Efficient Language Modeling with Automatic Relevance Determination in Recurrent Neural Networks

Maxim Kodryan*
Moscow State University
Moscow, Russia
maxkordn54@gmail.com

Artem Grachev*
Samsung R&D Institute,
National Research University
Higher School of Economics
Moscow, Russia
grachev.art@gmail.com

Dmitry Ignatov
National Research University
Higher School of Economics
Moscow, Russia

Dmitry Vetrov
Samsung AI Center,
National Research University
Higher School of Economics
Moscow, Russia

Abstract

Reduction of the number of parameters is one of the most important goals in Deep Learning. In this article we propose an adaptation of Doubly Stochastic Variational Inference for Automatic Relevance Determination (DSVI-ARD) for neural networks compression. We find this method to be especially useful in language modeling tasks, where large number of parameters in the input and output layers is often excessive. We also show that DSVI-ARD can be applied together with encoder-decoder weight tying allowing to achieve even better sparsity and performance. Our experiments demonstrate that more than 90% of the weights in both encoder and decoder layers can be removed with a minimal quality loss.

1 Introduction

The problem of neural networks compression has recently gained more interest as the number of parameters (and hence memory size) of modern neural networks increased drastically. Moreover, only a few weights prove to be relevant for prediction while the majority are de facto redundant (Han et al., 2015).

In this paper we suggest an adaptation of a Bayesian approach called *Automatic Relevance Determination* (ARD) for neural networks compression in language modeling tasks, where the first and the last linear layers often have enormous size. We derive the *Doubly Stochastic Variational Inference* (DSVI) algorithm for non-iid (not independent and identically distributed) objects, a common case in language modeling, and use it to perform optimization of our models.

Furthermore, we extend this approach so that it could be applied together with the weight tying technique (Press and Wolf, 2017; Inan et al., 2016), i.e., using the same set of parameters for both weight matrices of the first and the last layers, which has been proved highly efficient.

2 Related works

Most of the works on neural networks compression can be roughly divided into two categories: those dealing with matrix decomposition approaches (Lu et al., 2016; Arjovsky et al., 2016; Tjandra et al., 2017; Grachev et al., 2019) and those that leverage pruning techniques (Han et al., 2015; Narang et al., 2017). From this point of view methods based on Bayesian techniques (Louizos et al., 2017; Molchanov et al., 2017) can be considered as a more mathematically justified version of pruning.

We have focused on the pruning in application to word-level language modeling as this task usually involves a large vocabulary, hence, causing weight matrices of the first and the last layers to be huge. Chirkova et al. (2018) also consider Bayesian pruning in language modeling, though their approach is based on the Variational Dropout (VD) technique, which has been proved to be

*These two authors contributed equally; the ordering of their names was chosen arbitrarily. The work was done when the first author was an intern at the Samsung R&D Institute.

Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019), pages 40–48
Florence, Italy, August 2, 2019.

poorly theoretically justified (Hron et al., 2018), whereas ARD does not encounter these issues while maintaining similar efficacy (Kharitonov et al., 2018).

At last, as far as we are concerned, combining DSVI-ARD (or other Bayesian prunning techniques) with weight tying has not been considered previously.

3 Language modeling with neural networks

The language modeling problem is one of the important classical NLP problems and has various applications such as machine translation and text classification.

This problem is usually formulated as probabilistic prediction of a word sequence $(w_1, \dots, w_T) = (w_i)_{i=1}^T$ as follows:

$$\mathsf{P}\left((w_i)_{i=1}^T\right) = \mathsf{P}\left(w_T | \, (w_i)_{i=1}^{T-1}\right)\mathsf{P}\left((w_i)_{i=1}^{T-1}\right) = \\ = \prod_{t=1}^{T} \mathsf{P}\left(w_t | \, (w_i)_{i=1}^{t-1}\right) \approx \prod_{t=1}^{T} \mathsf{P}\left(w_t | \, (w_i)_{i=t-T_0}^{t-1}\right) \tag{1}$$

The last equation in (1) is approximated as it is almost always impossible to calculate this expression exactly for a sequence of words of arbitrary length. Therefore, the calculation is performed only within a context of a fixed size T_0.

Nowadays, approaches that perform the approximation for whole words involve different variations of RNNs (Bengio et al., 2003; Mikolov, 2012) such as LSTM or GRU (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). In word-level models the input layer maps words from the vocabulary $\mathbb{V}$ to some vector representation, and vice versa for the output layer: from a vector to a distribution over words in the vocabulary. It leads to sizes of these layers being proportional to the vocabulary size $|\mathbb{V}|$ which is tens of thousands in a typical use case.

