Random Fourier Features for Scaling Markov Random Fields

January 25, 2021

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 $\phi : \mathbb{R}^n \to \mathbb{R}^d$ [5]. 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 [5] 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?

change all this

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 $c_{\text{mm}}(d)$ is the cost of a matrix multiplication which is polynomial in d. 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 Hidden Markov Models

An HMM consists of transitions between latent states $p(z_t \mid z_{t-1})$ and emissions from latent states to observed words $p(x_t \mid z_t)$, where these conditional distributions are locally normalized by definition. The joint distribution is given by $p(x, z) = \prod_t p(z_t \mid z_{t-1}) p(x_t \mid z_t)$ with a dummy state $z_0 = \epsilon$. In order to compute the evidence of the observed words, the latent states must be marginalized over: $p(x) = \sum_z p(x, z)$. Applying the RFF approximation to sequence x with $|x| = 2$, we have a linear-chain MRF plus some scaling terms:

$$\begin{aligned} p(x) &= \sum_{z_1} p(z_1 \mid z_0) p(x_1 \mid z_1) \sum_{z_2} p(z_2 \mid z_1) p(x_2 \mid z_2) \\ &= \sum_{z_1} \frac{\phi(\mathbf{z}_0)^T \phi(\mathbf{z}_1)}{\sum_{z_1'} \phi(\mathbf{z}_0)^T \phi(\mathbf{z}_1')} \frac{\phi(\mathbf{z}_1)^T \phi(\mathbf{x}_1)}{\sum_{x_1} \phi(\mathbf{z}_1)^T \phi(\mathbf{x}_1)} \sum_{z_2} \frac{\phi(\mathbf{z}_1)^T \phi(\mathbf{z}_2)}{\sum_{z_2'} \phi(\mathbf{z}_1)^T \phi(\mathbf{z}_2')} \frac{\phi(\mathbf{z}_2)^T \phi(\mathbf{x}_2)}{\sum_{x_2} \phi(\mathbf{z}_2)^T \phi(\mathbf{x}_2)} \\ &= \left(\frac{\phi(\mathbf{z}_0)^T}{\phi(\mathbf{z}_0)^T \sum_{z_1'} \phi(\mathbf{z}_1')} \right) \left(\sum_{z_1} \frac{\left(\phi(\mathbf{z}_1)^T \phi(\mathbf{x}_1)\right) \phi(\mathbf{z}_1) \phi(\mathbf{z}_1)^T}{\left(\phi(\mathbf{z}_1)^T \sum_{x_1} \phi(\mathbf{x}_1)\right) \left(\phi(\mathbf{z}_1)^T \sum_{z_2'} \phi(\mathbf{z}_2')\right)} \right) \left(\sum_{z_2} \frac{\left(\phi(\mathbf{z}_2)^T \phi(\mathbf{x}_2)\right) \phi(\mathbf{z}_2)}{\phi(\mathbf{z}_2)^T \sum_{x_2} \phi(\mathbf{x}_2)} \right). \end{aligned} \tag{5}$$

In the above equation, we applied the RFF trick to both the transitions and emissions. The trick also results in a speedup of the same order of magnitude when applied only to the transition.

$$\begin{aligned} p(x) &= \sum_{z_1} p(z_1 \mid z_0) p(x_1 \mid z_1) \sum_{z_2} p(z_2 \mid z_1) p(x_2 \mid z_2) \\ &= \sum_{z_1} \frac{\phi(\mathbf{w}_{z_0})^T \phi(\mathbf{u}_{z_1})}{\sum_{z_1'} \phi(\mathbf{w}_{z_0})^T \phi(\mathbf{u}_{z_1'})} p(x_1 \mid z_1) \sum_{z_2} \frac{\phi(\mathbf{w}_{z_1})^T \phi(\mathbf{u}_{z_2})}{\sum_{z_2'} \phi(\mathbf{w}_{z_1})^T \phi(\mathbf{u}_{z_2'})} p(x_2 \mid z_2) \\ &= \left(\frac{\phi(\mathbf{w}_{z_0})^T}{\phi(\mathbf{w}_{z_0})^T \sum_{z_1'} \phi(\mathbf{u}_{z_1'})} \right) \left(\sum_{z_1} \frac{p(x_1 \mid z_1) \phi(\mathbf{u}_{z_1}) \phi(\mathbf{w}_{z_1})^T}{\phi(\mathbf{w}_{z_1})^T \sum_{z_2'} \phi(\mathbf{u}_{z_2'})} \right) \left(\sum_{z_2} p(x_2 \mid z_2) \phi(\mathbf{u}_{z_2}) \right), \end{aligned} \tag{6}$$

where $\mathbf{w}_z, \mathbf{u}_z \in \mathbb{R}^n$ are embeddings for state z.

