Sampled Softmax with Random Fourier Features

Ankit Singh Rawat, Jiecao Chen, Felix Xinnan Yu, Ananda Theertha Suresh, Sanjiv Kumar

Google Research, New York

Problem

- Computational cost of training with softmax based cross-entropy loss scales linearly with number of classes.
- Infeasible for many real-life applications.
- **Sampled softmax** computes loss based on a small subset of sampling negative classes.
- Sampling distribution plays a crucial role in the training speed and the quality of the final model.

Goal: Design provably accurate sampling methods with low computational cost.

Background

- Let $\boldsymbol{\theta}$ denote the model parameters and $\mathbf{h}$ be the embedding generated by the model for input $\mathbf{x}$.
- Let $\mathbf{c}_1, \ldots, \mathbf{c}_n \in \mathbb{R}^d$ be the class embeddings.

Softmax cross-entropy loss

- The model assigns probability to the i-th class according to the softmax distribution:

$$p_i = e^{o_i}/Z$$

$o_i = \tau \mathbf{h}^T \mathbf{c}_i$: logit for class i and $Z = \sum_{i \in [n]} e^{o_i}$.

- The full softmax cross-entropy loss is defined as

$$\mathcal{L}(\mathbf{x}, t) := -\log p_t = -o_t + \log Z$$

$t \in [n]$: true class for the input $\mathbf{x}$.

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{x}, t) = -\nabla_{\boldsymbol{\theta}} o_t + \sum_{i=1}^{n} \left(e^{o_i}/Z\right) \cdot \nabla_{\boldsymbol{\theta}} o_i$$

Sampled Softmax [Bengio and Senécal'08]

- Sample m negative classes $s_1, \ldots, s_m \overset{\text{i.i.d.}}{\sim} q$.
- Define the sampled softmax distribution

$$p'_i = e^{o'_i}/Z'$$

$o'_{i+1} = o_{s_i} - \log(m q_{s_i})$: adjusted logit for the s_i-th class.

- Sampled softmax loss:

$$\mathcal{L}'(\mathbf{x}, t) = -\log p'_t = -o_t + \log Z'.$$

$O(nd)$ vs. $O(md)$ cost for gradient computation.

Bengio and Senécal'08. Adaptive importance sampling to accelerate training of a neural probabilistic language model, IEEE Transactions on Neural Networks, 2008.

Our contributions: an overview

Only the softmax distribution provides an *unbiased* gradient estimate, which again has $O(nd)$ cost.

- We characterize the bias of the gradient estimate for a generic sampling distribution.
- We propose RF-softmax, a kernel-based sampling method via D Random Fourier features (RFF).
  - Provably small bias with large enough D.
  - $O(D \log n)$ sampling cost.

Gradient bias of sampled softmax

For a sampling distribution q over $[n] \backslash \{t\}$,

$$\text{LB} \le \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \mathcal{L}'\right] - \nabla_{\boldsymbol{\theta}} \mathcal{L} \le \text{UB}$$

where

$$\text{LB} \triangleq -\frac{M \sum_{k \in \mathcal{N}_t} e^{o_k} \left| Z_t - \frac{e^{o_k}}{q_k} \right|}{m Z^2} \left(1 - o\left(\frac{1}{m}\right)\right) \cdot \mathbf{1}$$

$$\text{UB} \triangleq \left(\frac{2M}{m} \frac{\max_{i,i' \in \mathcal{N}_t} \left| \frac{e^{o_i}}{q_i} - \frac{e^{o_{i'}}}{q_{i'}} \right| Z_t}{Z^2 + \sum_{j \in \mathcal{N}_t} \frac{e^{2 o_j}}{q_j}} + o\left(\frac{1}{m}\right) \right) \cdot \mathbf{1} + \left(\frac{\sum_{j \in \mathcal{N}_t} \frac{e^{2 o_j}}{q_j} - Z_t^2}{m Z^3} + o\left(\frac{1}{m}\right) \right) \cdot \mathbf{g}$$

$Z_t \triangleq \sum_{j \in \mathcal{N}_t} e^{o_j}$, $\mathbf{g} \triangleq \sum_{j \in \mathcal{N}_t} e^{o_j} \nabla_{\boldsymbol{\theta}} o_j$ and $\mathbf{1}$: all one vector.

A good sampling distribution should ensure a tight multiplicative approximation of the softmax distribution in a computational efficient manner.

