Supplementary Material

Algorithm 1 Streaming Probabilistic Deep Tensor Factorization (SPIDER) (the equations are referenced from the main paper)

```
Initialize the spike-and-slab prior approximation and multiply it with all the other priors to initialize $q_{\text{cur}}(\cdot)$
while a new batch of observed tensor entries $\mathcal{B}_t$ arrives do
    for each entry $\mathbf{i}_n$ in $\mathcal{B}_t$ do
        Approximate the running model evidence with (10) (binary data) or (11) (continuous data).
        Update the posterior for $\mathcal{W}$ and the associated embeddings $\{u^k_{i_{nj}}\}_{k,j}$ with (7).
        if continuous data then
            Update the posterior for inverse noise variance $\tau$ via conditional moment matching.
        end if
        Update the spike-and-slab prior approximation with standard EP.
    end for
end while
return the current posterior $q_{\text{cur}}(\cdot)$.
```

1 Online Posterior Update for the Inverse Noise Variance

To update $q_{\text{cur}}(\tau)$, we consider the blending distribution only in terms of the NN output f_o and τ,

$$\tilde{p}(f_o, \tau) \propto q_{\text{cur}}(f_o) q_{\text{cur}}(\tau) \mathcal{N}(y_{\mathbf{i}_n}|f_o, \tau^{-1}) = \mathcal{N}(f_o|\alpha_n, \beta_n)\text{Gamma}(\tau|a, b)\mathcal{N}(y_{\mathbf{i}_n}|f_o, \tau^{-1}). \quad (1)$$

Following the conditional expectation propagation (CEP) framework proposed by Wang and Zhe (2019), we first derive the conditional moments of τ given f_o and then approximate the expectation of the conditional moments to obtain the moments and update the posterior of τ. Specifically, from (1), we can easily derive the conditional blending distribution,

$$\tilde{p}(\tau|f_o) = \text{Gamma}(\tau|\hat{a}, \hat{b}) \quad (2)$$

where $\hat{a} = a + \frac{1}{2}$ and $\hat{b} = b + \frac{1}{2}(y^2_{\mathbf{i}_n} - 2f_o + f_o^2)$. We can obtain the conditional moments of τ,

$$\mathbb{E}_{\tilde{p}(\tau|f_o)}[\tau] = \frac{\hat{a}}{\hat{b}}, \qquad \mathbb{E}_{\tilde{p}(\tau|f_o)}[\log(\tau)] = \Psi(\hat{a}) - \log(\hat{b}).$$

where $\Psi(\cdot)$ is the digamma function. Note that these moments are based on the sufficient statistics of Gamma distribution, which are standard for moment matching in ADF and EP framework. The true moments can therefore be calculated by taking the expectation of the conditional moments,

$$\mathbb{E}_{\tilde{p}}[\tau] = \mathbb{E}_{\tilde{p}(f_o)}\mathbb{E}_{\tilde{p}(\tau|f_o)}[\tau] = \mathbb{E}_{\tilde{p}(f_o)}[\frac{\hat{a}}{\hat{b}}],$$

$$\mathbb{E}_{\tilde{p}}[\log(\tau)] = \mathbb{E}_{\tilde{p}(f_o)}\mathbb{E}_{\tilde{p}(\tau|f_o)}[\log(\tau)] = \mathbb{E}_{\tilde{p}(f_o)}[\Psi(\hat{a}) - \log(\hat{b})].$$

However, the normalization constant for (1) is intractable and it is difficult to compute the marginal blending distribution $\tilde{p}(f_o)$. To overcome this problem, we approximate $\tilde{p}(f_o)$ with the current posterior of f_o, namely $q_{\text{cur}}(f_o)$. This is reasonable, because $\tilde{p}(f_o)$ is an integration of $q(f_o)$ and one new data point; when we have processed many data points, adding one more data point is unlikely to significantly change the posterior. In other words, we can assume $q(f_o)$ and $\tilde{p}(f_o)$ are close in high density regions. Hence, we can approximate

$$\mathbb{E}_{\tilde{p}}[\tau] \approx \mathbb{E}_{q_{\text{cur}}(f_o)}[\frac{\hat{a}}{\hat{b}}],$$

$$\mathbb{E}_{\tilde{p}}[\log(\tau)] \approx \mathbb{E}_{q_{\text{cur}}(f_o)}[\Psi(\hat{a}) - \log(\hat{b})].$$