4 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), we cannot get the same speedups as in the linear-chain MRF. However, speedups due to parallelization and projection are still be possible.

4.1 RFF allows reording of tensor contraction

Consider a star MRF with the following joint distribution:

$$p(x) \propto \psi(x_1, x_2)\psi(x_2, x_3)\psi(x_2, x_4), \tag{7}$$

shown in Figure 1. Elimination takes time $O(V|\mathcal{X}|^2)$, where V is the number of nodes. The time

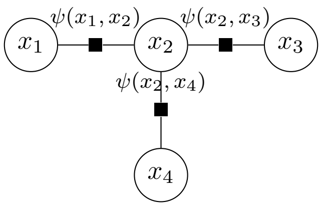

Figure 1

complexity can be lowered on a parallel device to $O(|\mathcal{X}|^2)$ using tensor variable elimination [4].

Applying the RFF approximation, elimination is then given by

$$\begin{aligned} Z &= \sum_{\mathbf{z}} \psi(x_1, x_2)\psi(x_2, x_3)\psi(x_2, x_4) \\ &\approx \sum_{x_1}\sum_{x_2} \phi(x_1)^T\phi(x_2) \sum_{x_3} \phi(x_2)^T\phi(x_3) \sum_{x_4} \phi(x_2)^T\phi(x_4) \\ &= \sum_{x_1} \phi(x_1)^T \sum_{x_2} \phi(x_2) \left(\phi(x_2)^T \sum_{x_3} \phi(x_3)\right) \left(\phi(x_2)^T \sum_{x_4} \phi(x_4)\right). \end{aligned} \tag{8}$$

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

$$Z \approx \left(\sum_{x_1} \phi(x_1)^T\right) \left(\sum_{x_2} \phi(x_2) \otimes \phi(x_2) \otimes \phi(x_2)\right) \left(\sum_{x_3} \phi(x_3)\right) \left(\sum_{x_4} \phi(x_4)\right), \tag{9}$$

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 $O(E|\mathcal{X}|^2)$ for variable elimination to $O(Ed^2 + |\mathcal{X}|d^n)$ by applying the RFF approximation and reordering contraction, where E is the number of edges.

definitely something slightly weaker than the commutative property is being used here, which is clear in the einsum version. not sure what to call this? is this a generalized cyclical trace property or just the tensor analogue of polynomials

4.2 Approximation error

Tensor version, einsum version

asdf

5 Application to globally normalized HMMs

The tree example of RFF-MRFs above extends directly to globally normalized HMMs. Consider an HMM with discrete observations $x_t \in \mathcal{X}$ and discrete latent states $z_t \in \mathcal{Z}$. The joint distribution is given by

$$p(x, z) \propto \prod_t \psi(x_{t-1}, x_t)\psi(z_t, x_t) = \prod_t \exp(\mathbf{x}_{t-1}, \mathbf{x}_t)\exp(\mathbf{z}_t, \mathbf{x}_t), \tag{10}$$

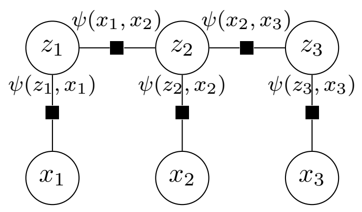

Figure 2

with $\mathbf{x}_t, \mathbf{z}_t \in \mathbb{R}^d$. The graphical model is given in Figure 2.

To compute the partition function we perform the same computation as in Equation 8:

$$\begin{aligned} Z &= \sum_x \sum_z \prod_t \psi(x_{t-1}, x_t)\psi(z_t, x_t) \\ &\approx \left(\left(\sum_{x_1} \phi(\mathbf{x}_1)^T \right) \left(\sum_{z_1} \phi(\mathbf{z}_1) \otimes \phi(\mathbf{z}_1) \right) \right) \\ &\quad \cdot \left(\left(\sum_{x_2} \phi(\mathbf{x}_2) \right) \left(\sum_{z_2} \phi(\mathbf{z}_2) \otimes \phi(\mathbf{z}_2) \otimes \phi(\mathbf{z}_2) \right) \right) \\ &\quad \cdot \left(\left(\sum_{x_3} \phi(\mathbf{x}_3) \right) \left(\sum_{z_3} \phi(\mathbf{z}_3) \otimes \phi(\mathbf{z}_3) \otimes \phi(\mathbf{z}_3) \right) \right) \cdots \\ &\quad \cdot \left(\left(\sum_{z_T} \phi(\mathbf{z}_T) \otimes \phi(\mathbf{z}_T) \right) \left(\sum_{x_T} \phi(\mathbf{x}_T) \right) \right). \end{aligned} \tag{11}$$