Kernel-based sampling (I)

Given a kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that can be linearized by a mapping $\phi : \mathbb{R}^d \to \mathbb{R}^d$, define

$$q_i = \frac{K(\mathbf{h}, \mathbf{c}_i)}{\sum_{j=1}^{n} \phi(\mathbf{h})^T \phi(\mathbf{c}_j)} = \frac{\phi(\mathbf{h})^T \phi(\mathbf{c}_i)}{\phi(\mathbf{h})^T \sum_{j=1}^{n} \phi(\mathbf{c}_j)}.$$

- The distribution q enables sampling one class with cost $O(D \log n)$ [Blanc and Steffen'18]

Blanc and Steffen'18. Adaptive sampled softmax with kernel based sampling, ICML 2018.

Kernel-based sampling (II)

- [Blanc and Steffen'18] proposed kernel-based sampling with the quadratic kernel

$$K_{\text{quad}} = \alpha \cdot (\mathbf{h}^T \mathbf{c}_i)^2 + 1$$

with $\phi(\mathbf{z}) = [\sqrt{\alpha}(\mathbf{z} \otimes \mathbf{z}), 1] \in \mathbb{R}^{d^2}$.

- Poor approximation of the exponential kernel and prohibitively large $O(d^2 n)$ computational cost.

Random Fourier softmax (RF-softmax)

Method	Quadratic	Random Fourier features			Random Maclaurin features
D	256^2	100	1000	256^2	256^2
MSE	2.8e-3	2.6e-3	2.7e-4	5.5e-6	8.8e-2

Table 1:MSE for approximating e^o using different methods.

- For **normalized** input and class embeddings,

$$e^o = e^{\tau \mathbf{h}^T \mathbf{c}} = e^{\tau} e^{-\frac{\tau \|\mathbf{h} - \mathbf{c}\|_2^2}{2}}$$

- Random Fourier features

$$\phi(\mathbf{u}) = \frac{1}{\sqrt{D}} \left[\cos(\mathbf{w}_1^T \mathbf{u}), \ldots, \cos(\mathbf{w}_D^T \mathbf{u}), \sin(\mathbf{w}_1^T \mathbf{u}), \ldots, \sin(\mathbf{w}_D^T \mathbf{u}) \right],$$

with $\mathbf{w}_i \sim N(0, \mathbf{I}/\nu)$, give an unbiased estimator of the shift-invariant Gaussian kernel

$$K(\mathbf{x} - \mathbf{y}) = e^{-\frac{\nu \|\mathbf{x} - \mathbf{y}\|_2^2}{2}}$$

- Given an input embedding $\mathbf{h}$, RF-softmax picks the i-th class with probability

$$q_i \propto \phi(\mathbf{c}_i)^T \phi(\mathbf{h})$$

Quality of approximation: For normalized embeddings, as long as, $e^{2\nu} \le \frac{\gamma}{\rho \sqrt{d}} \cdot \frac{\sqrt{D}}{\log D}$, the following holds with probability at least $1 - \left(\frac{1}{D^2}\right)$.

$$e^{(\tau - \nu) \mathbf{h}^T \mathbf{c}_i} \cdot (1 - 2\gamma) \le \frac{1}{\sum_{i \in \mathcal{N}_t} e^{o_i}} \cdot \left| \frac{e^{o_i}}{q_i} \right| \le e^{(\tau - \nu) \mathbf{h}^T \mathbf{c}_i} \cdot (1 + a\gamma^3),$$

where γ, ρ, and a are positive constants.

- With large enough D,

$$q_i \propto (1 \pm o_D(1)) \cdot p_i.$$

In particular, at $D = \infty$, we have $q_i \propto p_i$.

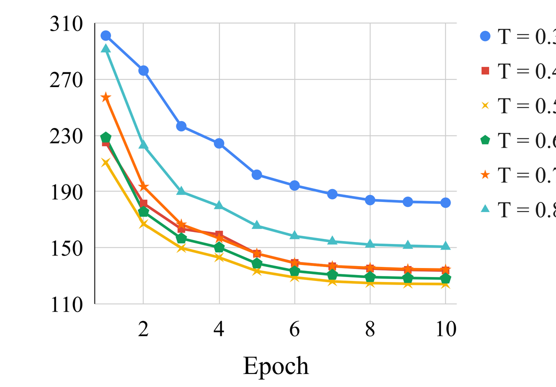
Bias vs. variance trade-off dictates the choice of ν for RF-softmax.

