Fast Sketching of Polynomial Kernels of Polynomial Degree

ICML 2021 speaker: Lichen Zhang (CMU)

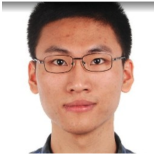
June 16, 2021

Zhao Song

Princeton & IAS

David P. Woodruff

CMU



Zheng Yu

Princeton

Motivation

- We consider the problem of efficiently computing a kernel matrix.

Motivation

- We consider the problem of efficiently computing a kernel matrix.
- Suppose there are n data points $\{x_i\}_{i=1}^n$, each of dimension d. The goal is to compute the $n \times n$ PSD kernel matrix K where each entry $K_{i,j} = k(x_i, x_j)$ under certain kernel function k.

Motivation

- We consider the problem of efficiently computing a kernel matrix.
- Suppose there are n data points $\{x_i\}_{i=1}^n$, each of dimension d. The goal is to compute the $n \times n$ PSD kernel matrix K where each entry $K_{i,j} = k(x_i, x_j)$ under certain kernel function k.
- Additionally, we consider the high-dimensional, or over-parametrized setting, where $d \gg n$. This is of particular interest in NLP, biology and training over-parametrized neural networks.

Polynomial kernel

- Instead of designing algorithm for any kernel, we focus on polynomial kernels.

Polynomial kernel

- Instead of designing algorithm for any kernel, we focus on polynomial kernels.
- Advantage: many popular kernels can be well-approximated using their Taylor expansions, e.g., Gaussian kernel, arc-cosine kernel (Cho and Saul, 09) and neural-tangent kernel (Jacot, Gabriel and Hongler, 18).

Polynomial kernel

- Instead of designing algorithm for any kernel, we focus on polynomial kernels.
- Advantage: many popular kernels can be well-approximated using their Taylor expansions, e.g., Gaussian kernel, arc-cosine kernel (Cho and Saul, 09) and neural-tangent kernel (Jacot, Gabriel and Hongler, 18).
- Polynomial kernel of degree q is defined as $P_q(x, y) = \langle x, y \rangle^q$.

Intuition

- Instead of computing P_q directly, we compute $P_q = \Phi^\top \Phi$, where $\Phi \in \mathbb{R}^{d^q \times n}$ is defined using a *lifting function* $\phi : \mathbb{R}^d \to \mathbb{R}^{d^q}$ with $\phi(x)_{i_1, i_2, \ldots, i_q} = x^{\otimes q} = x_{i_1} x_{i_2} \ldots x_{i_q}$ for $i_1, i_2, \ldots, i_q \in \{1, 2, \ldots, d\}$.

Intuition

- Instead of computing P_q directly, we compute $P_q = \Phi^\top \Phi$, where $\Phi \in \mathbb{R}^{d^q \times n}$ is defined using a *lifting function* $\phi : \mathbb{R}^d \to \mathbb{R}^{d^q}$ with $\phi(x)_{i_1, i_2, \ldots, i_q} = x^{\otimes q} = x_{i_1} x_{i_2} \ldots x_{i_q}$ for $i_1, i_2, \ldots, i_q \in \{1, 2, \ldots, d\}$.
- $\Phi = X^{\otimes q} \in \mathbb{R}^{d^q \times n}$.

Intuition

- Instead of computing P_q directly, we compute $P_q = \Phi^\top \Phi$, where $\Phi \in \mathbb{R}^{d^q \times n}$ is defined using a *lifting function* $\phi : \mathbb{R}^d \to \mathbb{R}^{d^q}$ with $\phi(x)_{i_1, i_2, \ldots, i_q} = x^{\otimes q} = x_{i_1} x_{i_2} \ldots x_{i_q}$ for $i_1, i_2, \ldots, i_q \in \{1, 2, \ldots, d\}$.
- $\Phi = X^{\otimes q} \in \mathbb{R}^{d^q \times n}$.
- Goal: approximate $X^{\otimes q}$ efficiently without explicitly forming it. Typically, it involves looking for a matrix $\Pi \in \mathbb{R}^{s \times d^q}$ with $s \ll d^q$ and forming $\Pi X^{\otimes q}$ very fast.