A second problem is that due to the nonlinearity of the conditional moments, even with $q_{\text{cur}}(f_o)$ (which has a nice Gaussian form), we still cannot analytically compute the expectation. To address this issue, we further observe that the conditional moments are functions of f_o and f_o^2,

$$h_1(f_o, f_o^2) = \frac{a + \frac{1}{2}}{b + \frac{1}{2}(y^2_{\mathbf{i}_n} - 2f_o + f_o^2)},$$

$$h_2(f_o, f_o^2) = \Psi(a + \frac{1}{2}) - \log\big(b + \frac{1}{2}(y^2_{\mathbf{i}_n} - 2f_o + f_o^2)\big).$$

Define $\mathbf{f} = [f_o, f_o^2]^\top$. We use a Taylor expansion at the mean of f_o and f_o^2 to approximate the conditional moments,

$$\begin{aligned}
h_1(f_o, f_o^2) &\approx h_1(\mathbb{E}_{q_{\text{cur}}}[f_o], \mathbb{E}_{q_{\text{cur}}}[f_o^2]) + (\mathbf{f} - \mathbb{E}_{q_{\text{cur}}}[\mathbf{f}])^\top \nabla h_1|_{\mathbf{f}=\mathbb{E}_{q_{\text{cur}}}[\mathbf{f}]}, \\
h_2(f_o, f_o^2) &\approx h_2(\mathbb{E}_{q_{\text{cur}}}[f_o], \mathbb{E}_{q_{\text{cur}}}[f_o^2]) + (\mathbf{f} - \mathbb{E}_{q_{\text{cur}}}[\mathbf{f}])^\top \nabla h_2|_{\mathbf{f}=\mathbb{E}_{q_{\text{cur}}}[\mathbf{f}]}. \qquad (3)
\end{aligned}$$

We take expectation over the Taylor expansion, and obtain a closed-form result,

$$\mathbb{E}_{\tilde{p}}[\tau] \approx \frac{a^*}{b^*}, \quad \mathbb{E}_{\tilde{p}}[\tau] \approx \Psi(a^*) - \log(b^*) \tag{4}$$

where

$$a^* = a + \frac{1}{2}, \quad b^* = b + \frac{1}{2}((y_{\mathbf{i}_n} - \alpha_n)^2 + \beta_n).$$

Finally, from these moments, we can obtain the updated the posterior, $q(\tau) = \mathrm{Gamma}(\tau | a^*, b^*)$.

2 The Updates for Spike-and-Slab Prior Approximation

In our streaming posterior inference, after we execute ADF to process all the entries in the newly received batch, we use standard EP to update the spike-and-slab prior approximation. In this way, we can refine the approximation quality so as to effectively sparsify and condense the neural network to prevent overfitting. Specifically, for each weight w_{mjt}, we first divide the posterior by the prior approximation to obtain the calibrated (or context) distribution,

$$q^{\backslash}(w_{mjt}, s_{mjt}) \propto \frac{q_{\text{cur}}(w_{mjt}, s_{mjt})}{A(w_{mjt}, s_{mjt})} = \mathrm{Bern}(s_{mjt}|\rho_0)\mathcal{N}(w_{mjt}|\mu_{mjt}^{\backslash}, v_{mjt}^{\backslash})$$

where $A(w_{mjt}, s_{mjt}) = \mathrm{Bern}\big(s_{mjt}|c(\rho_{mjt})\big)\mathcal{N}(w_{mjt}|\mu_{mjt}^0, v_{mjt}^0)$ (see (13) in the main paper). Because both q_{cur} and A belong to the exponential family, this can be easily done by subtracting the natural parameters. Note that $\mathrm{Bern}(s_{mjt}|\rho_0)$ is the prior of s_{mjt} (see (4) and (5) in the main paper) — this comes from the fact that the (approximate) posterior of s_{mjt} is proportional to the product of its prior and the approximation term in A.

Next, we combine the calibrated distribution and the exact prior to obtain a tilted distribution (which is similar to the blending distribution in the streaming case),

$$\tilde{p}(w_{mjt}, s_{mjt}) \propto q^{\backslash}(w_{mjt}, s_{mjt})\big(s_{mjt}\mathcal{N}(w_{mjt}|0, \sigma_0^2) + (1 - s_{mjt})\delta(w_{mjt})\big). \tag{5}$$

We then project $\tilde{p}$ to the exponential family to obtain the updated posterior,

$$q^*(w_{mjt}, s_{mjt}) = \mathrm{Bern}(s_{mjt}|c(\rho_{mjt}^*))\mathcal{N}(w_{mjt}|\mu_{mjt}^*, v_{mjt}^*),$$