Experiments

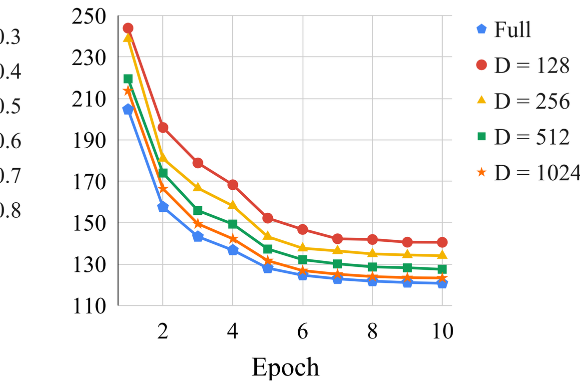
Wall time: Batch size $= 10, m = 10, d = 64$

# classes (n)	Method	Wall time
10,000	Exp	1.4 ms
	Quadratic	6.5 ms
	RF-softmax ($D = 50$)	0.5 ms
	RF-softmax ($D = 200$)	0.6 ms
	RF-softmax ($D = 500$)	1.2 ms
	RF-softmax ($D = 1,000$)	1.4 ms
500,000	Exp	32.3 ms
	Quadratic	8.2 ms
	RF-softmax ($D = 50$)	1.6 ms
	RF-softmax ($D = 200$)	1.7 ms
	RF-softmax ($D = 500$)	2.0 ms
	RF-softmax ($D = 1,000$)	2.4 ms

Design choices: Penn tree bank (PTB), $m = 100$

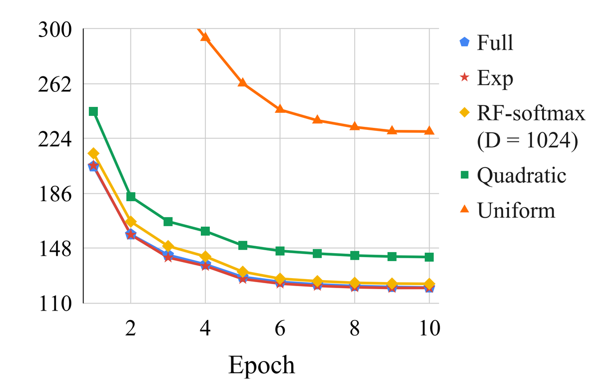


(a) Fixed D, varying $T = 1/\sqrt{\nu}$

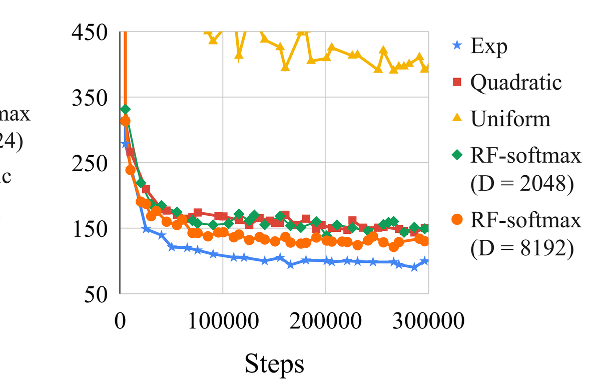


(b) Fixed T, varying D

Performance: NLP datasets.



(c) Validation perplexity vs. training steps (Penn tree bank, $m = 100$).



(d) Validation perplexity vs. training steps (Bnews, $m = 100$).

Performance: Extreme classification datasets

Dataset	Method	Prec@1	Prec@3	Prec@5
AmazonCat-13k $n = 13,330$ $v = 203,882$	Exp	0.87	0.76	0.62
	Uniform	0.83	0.69	0.55
	Quadratic	0.84	0.74	0.60
	RF-softmax	0.87	0.75	0.61
Delicious-200k $n = 205,443$ $v = 782,585$	Exp	0.42	0.38	0.37
	Uniform	0.36	0.34	0.32
	Quadratic	0.40	0.36	0.34
	RF-softmax	0.41	0.37	0.36
WikiLSHTC $n = 325,056$ $v = 1,617,899$	Exp	0.58	0.37	0.29
	Uniform	0.47	0.29	0.22
	Quadratic	0.57	0.37	0.28
	RF-softmax	0.56	0.35	0.26