Assume using LSTM cells as recurrent units. We can compute the total number of parameters in a network via the following formula:

$$n_{total} = 8LD^2 + 2|\mathbb{V}|D, \tag{2}$$

where L is the number of recurrent hidden layers, and D is the hidden layers size (for simplicity we let all hidden layers be of the same size). Various designs (Bengio and Senecal, 2003; Chen et al., 2016) have been proposed to reduce it, but still in word-level language modeling tasks with a significantly large vocabulary size the second term in the sum (2) makes the largest contribution. Sometimes the softmax (decoder) layer can solely occupy up to a third of the whole network memory space.

The following section describes the technique that performs efficient reduction of parameters in linear layers. This technique can be applied to the decoder layer of RNN (or to both decoder and encoder layers in tied-weight setting) providing the overall network compression with a negligible drop in quality.

4 DSVI-ARD

In this section we describe how the Doubly Stochastic Variational Inference algorithm for Automatic Relevance Determination (DSVI-ARD) originally proposed in (Titsias and Lázaro-Gredilla, 2014) can be adopted for solving multiclass classification task and, thus, leveraged in neural networks training for compressing their dense layers (in particular, decoder layers in RNNs).

4.1 Automatic Relevance Determination

We formulate the multiclass classification problem in a Bayesian framework that can provide a useful tool for feature selection — the so-called Automatic Relevance Determination (ARD).[1]

Consider a discriminative probabilistic model given a training dataset $(X, Y) = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ of N independent objects:

$$p(W, Y \mid X, \Lambda) = p(W \mid \Lambda) \prod_{n=1}^{N} p(y_n \mid W, \boldsymbol{x}_n) = \\ = \prod_{i=1}^{K}\prod_{j=1}^{D} \mathcal{N}\left(w_{ij} \mid 0, \lambda_{ij}\right) \prod_{n=1}^{N} \mathrm{Softmax}\left(W\boldsymbol{x}_n\right)_{y_n} \tag{3}$$

Here $\boldsymbol{x_n} \in \mathbb{R}^D$ is the feature vector of the n-th object, $y_n \in \{1, \dots, K\}$ is the label of the n-th object's class, $W \in \mathbb{R}^{K \times D}$ is the matrix of model parameters and $\Lambda \in \mathbb{R}^{K \times D}$ is the matrix of hyperparameters defining the prior distribution $p(W \mid \Lambda)$ over parameters W.

[1] Here we consider linear classification for simplicity, although the same model is applicable to neural networks, see subsection 4.3.

The prior distribution $p(W \mid \Lambda)$ is considered to be element-wise factorized Gaussian over each element w_{ij} (zero mean and tunable variance λ_{ij}). The likelihood function $p(y_n \mid W, \boldsymbol{x}_n)$ is the y_n-th element of the softmax vector of linear logits $W\boldsymbol{x}_n$. Such a likelihood function is quite typical in classification tasks and can be encountered in multivariate Logistic Regression.

The two goals of the subsequent Bayesian inference are, first, obtaining posterior distribution over the model parameters conditioned on the training dataset $p(W \mid X, Y, \Lambda)$, which can be leveraged in prediction on the test set, and, second, optimal model selection, i.e., hyperparameters tuning. Both problems can be solved simultaneously by maximizing the Evidence Lower Bound (ELBO):

$$\begin{aligned}\mathcal{L}(q, \Lambda) = \mathbb{E}_{W\sim q(W)} \left[\log p(Y \mid W, X)\right] - \\ - KL\left(q(W) \| p(W \mid \Lambda)\right).\end{aligned} \tag{4}$$

ELBO is a function of two variables: an arbitrary *variational distribution* over parameters $q(W)$ and model hyperparameters Λ, and can be decomposed into two parts: the data term $\mathbb{E}_{W\sim q(W)} \left[\log p(Y \mid W, X)\right]$ and the negative KL-divergence between the variational approximation $q(W)$ and the prior distribution $p(W \mid \Lambda)$ (KL-term) $-KL\left(q(W) \| p(W \mid \Lambda)\right)$. We further show that maximization of ELBO with respect to both q and Λ solves the model selection problem while also fitting q to the posterior.

