$\min_{\beta^q, \beta^a} \sum_{i=1}^{n^q} (K^q(i,:)\beta^q - Y^q(i))^2 + \theta \sum_{i=1}^{n^q} (K^q(i,:)\beta^q - M(i,:)K^a\beta^a)^2$$

  2. **Answer prediction**

  $$\sum_{i=1}^{n^a} (K^a(i,:)\beta^a - Y^a(i))^2 + \lambda ((\beta^a)'K^q\beta^q + (\beta^a)'K^a\beta^a)$$

  3. **Voting Consistency**

  - Closed-form solution:

  $$\beta = \begin{bmatrix} (\theta + 1)K^q + \lambda I \\ -\theta M'K^q \\ K^a + \theta M'MK^a + \lambda I \end{bmatrix}^{-1} \begin{bmatrix} Y^q \\ Y^a \end{bmatrix}$$

  - **Complexity:** $O(n^3)$
Modeling Dynamics

- Basic idea: recursively update LIP-KM
- Details - LIP-KIM:

\[ S_t = \begin{bmatrix} \frac{1}{2}K_t^q + \lambda I & -\theta M_t K_t^a \\ -\theta M_t' K_t^q & K_t^a + \theta M_t' M_t K_t^a + \lambda I \end{bmatrix} \]

\[ \beta_t = S_t^{-1} \begin{bmatrix} Y_t^q \\ Y_t^a \end{bmatrix} \quad \beta_{t+1} = S_{t+1}^{-1} \begin{bmatrix} Y_{t+1}^q \\ Y_{t+1}^a \end{bmatrix} \]

\[ \beta_{t+1} = E_1 \left[ \beta_t + S_t^{-1} S_1 D (S_2 \beta_t - [y_{t+1}^q ; y_{t+1}^a]) - S_3^{-1} S_2 (\beta_t + S_t^{-1} S_1 DS_2 \beta_t) + D[y_{t+1}^q ; y_{t+1}^a] \right] \]

- Complexity: \( O(n^3) \rightarrow O(n^2) \)
Roadmap

- Background and Motivations
- Modeling Multi-aspect
- Computation Speedup
- Empirical Evaluations
- Conclusions
Approximation Method (1)

- Basic idea: compress the kernel matrix
- Details - LIP-KIMA:
  - 1) Separate decomposition
  - 2) Make decomposition reusable
  - 3) Apply decomposition on LIP-KIM

\[
K_{t+1}^q = S_{t+1}^{-1} = \begin{bmatrix} K_t^q & 0 \\ 0 & K_{t+1}^a \end{bmatrix} \approx (I + \theta G) \begin{bmatrix} K_t^q & 0 \\ 0 & K_{t+1}^a \end{bmatrix}
\]

\[
= I + \lambda UU' + \theta GGU'
\]

\[
= I + \lambda AB
\]

\[
\beta_{t+1} = \frac{1}{\lambda} (I - A(\lambda I + BA)^{-1}B) \begin{bmatrix} Y_t^q \\ Y_a^t \end{bmatrix}
\]

- Complexity: \(O(n^2) \rightarrow O(n)\)
Approximation Method (2)

- Basic idea: filter less informative examples
- Details - LIP-KIMAA:

$$\begin{align*}
\text{Existing examples} &: (x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t) \\
\text{Current model} &: \text{Model}_t \\
\text{New examples} &: (x_{t+1}, y_{t+1}) \\
\text{New model} &: \text{Model}_{t+1}
\end{align*}$$

- Complexity: $O(n) \rightarrow <O(n)$
Summary

LIP-KIMAA <O(n)
LIP-KIMA O(n)
LIP-KIM O(n^2)
LIP-KM O(n^3)
LIP-KI (Recursive Kernel Ridge Regression)
LIP-I (Recursive Ridge Regression)
LIP-M (CoPs)
LIP-IM

K: Non-linearity
I: Dynamics
M: Coupling
A: Approximation

Non-linearity
Coupling
Ridge Regression
Dynamics
Roadmap

- Background and Motivations
- Modeling Multi-aspect
- Computation Speedup
- Empirical Evaluations
- Conclusions
Experiment Setup

- **Datasets** (http://blog.stackoverflow.com/category/cc-wiki-dump/)
  - Stack Overflow, Mathematics Stack Exchange

- **Features**
  - Content (bag-of-words) & contextual features
Evaluation Objectives

- **O1: Effectiveness**
  - How accurate are the proposed algorithms for long-term impact prediction?

- **O2: Efficiency**
  - How scalable are the proposed algorithms?
Effectiveness Results

Comparisons with existing models.

Our methods

(better)

Comparisons with existing models.
Efficiency Results

The speed comparisons.

- KRR
- SVR
- LIP-KIM
- LIP-KIMA
- LIP-KIMAA

LIP-KIMAA ↓ (sub-linear)

(better)

The speed comparisons.

Ours
Quality-Speed Balance-off

Our methods
(better)

RMSE

Wall-clock time (second)

CoPs
LIP-KIMAA
LIP-KIMA
LIP-KIM
SVR
KRR
LIP-KM
Roadmap

- Background and Motivations
- Modeling Multi-aspect
- Computation Speedup
- Empirical Evaluations
- Conclusions
Conclusions

A family of algorithms for long-term impact prediction of CQA posts

- Q1: how to capture coupling, non-linearity, and dynamics?
  - A1: voting consistency + kernel trick + recursive updating

- Q2: how to make the algorithms scalable?
  - A2: approximation methods

Empirical Evaluations

- Effectiveness: up to 35.8% improvement
- Efficiency: up to 390x speedup and sub-linear scalability
Authors: Yuan Yao, Hanghang Tong, Feng Xu, and Jian Lu

Yuan Yao, yyao@smail.nju.edu.cn

Thanks!

Q&A