Random Fourier Features for Scaling Markov Random Fields

November 25, 2020

The exponential kernel, $\exp(\mathbf{x}^T\mathbf{y})$ with $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, is widely used to parameterize discrete distributions. It can be well-approximated by Random Fourier Features (RFF):

$$\exp(\mathbf{x}^T\mathbf{y}) \approx \phi(\mathbf{x})^T\phi(\mathbf{y}), \tag{1}$$

with random projections $\phi : \mathbb{R}^n \to \mathbb{R}^d$ [3]. Importantly, this approximation is linear, and therefore admits reordering tricks based on the distributive and associative property to improve the efficiency of certain operations by polynomial factors.

1 Attention

One such operation is attention, a popular operation used in neural networks. Naively, attention requires quadratic time complexity. However, the sampled softmax with RFF [3] can be applied to compute attention with linear time complexity [1] (and many others, such as Random Feature Attention, LinFormers, etc.).

1.1 Linear attention with RFF

Given queries $\mathbf{q}_j \in \mathbb{R}^n$, keys $\mathbf{k}_i \in \mathbb{R}^n$, and values $\mathbf{v}_i \in \mathbb{R}^n$ with $t \in [T], i \in [I]$, attention computes outputs

$$o_t = \sum_i \frac{\exp(\mathbf{q}_t^T\mathbf{k}_i)\mathbf{v}_i^T}{\sum_j \exp(\mathbf{q}_t^T\mathbf{k}_j)}. \tag{2}$$

This requires $O(TIn)$ time to compute for all o_t. Applying the RFF approximation, we have

$$o_t \approx \sum_i \frac{(\phi(\mathbf{q}_t)^T\phi(\mathbf{k}_i))\mathbf{v}_i^T}{\sum_j \phi(\mathbf{q}_t)^T\phi(\mathbf{k}_j)} = \frac{\phi(\mathbf{q}_t)^T\sum_i \phi(\mathbf{k}_i)\mathbf{v}_i^T}{\phi(\mathbf{q}_t)^T\sum_j \phi(\mathbf{k}_j)}. \tag{3}$$

The terms $\sum_i \phi(\mathbf{k}_i)\mathbf{v}_i^T$ and $\sum_j \phi(\mathbf{k}_j)$ can be computed once for all queries, reducing the time complexity of computing all o_t to $O(Td + Id^2)$.

Matrix version Matrix form [1] is more informative, write up later. Just breaks up A matrix into linear decomposition, then applies associative property of matmul.

1.2 Approximation error

todo

2 Linear Chain MRFs

The RFF approximations work well in unstructured distributions. Can we get even tighter approximations when distributions have structure? Does the approximation even work?

2.1 Drop-in substitution of kernel approximation

We start with a linear-chain MRF:

$$p(x) \propto \prod_t \psi(x_{t-1}, x_t) = \prod_t \exp(\mathbf{x}_{t-1}^T \mathbf{x}_t),$$

with the variables $x_t \in \mathcal{X}$ and embeddings $\mathbf{x}_t \in \mathbb{R}^n$. As before, we approximate $\psi_t(x_{t-1}, x_t) = \exp(\mathbf{x}_{t-1}^T \mathbf{x}_t) \approx \phi(\mathbf{x}_{t-1})^T \phi(\mathbf{x}_t)$, with the random projection $\phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^d$ chosen appropriately. To start, consider computing the partition function of a simple example with $T = 3$:

$$\begin{aligned} Z &= \sum_{x_1} \sum_{x_2} \psi_1(x_1, x_2) \sum_{x_3} \psi_2(x_2, x_3) \\ &\approx \sum_{x_1} \sum_{x_2} \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2) \sum_{x_3} \phi(\mathbf{x}_2)^T \phi(\mathbf{x}_3) \\ &= \left(\sum_{x_1} \phi(\mathbf{x}_1)^T \right) \left(\sum_{\mathbf{x}_2} \phi(\mathbf{x}_2) \phi(\mathbf{x}_2)^T \right) \left(\sum_{x_3} \phi(\mathbf{x}_3) \right). \end{aligned} \tag{4}$$

We can precompute the sum of outer products $\sum_{\mathbf{x}_2} \phi(\mathbf{x}_2)\phi(\mathbf{x}^T)$ independently, resulting in time complexity $O(Td^2 + T|\mathcal{X}|d^2)$ in serial, and $O(Td^2 + |\mathcal{X}|d^2)$ if the sums of outer products can be computed in parallel. We can also apply a divide-and-conquer approach to further reduce time to $O(c_{\text{mm}}(\log T + \log |\mathcal{X}|))$ on a parallel machine, where c_{mm} is the cost of a matrix multiplication. Compared to the original time complexity of $O(T|\mathcal{X}|^2)$, this is particularly beneficial if $d \ll |\mathcal{X}|$.