ELBO has several useful properties such as $\mathcal{L}(q, \Lambda) \leq \log p(Y \mid X, \Lambda), \forall q, \Lambda$ (bounds the evidence logarithm $\log p(Y \mid X, \Lambda)$ from below) and $\mathcal{L}(q, \Lambda) = \log p(Y \mid X, \Lambda)$ if and only if $q(W) = p(W \mid X, Y, \Lambda)$, so maximization of ELBO with respect to q for fixed Λ is equivalent to fitting q to the posterior $p(W \mid X, Y, \Lambda)$, hence solving the first of the mentioned Bayesian inference problems.

Maximization of the evidence $p(Y \mid X, \Lambda)$ with respect to hyperparameters Λ is a well-known Bayesian model selection method, also known as empirical Bayes estimation (Carlin and Louis, 1997). A model with the highest evidence is considered to be "the best" in terms of both data fit and model complexity. Evidence maximization can be performed via ELBO maximization as

$$\max_{\Lambda} \log p(Y \mid X, \Lambda) = \max_{q, \Lambda} \mathcal{L}(q, \Lambda). \tag{5}$$

Finally, this double maximization procedure, as we have shown above, handles the model selection problem while also fitting q to the posterior.

From the view of the ELBO functional it is clear that only the KL-term $KL\left(q(W) \| p(W \mid \Lambda)\right)$ depends on Λ, hence maximization of ELBO with respect to Λ is equivalent to minimization of the KL-term with respect to Λ.

Now we restrict the variational distribution q to the factorized Gaussian:

$$q(W \mid \boldsymbol{\mu}, \boldsymbol{\sigma}) = \prod_{i=1}^{K} \prod_{j=1}^{D} \mathcal{N}(w_{ij} \mid \mu_{ij}, \sigma_{ij}^2), \tag{6}$$

where $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^{K\times D}$ are the *variational parameters*.

This way ELBO maximization (or equivalently, KL-term minimization) with respect to Λ can be performed analytically with the solution at

$$\lambda_{ij}^* = \mu_{ij}^2 + \sigma_{ij}^2. \tag{7}$$

After substituting Λ^* from (7) into the ELBO equation (4) and taking into account the variational family restriction (6) we can rewrite the maximization problem (5) as follows:

$$\begin{aligned}\mathbb{E}_{W\sim q(W|\boldsymbol{\mu},\boldsymbol{\sigma})} \left[\log p(Y \mid W, X)\right] + \\ + \frac{1}{2} \sum_{i=1}^{K} \sum_{j=1}^{D} \log \frac{\sigma_{ij}^2}{\mu_{ij}^2 + \sigma_{ij}^2} \longrightarrow \max_{\boldsymbol{\mu},\boldsymbol{\sigma}}\end{aligned} \tag{8}$$

This equation (8) is the final form of the ARD ELBO maximization problem. We can see that the first term (data term) induces the variational parameters to describe the observed data well by sharpening the variational distribution at the maximum likelihood point, while the second term (KL-term) makes irrelevant parameters shrink. The mutual maximization of both terms leads to a sparse solution (in the limit), at which all redundant features are zeroed. The following subsections describe how it can be performed in practice, especially in application to recurrent neural networks.

4.2 DSVI

The Doubly Stochastic Variational Inference (DSVI) is a method of stochastic gradient maximization of ELBO with respect to the variational parameters. We provide the standard DSVI-ARD description in Algorithm 1. At each iteration two types of random variables are sampled: a mini-batch of objects $\{\boldsymbol{x}_m, y_m\}_{m=1}^{M} \subseteq (X, Y)$

and a set of "proto-weights" for each object in the mini-batch $\boldsymbol{\epsilon}_m \sim \mathcal{N}(\mathbf{0}, I), \boldsymbol{\epsilon}_m \in \mathbb{R}^{K\times D}$, which are used to obtain stochastic gradients of the log-likelihood with respect to the variational parameters via the reparametrization trick (RT) (Kingma et al., 2015). DSVI does not depend on a specific form of the log-likelihood function $\log p(y \mid W, x)$, but only requires its gradient $\nabla_W \log p(y \mid W, x)$, so the same procedure is applicable for different models with differentiable log-likelihoods. DSVI can also be regarded as efficient SGD minimization of the negative ELBO loss functional (8), which consists of the data term and the KL-regularizer.