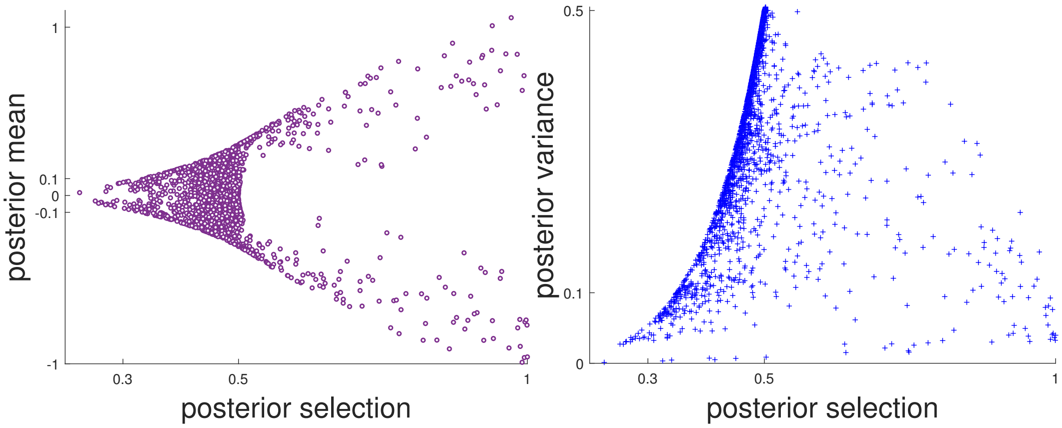
where $c(\cdot)$ is the sigmoid function,

$$\begin{aligned}
\rho_{mjt}^* &= \log\big(\frac{\mathcal{N}(\mu_{mjt}^{\backslash}|0, \sigma_0^2 + v_{mjt}^{\backslash})}{\mathcal{N}(\mu_{mjt}^{\backslash}|0, v_{mjt}^{\backslash})}\big), && (6) \\
\mu_{mjt}^* &= c(\widehat{\rho}_{mjt})\widehat{\mu}_{mjt}, && (7) \\
v_{mjt}^* &= c(\widehat{\rho}_{mjt})\big(\widehat{v}_{mjt} + (1 - c(\widehat{\rho}_{mjt}))\widehat{\mu}_{mjt}^2\big), && (8) \\
& && (9)
\end{aligned}$$

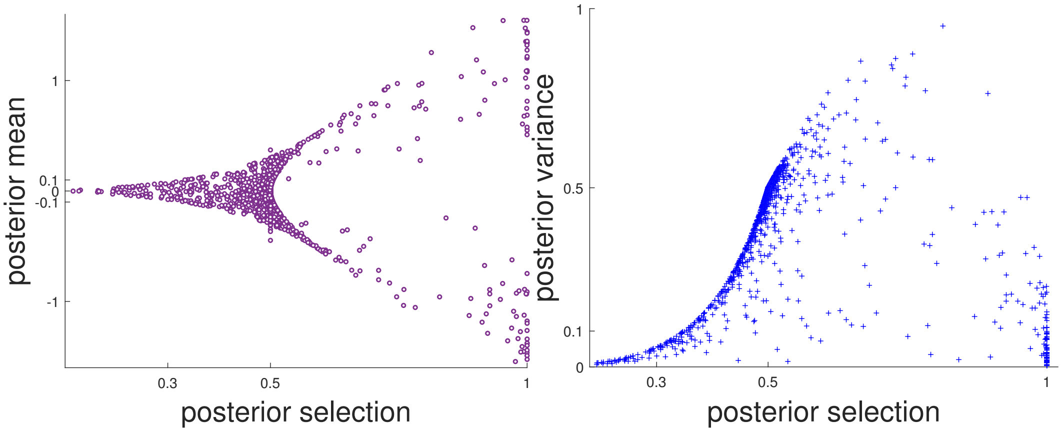
and

$$\begin{aligned}
\widehat{\rho}_{mjt} &= \rho_{mjt}^* + c^{-1}(\rho_0), \\
\widehat{v}_{mjt} &= \big((v_{mjt}^{\backslash})^{-1} + \sigma_0^{-2}\big)^{-1}, \\
\widehat{\mu}_{mjt} &= \widehat{v}_{mjt}\frac{\mu_{mjt}^{\backslash}}{v_{mjt}^{\backslash}}.
\end{aligned}$$

Finally, we can obtain the updated the prior approximation term via dividing the updated posterior by the calibrated distribution, $A^*(w_{mjt}, s_{mjt}) \propto q^*(w_{mjt}, s_{mjt})/q^{\backslash}(w_{mjt}, s_{mjt})$. Now, we replace the current prior approximation by A^* and set $q_{\text{cur}} = q^*$, to prepare for the steaming inference in the next batch.