Matrix version Probably much clearer here too. todo

2.2 Approximation error

todo

3 Tree MRFs

We proceed from first order linear-chain MRFs, where nodes only have 2 neighbours, to the next simplest model: trees. With tree MRFs, nodes may have more than 2 neighbours depending on the arity of the tree, but the dependencies are still simple and exact inference is tractable.

In linear-chain MRFs all elimination orders are equivalent, as the elimination of a variable cannot result in the addition of any fill-in edges. Indeed, this is the reason why divide-and-conquer strategies are possible in the first place. As this is not the case for tree MRFs (consider removing a node in the middle of a tree), it may not be possible to apply the distributive property in order to improve the time complexity of variable elimination. However, it may be still be possible to obtain speedups through parallelization.

3.1 RFF allows reording of tensor contraction

Consider a star MRF with the following joint distribution:

$$p(x) \propto \psi(x_1, x_2)\psi(x_2, x_3)\psi(x_2, x_4), \tag{5}$$

shown in Figure 1. Elimination takes time $O(V|\mathcal{X}|^2)$, where V is the number of nodes. The time complexity can be lowered on a parallel device to $O(|\mathcal{X}|^2)$ using tensor variable elimination [2].

Applying the RFF approximation, elimination is then given by

$$\begin{aligned} Z &= \sum_{\mathbf{z}} \psi(x_1, x_2)\psi(x_2, x_3)\psi(x_2, x_4) \\ &\approx \sum_{x_1} \sum_{x_2} \phi(x_1)^T \phi(x_2) \sum_{x_3} \phi(x_2)^T \phi(x_3) \sum_{x_4} \phi(x_2)^T \phi(x_4) \\ &= \sum_{x_1} \phi(x_1)^T \sum_{x_2} \phi(x_2) \left(\phi(x_2)^T \sum_{x_3} \phi(x_3) \right) \left(\phi(x_2)^T \sum_{x_4} \phi(x_4) \right). \end{aligned} \tag{6}$$

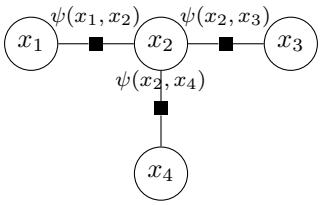

Figure 1

We can un-distribute $\phi(x_2)^T$ from the last two terms with a tensor product:

$$Z \approx \left(\sum_{x_1} \phi(x_1)^T\right) \left(\sum_{x_2} \phi(x_2) \otimes \phi(x_2) \otimes \phi(x_2)\right) \left(\sum_{x_3} \phi(x_3)\right) \left(\sum_{x_4} \phi(x_4)\right), \tag{7}$$

where this approximation can be computed in time $O(|\mathcal{X}|d^3)$.

Generalizing to n-arity trees, we can go from the original runtime of $O(V|\mathcal{X}|^2)$ for variable elimination to $O(Vd^2 + |\mathcal{X}|d^n)$ by applying the RFF approximation and reordering contraction.

3.2 Approximation error

Tensor version, einsum version asdf

3.3 Comparison to Bethe Free Energy?

4 Application to PCFGs

todo

4.1 Approximation error

todo

References

[1] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020.

[2] F. Obermeyer, Eli Bingham, M. Jankowiak, Justin T Chiu, Neeraj Pradhan, Alexander M. Rush, and Noah D. Goodman. Tensor variable elimination for plated factor graphs. *ArXiv*, abs/1902.03210, 2019.

[3] Ankit Singh Rawat, Jiecao Chen, Felix X. Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random fourier features. *CoRR*, abs/1907.10747, 2019. URL `http://arxiv.org/abs/1907.10747`.