Algorithm 1 Doubly stochastic variational inference

Input: log-likelihood $\log p(y \mid W, \mathbf{x})$, training dataset (X, Y) of size N, learning rates $\{\rho_k\}$, mini-batch size M
Initialize the variational parameters $\boldsymbol{\mu}^{(0)}, \boldsymbol{\sigma}^{(0)}$, $k = 0$
repeat
  $k = k + 1$
  Sample a mini-batch $\{\boldsymbol{x}_m, y_m\}_{m=1}^M \subseteq (X, Y)$
  for all objects $(\boldsymbol{x}_m, y_m)$ in the mini-batch **do**
    Sample $\boldsymbol{\epsilon}_m \sim \mathcal{N}(\mathbf{0}, I), \boldsymbol{\epsilon}_m \in \mathbb{R}^{K\times D}$
    $W_m \coloneqq \boldsymbol{\mu}^{(k-1)} + \boldsymbol{\sigma}^{(k-1)} \odot \boldsymbol{\epsilon}_m$
  end for
  $g_{\boldsymbol{\mu}}^{Data} \coloneqq \frac{N}{M}\sum_{m=1}^M \nabla_{\boldsymbol{\mu}} \log p(y_m \mid W_m, \boldsymbol{x}_m)$
  $g_{\boldsymbol{\sigma}}^{Data} \coloneqq \frac{N}{M}\sum_{m=1}^M \nabla_{\boldsymbol{\sigma}} \log p(y_m \mid W_m, \boldsymbol{x}_m)$
  $g_{\boldsymbol{\mu}}^{KL} \coloneqq \nabla_{\boldsymbol{\mu}} \left[\frac{1}{2}\sum_{i=1}^K\sum_{j=1}^D \log\frac{\sigma_{ij}^2}{\mu_{ij}^2+\sigma_{ij}^2}\right]\Big|_{\substack{\boldsymbol{\mu}=\boldsymbol{\mu}^{(k-1)}\\ \boldsymbol{\sigma}=\boldsymbol{\sigma}^{(k-1)}}}$
  $g_{\boldsymbol{\sigma}}^{KL} \coloneqq \nabla_{\boldsymbol{\sigma}} \left[\frac{1}{2}\sum_{i=1}^K\sum_{j=1}^D \log\frac{\sigma_{ij}^2}{\mu_{ij}^2+\sigma_{ij}^2}\right]\Big|_{\substack{\boldsymbol{\mu}=\boldsymbol{\mu}^{(k-1)}\\ \boldsymbol{\sigma}=\boldsymbol{\sigma}^{(k-1)}}}$
  $g_{\boldsymbol{\mu}} \coloneqq g_{\boldsymbol{\mu}}^{Data} + g_{\boldsymbol{\mu}}^{KL}$
  $g_{\boldsymbol{\sigma}} \coloneqq g_{\boldsymbol{\sigma}}^{Data} + g_{\boldsymbol{\sigma}}^{KL}$
  $\boldsymbol{\mu}^{(k)} = \boldsymbol{\mu}^{(k-1)} + \rho_k g_{\boldsymbol{\mu}}$
  $\boldsymbol{\sigma}^{(k)} = \boldsymbol{\sigma}^{(k-1)} + \rho_k g_{\boldsymbol{\sigma}}$
until convergence criterion is met

4.3 DSVI-ARD in Recurrent Neural Networks

As was noted above, DSVI can be applied to any probabilistic ARD model with differentiable likelihood. A neural network with a softmax layer introduces a likelihood function similar to the one considered in (3). Hence, we suggest replacing the softmax output layer with the ARD layer for multiclass classification and train it with the DSVI algorithm computing its log-likelihood gradients via backpropagation due to the usage of the RT.

When training RNN with a DSVI-ARD layer as a decoder (softmax layer in this case) we encounter the question of sampling strategy for parameters: one sample per object or once for the whole mini-batch of objects. The first strategy is typical for standard classification tasks and is implemented in the classical DSVI algorithm 1. The second one is more justified in the RNN case because objects in one sequence (mini-batch) are not independent and should better be processed with the same weights. We propose Algorithm 2, which is applicable in the case of non-iid objects in a mini-batch. Summing it up, it differs from the standard DSVI only in that the "proto-weights" $\boldsymbol{\epsilon} \in \mathbb{R}^{K\times D}$ are sampled once for the whole mini-batch at each iteration.