(a) *Tanh* (b) *Tanh*



(c) *ReLU* (d) *ReLU*

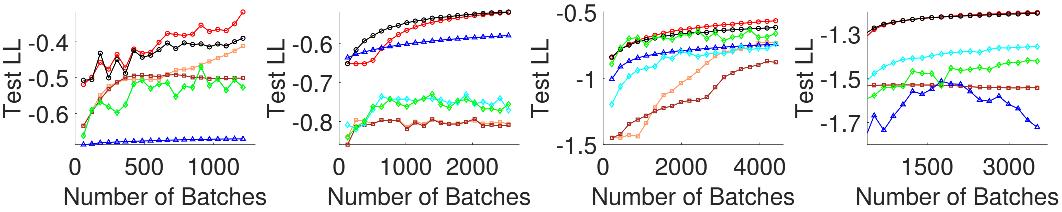
Figure 1: Posterior selection probability *v.s.* posterior mean and variance for each NN weight.

3 Posterior of the NN Weights

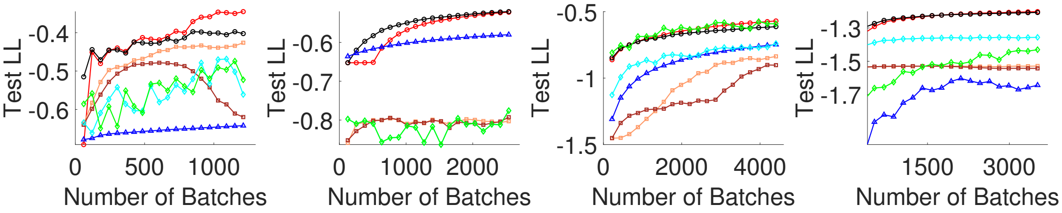
To see if SPIDER can indeed sparsify the network in streaming factorization, we looked into the estimated posterior distribution of the NN weights. We set the rank to 8 and streaming batch-size to 256, and ran SPIDER on *DBLP* dataset. In Fig. 1a and c, we show the pair of the posterior selection probability ρ_{mjt} and the posterior mean μ_{mjt} for each weight w_{mjt}, and in Fig. 1b and d, we show pairs of ρ_{mjt} and the posterior variance v_{mjt} for each weight. As we can see, when the posterior selection probability ρ_{smt} is less than 0.5, *i.e.,* the weight w_{mjt} is likely to be useless/redundant, both the posterior mean μ_{mjt} and variance v_{mjt} are small and close to 0. The more ρ_{mjt} approaches 0, the closer to 0 both μ_{mjt} and v_{mjt} are, exhibiting a shrinkage effect. Thereby the corresponding weight w_{mjt} is inhibited or deactivated. By contrast, when ρ_{smt} is bigger than 0.5, the posterior mean and variance have much larger scales and ranges, implying that the corresponding weight is active and estimated from data freely. Therefore, the learned posterior weights and selection probabilities are consistent, and they effectively deactivate many weights to adjust the complexity of the network.

4 Running Time

We implemented our method SPIDER by Theano and SVB-DTF, SVB-GPTF, SS-GPTF by TensorFlow. POST was implemented with Matlab. We ran all the methods on a desktop machine with Intel i9-9900K CPU and 32GB memory. We did not use GPU acceleration to run SPIDER for a fair comparison. SPIDER is faster than POST on *MovieLen1M* but slower on the other datasets. For example, for rank $r = 8$ and batch-size 128, the running time (in seconds) are $\{$SPIDER-ReLU: 6,928, SPIDER-tanh: 8,566, POST: 40,551$\}$ on *MovieLen1M*, $\{$SPIDER-ReLU: 13,742, SPIDER-tanh: 15,641, POST: 1,442$\}$ on *ACC*, $\{$SPIDER-ReLU: 5,255, SPIDER-tanh: 5,356, POST: 2,459$\}$ on *Anime* and $\{$SPIDER-ReLU: 1,349, SPIDER-tanh: 1,481, POST: 371$\}$ on *DBLP*. Note that on different datasets, POST may need a different number of iterations to converge in the mean-field variational updates. The tolerance level was set to 10^{-5} and the maximum number of iterations 500. The results are reasonable, because SPIDER is based on neural networks, including much more parameters than CP factorization, and the computation is much more complex. The other methods are in general faster than SPIDER. This might be partly due to the difference in efficiency between Theano and TensorFlow libraries. Nonetheless, as we can see from the main paper, the predictive performance of those methods are much worse than SPIDER and even often worse than POST.



(a) *DBLP* (r = 3) (b) *Anime* (r = 3) (c) *ACC* (r = 3) (d) *MovieLen1M* (r = 3)



(e) *DBLP* (r = 8) (f) *Anime* (r = 8) (g) *ACC* (r = 8) (h) *MovieLen1M* (r = 8)

Figure 2: Running prediction accuracy along with the number of processed streaming batches. The batch size was fixed to 256.

References

Wang, Z. and Zhe, S. (2019). Conditional expectation propagation. In *UAI*, page 6.