Prior works

- Prior methods:

Prior works

- Prior methods:
  - Random Fourier features (Rahimi and Recht, 07);

Prior works

- Prior methods:
  - Random Fourier features (Rahimi and Recht, 07);
  - Nystrom method (Williams and Seeger, 01; Musco and Musco, 17);

Prior works

- Prior methods:
  - Random Fourier features (Rahimi and Recht, 07);
  - Nystrom method (Williams and Seeger, 01; Musco and Musco, 17);
  - Oblivious sketching (Ahle, Kapralov, Knudsen, Pagh, Velingker, Woodruff and Zandieh, 20);

Prior works

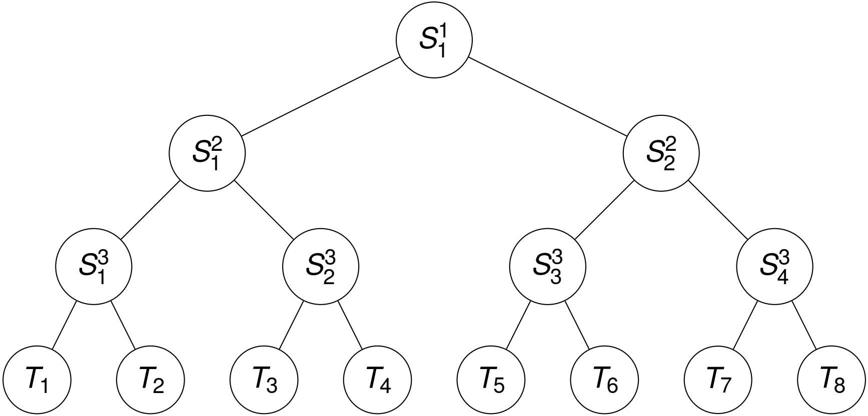
- Prior methods:
  - Random Fourier features (Rahimi and Recht, 07);
  - Nystrom method (Williams and Seeger, 01; Musco and Musco, 17);
  - Oblivious sketching (Ahle, Kapralov, Knudsen, Pagh, Velingker, Woodruff and Zandieh, 20);
  - Adaptive Leverage score sampling (Woodruff and Zandieh, 20).

Prior works

- Prior methods:
  - Random Fourier features (Rahimi and Recht, 07);
  - Nystrom method (Williams and Seeger, 01; Musco and Musco, 17);
  - Oblivious sketching (Ahle, Kapralov, Knudsen, Pagh, Velingker, Woodruff and Zandieh, 20);
  - Adaptive Leverage score sampling (Woodruff and Zandieh, 20).
- State-of-the-art are oblivious sketching and adaptive leverage score sampling.

Oblivious sketching

Oblivious sketching: view the computation of $x^{\otimes q}$ as a binary tree with q nodes.



Oblivious sketching

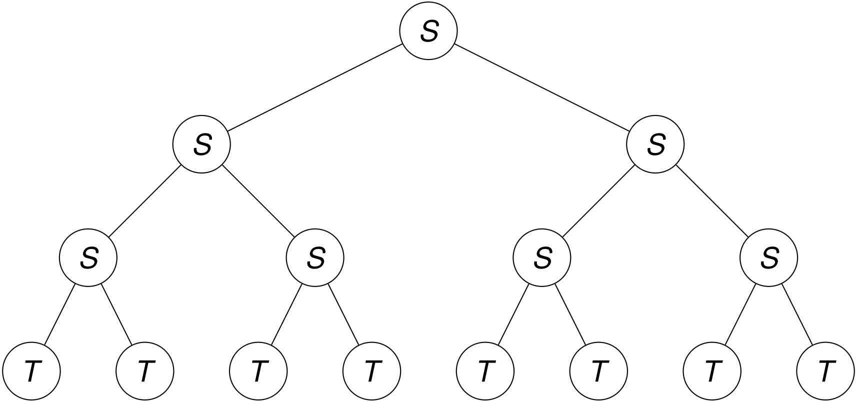
- Advantage: algorithm enjoys a great amount of independence and randomness, therefore has strong guarantee.

Oblivious sketching

- Advantage: algorithm enjoys a great amount of independence and randomness, therefore has strong guarantee.
- Can we reduce the amount of randomness and improve the running time of oblivious sketching?

Our approach: constructing the tree with limited randomness

Instead of using q different random sketches, we just use *2* of them.



Our approach: constructing the tree with limited randomness

- Advantage:

Our approach: constructing the tree with limited randomness

- Advantage:
  - Each layer of the tree only needs to compute one node.