Algorithm 2 Doubly stochastic variational inference for non-independent data

Input: log-likelihood $\log p(y \mid W, \mathbf{x})$, training dataset (X, Y) of size N, learning rates $\{\rho_k\}$, mini-batch size M
Initialize the variational parameters $\boldsymbol{\mu}^{(0)}, \boldsymbol{\sigma}^{(0)}$, $k = 0$
repeat
  $k = k + 1$
  Sample a mini-batch $\{\boldsymbol{x}_m, y_m\}_{m=1}^M \subseteq (X, Y)$ of non-iid objects
  Sample one $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I), \boldsymbol{\epsilon} \in \mathbb{R}^{K\times D}$
  $W \coloneqq \boldsymbol{\mu}^{(k-1)} + \boldsymbol{\sigma}^{(k-1)} \odot \boldsymbol{\epsilon}$
  $g_{\boldsymbol{\mu}}^{Data} \coloneqq \frac{N}{M}\sum_{m=1}^M \nabla_{\boldsymbol{\mu}} \log p(y_m \mid W, \boldsymbol{x}_m)$
  $g_{\boldsymbol{\sigma}}^{Data} \coloneqq \frac{N}{M}\sum_{m=1}^M \nabla_{\boldsymbol{\sigma}} \log p(y_m \mid W, \boldsymbol{x}_m)$
  $g_{\boldsymbol{\mu}}^{KL} \coloneqq \nabla_{\boldsymbol{\mu}} \left[\frac{1}{2}\sum_{i=1}^K\sum_{j=1}^D \log\frac{\sigma_{ij}^2}{\mu_{ij}^2+\sigma_{ij}^2}\right]\Big|_{\substack{\boldsymbol{\mu}=\boldsymbol{\mu}^{(k-1)}\\ \boldsymbol{\sigma}=\boldsymbol{\sigma}^{(k-1)}}}$
  $g_{\boldsymbol{\sigma}}^{KL} \coloneqq \nabla_{\boldsymbol{\sigma}} \left[\frac{1}{2}\sum_{i=1}^K\sum_{j=1}^D \log\frac{\sigma_{ij}^2}{\mu_{ij}^2+\sigma_{ij}^2}\right]\Big|_{\substack{\boldsymbol{\mu}=\boldsymbol{\mu}^{(k-1)}\\ \boldsymbol{\sigma}=\boldsymbol{\sigma}^{(k-1)}}}$
  $g_{\boldsymbol{\mu}} \coloneqq g_{\boldsymbol{\mu}}^{Data} + g_{\boldsymbol{\mu}}^{KL}$
  $g_{\boldsymbol{\sigma}} \coloneqq g_{\boldsymbol{\sigma}}^{Data} + g_{\boldsymbol{\sigma}}^{KL}$
  $\boldsymbol{\mu}^{(k)} = \boldsymbol{\mu}^{(k-1)} + \rho_k g_{\boldsymbol{\mu}}$
  $\boldsymbol{\sigma}^{(k)} = \boldsymbol{\sigma}^{(k-1)} + \rho_k g_{\boldsymbol{\sigma}}$
until convergence criterion is met

We also consider applying DSVI-ARD in a tied-weight setting. For that we slightly change the

model so that both the encoder and decoder layers contribute into the likelihood via the same set of weights. Now (in the non-iid DSVI algorithm 2) the same set of parameters (weight matrix) is sampled for both layers, and their gradients with respect to the variational parameters are summed to obtain the mutual gradient of $\log p(y \mid W, \boldsymbol{x})$ for the data term update g^{Data}. The KL-term remains the same as neither new random variables are added to the model nor its prior distribution or variational approximation changes. The only thing that varies is the likelihood of the model, i.e., the data term: now the encoder is also conditioned on the variational parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. This basically means that the gradients w.r.t. the encoder's weights are propagated back to the variational parameters.

5 Experiments

We have conducted several experiments to test the DSVI-ARD compression approach in language modeling. We used LSTM and LSTM with tied weights models from (Zaremba et al., 2014; Inan et al., 2016) respectively as our baselines: the experiments involved the same LSTM architecture with two hidden layers of size 650 and two datasets: PTB (Mikolov et al., 2010) and Wikitext2 (Merity et al., 2016); also each mini-batch of objects was constructed from bs word sequences ($bs = 10$ and $bs = 20$ for evaluation and training respectively) of length $bptt = 35$.

We applied dropout after the embedding (except for the tied-weight ARD models because ARD can be regarded as a special form of regularization by itself) and hidden layers, with a dropout rate as a hyperparameter. We used stochastic gradient descent (SGD) as an optimization procedure, with adaptive learning rate decreasing from the starting value by a multiplicative factor (both are hyperparameters) each time validation perplexity has stopped improving.

We also compared our approach to other compression techniques: matrix decomposition-based (Grachev et al., 2019) and VD-based (Chirkova et al., 2018). For the last one we used a similar model: a network with one LSTM layer of 256 hidden units.