Again, each of the summations and summands can be computed in parallel.

probably take out x_3

expand / finish

6 HSMM

7 Application to WCFGs

Algorithm 1 The inside algorithm from [2]

```
1:  function INSIDE($\pi$, $w$)
2:      initialize all $\beta[\cdot]$ to 0
3:      for $k := 1 \to n$ do                                    ▷ width-1 constituents
4:          for $A \in \mathcal{N}$ do
5:              $\beta[A^k_{k-1}] += \pi(A \to w_k)$                      ▷ $O(K^2)$
6:      for width $:= 2 \to n$ do                                ▷ wider constituents
7:          for $i := 0 \to n -$ width do                        ▷ start point
8:              $k := i +$ width                                  ▷ end point
9:              for $j := i + 1 \to k - 1$ do                     ▷ midpoint
10:                 for $A, B, C \in \mathcal{N}$ do
11:                     $\beta[A^k_i] += \pi(A \to BC)\beta[B^j_i]\beta[C^k_j]$
12:     return $Z := \beta[\text{ROOT}^n_0]$
```

The inside algorithm is outlined in Algorithm 1. With n words and $|\mathcal{N}|$ non-terminals, the complexity is

$$O(n^3|\mathcal{N}|^3), \tag{12}$$

where the major complexity comes from the updates at line 11: $\beta[A^k_i] += \pi(A \to BC)\beta[B^j_i]\beta[C^k_j]$.

If we make the assumption that

$$\pi(A \to BC) = \frac{\phi(A, B) \cdot \psi(A, C)}{\sum_{B'C'} \phi(A, B') \cdot \psi(A, C')}, \tag{13}$$

where ϕ and ψ are feature functions that maps A, B and A, C to vectors of length r, then we can calculate line 11 as

$$\beta[A_i^k] = \sum_{BC} \frac{\phi(A,B) \cdot \psi(A,C) \beta[B_i^j] \beta[C_j^k]}{\sum_{B'C'} \phi(A,B') \cdot \psi(A,C')} \tag{14}$$

$$= \frac{(\sum_B \beta[B_i^j]\phi(A,B)) \cdot (\sum_C \psi(A,C)\beta[C_j^k])}{(\sum_{B'} \phi(A,B')) \cdot (\sum_{C'} \psi(A,C'))}, \tag{15}$$

and reduce the total complexity to

$$O(n^3|\mathcal{N}|^2 r). \tag{16}$$

Note that the parameterization here is different from [3], where $\pi(A \to BC)$ is parameterized as

$$\pi(A \to BC) = \frac{\exp(\mathbf{u}_{BC}\mathbf{w}_A + \mathbf{b}_{BC})}{\sum_{B'C'} \exp(\mathbf{u}_{B'C'}\mathbf{w}_A + \mathbf{b}_{B'C'})}, \tag{17}$$

where $\mathbf{u}_{BC}$ is the embedding vector of (B, C), and $\mathbf{w}_A$ is the embedding vector of A.

8 Application to MRF-LM

n-gram potentials (yacine's paper?) loop over different lengths to compute partition fn?

9 Numerical Stability of Linear Kernel

Let $\phi(x) = \frac{1}{\sqrt{d}} \exp\left(W^T x - \frac{\|x\|_2^2}{2}\right)$, with $x \in \mathbb{R}^n$ and $W \in \mathbb{R}^{n \times d}$. We would like to compute:

$$f(x, y) = \langle \phi(x), \phi(y) \rangle, \tag{18}$$

with $x, y \in \mathbb{R}^n$.

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] Jason Eisner. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, 2016.

[3] Yoon Kim, Chris Dyer, and Alexander M Rush. Compound probabilistic context-free grammars for grammar induction. *arXiv preprint arXiv:1906.10225*, 2019.

[4] 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.

[5] 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`.