Our approach: constructing the tree with limited randomness

- Advantage:
  - Each layer of the tree only needs to compute one node.
  - Only need to compute $O(\log q)$ nodes using repeated squaring — critical in approximating kernels where q is large.

Our approach: constructing the tree with limited randomness

- Advantage:
  - Each layer of the tree only needs to compute one node.
  - Only need to compute $O(\log q)$ nodes using repeated squaring — critical in approximating kernels where q is large.
  - Depends nearly linear on nd.

Our approach: constructing the tree with limited randomness

- Advantage:
  - Each layer of the tree only needs to compute one node.
  - Only need to compute $O(\log q)$ nodes using repeated squaring — critical in approximating kernels where q is large.
  - Depends nearly linear on nd.
- However, since we use only a constant amount of randomness, the guarantee of $\Pi X^{\otimes q}$ is weaker than oblivious sketching: we can only spectrally approximate $X^{\otimes q}$.

Our approach: constructing the tree with limited randomness

- Advantage:
  - Each layer of the tree only needs to compute one node.
  - Only need to compute $O(\log q)$ nodes using repeated squaring — critical in approximating kernels where q is large.
  - Depends nearly linear on nd.
- However, since we use only a constant amount of randomness, the guarantee of $\Pi X^{\otimes q}$ is weaker than oblivious sketching: we can only spectrally approximate $X^{\otimes q}$.
- Nevertheless, it can be combined with other sketching techniques to achieve the overall strong guarantee of approximate matrix product as well and in a faster way.

Applications

- Our polynomial kernel approximation algorithm can be applied to various settings, such as approximate a Gaussian kernel, our algorithm gives the fastest result in over-parametrized and unregularized setting.

Applications

- Approximate a class of slow-decaying kernels. We can classify kernels based on the coefficients of their Taylor expansions.

Applications

- Approximate a class of slow-decaying kernels. We can classify kernels based on the coefficients of their Taylor expansions.
  - For Gaussian kernels, the coefficients are roughly $1/n!$, this means to approximate it using polynomial kernel, we only need a poly-logarithmic number of terms.

Applications

- Approximate a class of slow-decaying kernels. We can classify kernels based on the coefficients of their Taylor expansions.
  - For Gaussian kernels, the coefficients are roughly $1/n!$, this means to approximate it using polynomial kernel, we only need a poly-logarithmic number of terms.
  - For a broader class of kernels with coefficients being $1/n^c$ where $c > 1$, more terms are needed.

Applications

- Approximate a class of slow-decaying kernels. We can classify kernels based on the coefficients of their Taylor expansions.
  - For Gaussian kernels, the coefficients are roughly $1/n!$, this means to approximate it using polynomial kernel, we only need a poly-logarithmic number of terms.
  - For a broader class of kernels with coefficients being $1/n^c$ where $c > 1$, more terms are needed.
  - We propose a novel sampling scheme, where we exactly compute the first s terms in Taylor expansion, then sample s from the remaining terms. By carefully balancing the number of terms in exact computation and sampling, we achieve a faster runtime.

Applications

- Approximate a class of slow-decaying kernels. We can classify kernels based on the coefficients of their Taylor expansions.
  - For Gaussian kernels, the coefficients are roughly $1/n!$, this means to approximate it using polynomial kernel, we only need a poly-logarithmic number of terms.
  - For a broader class of kernels with coefficients being $1/n^c$ where $c > 1$, more terms are needed.
  - We propose a novel sampling scheme, where we exactly compute the first s terms in Taylor expansion, then sample s from the remaining terms. By carefully balancing the number of terms in exact computation and sampling, we achieve a faster runtime.
  - This leads to fast algorithm for approximating arc-cosine kernels and neural tangent kernels.

Applications

- Solve kernel linear systems using our algorithm+preconditioning.

Applications

- Solve kernel linear systems using our algorithm+preconditioning.
- Solve kernel ridge regression by composing with another sketch with even smaller size.

Thanks for attending the talk!

- If you have any questions regarding our paper, feel free to contact the authors via email:

- Zhao Song: magiclinuxkde@gmail.com
- David P. Woodruff: dwoodruf@andrew.cmu.edu
- Zheng Yu: zhengy@princeton.edu
- Lichen Zhang: lichenz@andrew.cmu.edu