5.1 Training and evaluation

The whole set of parameters of a model with DSVI-ARD layers can be divided into the variational parameters $\boldsymbol{\mu}, \boldsymbol{\sigma}$ and all the other network parameters (including biases of the DSVI-ARD layers). Variational optimization is performed with the DSVI-ARD algorithm, which, in turn, only requires gradients of the log-likelihood and KL-divergence. Therefore, overall model training is a standard gradient optimization of parameters based on backpropagation (specifically, BPTT in the RNN case) with negative ELBO as the loss function.

For more efficient training we applied the KL-cost annealing technique (Sønderby et al., 2016). The idea is to multiply the KL-term in ELBO by a variable weight, called the KL-weight, at training time. The weight gradually increases from zero to one during the first several epochs of training. This technique allows achieving better final performance of the model because such a train-

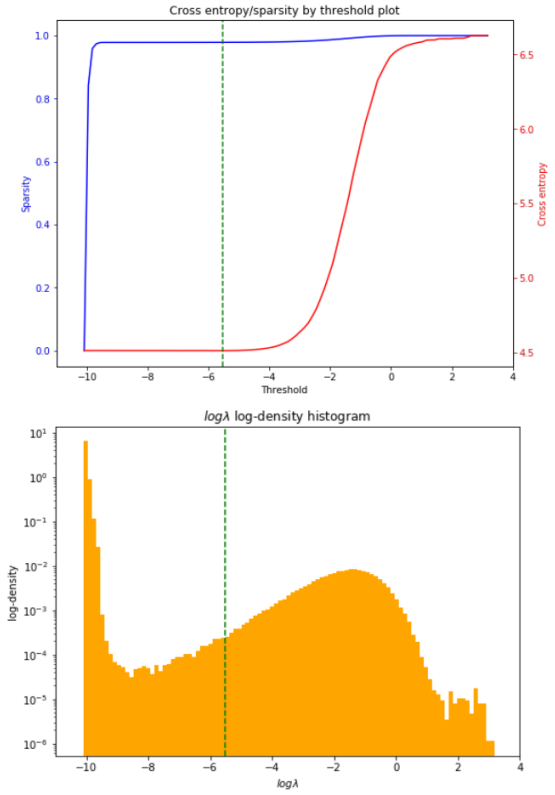

Figure 1: Plots of validation cross-entropy (red line) of a LSTM model with a DSVI-ARD softmax layer on the PTB dataset and its corresponding sparsity (blue line) for different possible threshold $\log \lambda_{thresh}$ values (**top**) and the distribution histogram of its prior log-variances $\log \lambda_{ij}$ (**bottom**). We display the density on a **log scale** due to a very sparse distribution. The threshold chosen for further model evaluation (the best in terms of perplexity on the validation set) $\log \lambda^{opt}_{thresh}$ is marked with a green dashed line.

Original model	Dataset	Architecture	Number of parameters, M (Full / Softmax)	Removed parameters, % (Full / Softmax)	Perplexity	Accuracy, %
LSTM, (Zaremba et al., 2014)	*PTB*	*Original*	19.8 / 6.5	No compression	80.85	27.4%
		DSVI-ARD (ours)	**13.4** / **0.14**	**32.1%** / **97.8%**	91.84	27.2%
		LR for Softmax, (Grachev et al., 2019)	14.5 / 1.19	26.8 % / 81.7 %	84.12	N/A
		TT for Softmax, (Grachev et al., 2019)	14.3 / 1.03	27.8 % / 84.2 %	88.55	N/A
	Wikitext2	*Original*	50.1 / 21.6	No compression	94.27	27.5%
		DSVI-ARD (ours)	**28.9** / **0.43**	**42.3%** / **98.0%**	103.54	**27.6%**
LSTM + tied weights, (Inan et al., 2016)	*PTB*	*Original*	13.3 / 6.5	No compression	75.68	27.7%
		DSVI-ARD (ours)	**7.4** / **0.66**	**44.0%** / **89.9%**	82.27	27.3%
	Wikitext2	*Original*	28.4 / 21.6	No compression	86.62	27.9%
		DSVI-ARD (ours)	**8.7** / **1.94**	**69.3%** / **91.0%**	87.36	**28.1%**
LSTM, (Chirkova et al., 2018)	*PTB*	*Original*	5.64 / 2.56	No compression	129.3	N/A
		VD, (Chirkova et al., 2018)	3.2 / 0.12	43.3 % / 95.5 %	109.2	N/A
		DSVI-ARD (Ours)	**3.18** / **0.1**	**43.6 %** / **96.1 %**	**106.2**	25.9 %

Table 1: Language modeling experiments results. We provide the number of parameters left after pruning (in millions) and the achieved compression ratios (in percents) of the whole network and the softmax layer alone along with the final quality (perplexity and accuracy) on the test set for each evaluated model. The original (uncompressed) models quality is provided for comparison.

