Scalable Bayesian Learning of Recurrent Neural Networks for Language Modeling: Supplementary Material

Zhe Gan*, **Chunyuan Li***†, **Changyou Chen, Yunchen Pu, Qinliang Su, Lawrence Carin**
Department of Electrical and Computer Engineering, Duke University
{zg27, cl319, cc448, yp42, qs15, lcarin}@duke.edu

A Gated Recurrent Units

Similar to the LSTM unit, the GRU (Cho et al., 2014) has gating units that modulate the flow of information inside the unit, however, without using a separate memory cell. Specifically, the GRU has two gates: the reset gate $\boldsymbol{r}_t$ and the update gate $\boldsymbol{z}_t$. The hidden units $\boldsymbol{h}_t$ are updated as follows:

$$\boldsymbol{r}_t = \sigma(\mathbf{W}_r \boldsymbol{x}_t + \mathbf{U}_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_r)\,, \tag{1}$$

$$\boldsymbol{z}_t = \sigma(\mathbf{W}_z \boldsymbol{x}_t + \mathbf{U}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z)\,, \tag{2}$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\mathbf{W}_h \boldsymbol{x}_t + \mathbf{U}_h(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h)\,, \tag{3}$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t\,, \tag{4}$$

where $\sigma(\cdot)$ denotes the logistic sigmoid function, and $\odot$ represents the element-wise multiply operator. $\mathbf{W}_{\{r,z,h\}}$ are *encoding weights*, and $\mathbf{U}_{\{r,z,h\}}$ are *recurrent weights*. $\boldsymbol{b}_{\{r,z,h\}}$ are bias terms.

B Model Details

B.1 Standard Language Modeling

For an input sequence $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, where $\boldsymbol{x}_t$ is the input data vector at time t, we define an output sequence $\mathbf{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$ with $\boldsymbol{y}_t = \boldsymbol{x}_{t+1}$ for $t = 1, \ldots, T-1$. $\boldsymbol{x}_1$ and $\boldsymbol{y}_T$ are always set to a special START and END token, respectively. The probability $p(\mathbf{Y}|\mathbf{X})$ is defined as

$$p(\mathbf{Y}|\mathbf{X}) = \prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{x}_{\leq t}) = \prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{h}_t)\,. \tag{5}$$

At each time t, there is a decoding function $p(\boldsymbol{y}_t|\boldsymbol{h}_t) = \text{softmax}(\mathbf{V}\boldsymbol{h}_t)$ to compute the distribution over words, where $\mathbf{V}$ are the *decoding weights*. The hidden states are recursively updated by $\boldsymbol{h}_t = \mathcal{H}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$, where $\mathcal{H}$ is a nonlinear function such as the LSTM or GRU defined above.

*Equal contribution. †Corresponding author.

B.2 Image Captioning

Image caption generation is considered as a conditional language modeling problem, where image features are first extracted by residual network (He et al., 2016) as a preprocessing step, and then fed into the RNN to generate the caption. Denote $\boldsymbol{z}$ as the image feature vector, using the same notation as in standard language modeling, the probablity $p(\mathbf{Y}|\mathbf{X}, \boldsymbol{z})$ is defined as

$$p(\mathbf{Y}|\mathbf{X}, \boldsymbol{z}) = \prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{x}_{\leq t}, \boldsymbol{z}) = \prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{h}_t)\,.$$

The only difference with a standard language model is that at the first time step, we use the image feature vector $\boldsymbol{z}$ to update $\boldsymbol{h}_1 = \mathcal{H}(\boldsymbol{h}_0, \boldsymbol{x}_1, \boldsymbol{z})$. $\boldsymbol{h}_0$ is set to a zero vector. The hidden states at other time steps are recursively updated by $\boldsymbol{h}_t = \mathcal{H}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$, as in standard language modeling. Note that the image feature vector $\boldsymbol{z}$ is only used to generate the first word, which works better in practice than when being used at each time step of the RNN (Vinyals et al., 2015).

C SGLD Algorithm

We list the SGLD algorithm below for clarity.