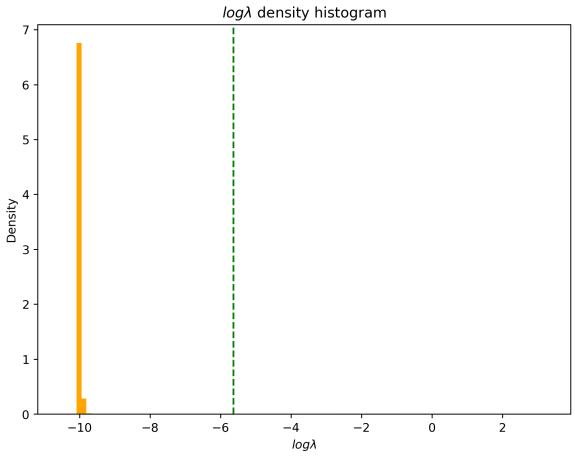

Figure 2: Distribution histogram of the prior log-variances $\log \lambda_{ij}$ obtained for a LSTM model with a DSVI-ARD softmax layer on the PTB dataset. We provide the standard-scaled density to justify the usage of a log scale in Fig. 1 (**bottom**).

ing procedure can be considered as pre-training on data (when the data term in ELBO dominates) and then starting fair optimization of the true ELBO (when the KL-weight reaches one). We used a simple linear KL-weight increasing strategy with a step selected as a hyperparameter.

During the evaluation of our models we do not sample parameters as we do in the training phase but instead set the approximated posterior mean $\boldsymbol{\mu}$ as DSVI-ARD layers weights. Then we zero out the weights with the corresponding logarithms of prior variances lower than a certain threshold $\log \lambda_{thresh}$ (a hyperparameter selected on validation):

$$\log \lambda^*_{ij} < \log \lambda_{thresh} \Rightarrow \mu_{ij} := 0. \qquad (9)$$

This procedure essentially provides the desired sparsity as redundant weights are being literally removed from the network.

Each experiment was conducted as follows. We trained several models for some number of epochs with different hyperparameter initialization (such as dropout rate, learning rate, etc.). Then we picked the best model in terms of cross-entropy (log-perplexity) on the validation set at the last training epoch. We did not zero weights during evaluation at this phase, in other words, $\log \lambda_{thresh} = -\infty$ in equation (9). After that, we started threshold selection for the picked model: we iterated over possible values of $\log \lambda_{thresh}$ from the "leave-all" to the "remove-all" extreme values and chose the one (denoted by $\log \lambda^{opt}_{thresh}$) at which the best validation perplexity was obtained. Finally, we evaluated the model on the test set using the chosen optimal threshold $\log \lambda^{opt}_{thresh}$.

In our results we report the achieved compression ratio $cr = \frac{\sum_{i=1}^{K}\sum_{j=1}^{D} \mathbb{1}[\log \lambda^*_{ij} < \log \lambda^{opt}_{thresh}]}{KD}$, perplexity and accuracy[2] on the test set.

5.2 Results

Table 1 concludes all the results obtained during our experiments.

The comparison of DSVI-ARD with other dense layers compression approaches revealed

[2]By accuracy, we mean the fraction of correctly predicted words. The prediction is performed by taking the argmax of the softmax distribution over vocabulary words.

that our models can exhibit comparable perplexity quality while achieving much higher compression (in Grachev et al. (2019) case) and even surpass models based on similar Bayesian compression techniques (in Chirkova et al. (2018) case).

Also it can be seen that encoder-decoder weight tying helps to obtain higher overall compression (from almost 45% to 70% reduction of all model weights), due to a smaller number of parameters in the whole network, and even better results in both perplexity and accuracy on both datasets. Quality improvement after weight tying is a common case, however, we see that it helps to especially enhance the performance of our models (perplexity drops by almost 10 points in PTB case and more than 16 points in Wikitext2 case), which gives grounds for the proposed combination of DSVI-ARD and weight tying.

One can argue that DSVI-ARD may lead to overpruning (Trippe and Turner, 2018) because in all our experiments (except the last one, comparing with Chirkova et al. (2018) results) a slight quality drop in terms of perplexity can be observed. However, we specifically provide the test accuracy as well, in terms of which we achieve comparable or even better results than original models. We suggest that this effect might be caused by prediction uncertainty (or entropy) increase rather than model quality deterioration.

Fig. 1 demonstrates plots of cross-entropy (log-perplexity) and sparsity for different thresholds $\log \lambda_{thresh}$ — essentially the compression-quality trade-off plot — and the log-scaled distribution histogram of the decoder layer's prior log-variances $\log \lambda_{ij}^{*}$ obtained for a DSVI-ARD LSTM model trained on the PTB dataset. We also provide the same distribution on the standard scale (Fig. 2) for comparison. The dashed green line denotes the value of the chosen threshold $\log \lambda_{thresh}^{opt}$ which provides the best validation perplexity. We can observe that the overwhelming majority of weights in the last layer are indeed redundant, i.e., have small prior variances, do not contribute to the inference, and can be removed without harming much model performance. We argue that DSVI-ARD eliminates weights that obstruct generalization while leaving only those actually necessary for correct prediction.

6 Conclusion

In this paper we adopted the DSVI-ARD algorithm for compressing recurrent neural networks. Our main contributions are extending DSVI-ARD to the case of non-iid objects and combining it with the weight tying technique. In our experiments involving LSTM networks in language modeling tasks, we have managed to obtain substantially high compression ratios at an acceptable quality loss. The proposed method turned out to be comparable to or even surpassing other compression techniques like matrix decomposition and variational dropout.

There are several possible avenues for future work. An intriguing application of the proposed DSVI-ARD method for RNNs is the compression of current state-of-the-art models (Yang et al., 2017; Dai et al., 2019), which require enormous amounts of memory and computational resources. At the same time, one of the drawbacks of the current Bayesian compression approaches is a lack of their expressive ability, i.e., most of them are based on oversimplified posterior approximations and prior distributions (e.g., factorized Gaussian), which may lead to overly rough estimates and overall model inefficiency. A rigorous study of this problem is required. Another possible direction is bringing Bayesian framework into matrix decomposition-based methods as well. This fusion may lead to more effective and justified compression techniques.

Acknowledgments

The article was prepared within the framework of the HSE University Basic Research Program and funded by the Russian Academic Excellence Project '5-100'.

References

Martín Arjovsky, Amar Shah, and Yoshua Bengio. 2016. Unitary evolution recurrent neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1120–1128.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio and Jean-Sébastien Senecal. 2003. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the Ninth In-*

ternational Workshop on Artificial Intelligence and Statistics, AISTATS 2003, Key West, Florida, USA, January 3-6, 2003.

Bradley P. Carlin and Thomas A. Louis. 1997. BAYES AND EMPIRICAL BAYES METHODS FOR DATA ANALYSIS. *Statistics and Computing*, 7(2):153–154.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Nadezhda Chirkova, Ekaterina Lobacheva, and Dmitry P. Vetrov. 2018. Bayesian compression for natural language processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2910–2915.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860.

Artem M. Grachev, Dmitry I. Ignatov, and Andrey V. Savchenko. 2019. Compression of recurrent neural networks for efficient language modeling. *Applied Soft Computing*, 79:354 – 362.

Song Han, Huizi Mao, and William J. Dally. 2015. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jiri Hron, Alexander G. de G. Matthews, and Zoubin Ghahramani. 2018. Variational bayesian dropout: pitfalls and fixes. *CoRR*, abs/1807.01969.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.

Valery Kharitonov, Dmitry Molchanov, and Dmitry Vetrov. 2018. Variational dropout via empirical bayes. *CoRR*, abs/1811.00596.

Diederik P. Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. *CoRR*, abs/1506.02557.

Christos Louizos, Karen Ullrich, and Max Welling. 2017. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298.

Zhiyun Lu, Vikas Sindhwani, and Tara N. Sainath. 2016. Learning compact recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 5960–5964.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Tomáš Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry P. Vetrov. 2017. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2498–2507.

Sharan Narang, Erich Elsen, Gregory Diamos, and Shubho Sengupta. 2017. Exploring sparsity in recurrent neural networks. *arXiv preprint arXiv:1704.05119*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 157–163.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. How to train deep variational autoencoders and probabilistic ladder networks. *CoRR*, abs/1602.02282.

Michalis K. Titsias and Miguel Lázaro-Gredilla. 2014. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1971–1979.

Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2017. Compressing recurrent neural network with tensor train. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 4451–4458.

Brian Trippe and Richard Turner. 2018. Overpruning in variational bayesian neural networks. *arXiv preprint arXiv:1801.06230*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2017. Breaking the softmax bottleneck: A high-rank RNN language model. *CoRR*, abs/1711.03953.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.