Algorithm 1: SGLD

Input: Learning rate schedule $\{\eta_t\}_{t=1}^{T}$.
Initialize: $\boldsymbol{\theta}_1 \sim \mathcal{N}(0, \mathbf{I})$;
for $t = 1, 2, \ldots, T$ **do**
  % Estimate gradient from minibatch $\mathcal{S}_m$
  $\tilde{\boldsymbol{f}}_t = \nabla \tilde{U}_t(\boldsymbol{\theta})$;
  % Parameter update
  $\boldsymbol{\xi}_t \sim \mathcal{N}(0, \eta_t \mathbf{I})$;
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \frac{\eta_t}{2}\tilde{\boldsymbol{f}}_t + \boldsymbol{\xi}_t$;
end

D Experimental Setup

For RNN training, orthogonal initialization is employed on all recurrent matrices (Saxe et al., 2014). Non-recurrent weights are initialized from a uniform distribution in $[-0.01, 0.01]$. All the bias terms are initialized to zero. It is observed that setting a high initial forget gate bias for LSTMs can give slightly better results (Le et al., 2015). Hence, the initial forget gate bias is set to 3 throughout the experiments. Word vectors are initialized with the publicly available *word2vec* vectors (Mikolov et al., 2013). These vectors have dimensionality 300 and were trained using a continuous bag-of-words architecture . Words not present in the set of pre-trained words are initialized at random. Gradients are clipped if the norm of the parameter vector exceeds 5 (Sutskever et al., 2014).

The hyperparameters for the algorithm include stepsize, minibatch size, thinning interval, number of burn-in epochs and variance of the Gaussian priors. We explain some hyperparameters that are unique in pSGLD as follows. RMSprop employs the same hyperparameter setting as pSGLD. Throughout the experiments, the dropout rate is set to 0.5.

Variance of Gaussian Prior The prior distributions on the weights of RNNs are Gaussian, with mean 0 and variance σ^2. The variance of this Gaussian distribution determines the prior belief of how strongly these weights should concentrate on 0. A larger variance in the prior leads to a wider range of weight choices, thus higher uncertainty. We set σ^2 to 1 throughout the experiments.

Burn-in To obtain a good initialization for parameter samples from regions of higher probability, we dispose of samples at the beginning of an MCMC run, prior to collection, this is called "burn-in".

Thinning Due to the fact of high autocorrelation time between samples in SG-MCMC methods, we suggest to thin the Markov chain which leaves fewer, less correlated samples. As with conventional MCMC, these thinned samples have a lower autocorrelation time and can help maintain a higher effective sample size while reducing the computational burden.

E Details of Classification Datasets

We test SG-MCMC methods on various benchmark datasets for sentence classification. Summary statistics of the datasets are in Table 1. For datasets without a standard validation set, we randomly select 10% of the training data as the validation set.

- **TREC**: This task involves classifying a question into 6 types (Li and Roth, 2002).
- **MR**: Movie reviews with one sentence per review. Classification involves predicting positive/negative reviews (Pang and Lee, 2005).
- **SUBJ**: Subjectivity dataset where the task is to classify a sentence as being subjective or objective (Pang and Lee, 2004).
- **CR**: Customer reviews of various products. This task is to predict positive/negative reviews (Hu and Liu, 2004).
- **MPQA**: Opinion polarity detection subtask of the MPQA dataset (Wiebe et al., 2005).

Table 1: Summary statistics for the datasets after tokenization. c: number of target classes. l: average sentence length. N: dataset size. $|V|$: vocabulary size. $Test$: Test set size (CV means there was no standard train/test split and thus 10-fold cross validation was used.)

Data	c	l	N	$\|V\|$	$Test$
TREC	6	10	5952	9764	500
MR	2	20	10662	18765	CV
SUBJ	2	23	10000	21322	CV
CR	2	19	3775	5339	CV
MPQA	2	3	10606	6246	CV

References

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. *SIGKDD* .

Q. V. Le, N. Jaitly, and G. E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv:1504.00941* .

X. Li and D. Roth. 2002. Learning question classifiers. *ACL* .

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *ACL* .

B. Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *ACL* .

A. M. Saxe, J. L. McClelland, and S. Ganguli. 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*.

J. Wiebe, T. Wilson, and C. Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation* .