ADAPTIVE DROPOUT WITH RADEMACHER COMPLEXITY REGULARIZATION

A PREPRINT

Sergey Makarychev
Department Of Data Science
Skolkovo Institute Of Science And Technology
Moscow
sergei.makarychev@skoltech.ru

Aleksandr Rozhnov
Department of Data Science
Skolkovo Institute Of Science And Technology
Moscow
aleksandr.rozhnov@skoltech.ru

October 26, 2018

***K*eywords** Variational Dropout, Rademacher Regularization

1 Introduction

In this work we consider Rademacher Variational Dropout technique, described in ADAPTIVE DROPOUT WITH RADEMACHER COMPLEXITY REGULARIZATION article of Ke Zhai and Huan Wang. They propose a novel framework to adaptively adjust the dropout rates for the deep neural network based on a Rademacher complexity bound. The state-of-the-art deep learning algorithms impose dropout strategy to prevent feature co-adaptation. However, choosing the dropout rates remains an art of heuristics or relies on empirical grid-search over some hyperparameter space. In this work, the authors show that the network Rademacher complexity is bounded by a function related to the dropout rate vectors and the weight coefficient matrices. Interestingly, these notes lead to extremely sparse solutions both in fullyconnected and convolutional layers in some problems. We provide several experiments which will be evidence of this fact.

2 Background

Deep neural networks (DNNs) are a widely popular family of models which is currently state-of-the-art in many important problems. However, DNNs often have many more parameters than the number of the training instances. This makes them prone to overfitting. There exist a number of different techniques which allow to avoid overfitting, one of them is regularization. One adds a term, which depend on parameters of the model and optimize objective which now includes also a term which does not allow it to become very complicated in some sense. In the article of our interest authors introduce Rademacher regularization term, that measure in some sense complexity of the model and don't allow weights vary to much. Besides this approach there are some other, which can lead to same behavior of the model. For example a commonly used regularizer is Binary Dropout (Hinton et al., 2012) that prevents co-adaptation of neurons by randomly dropping them during training. An equally effective alternative is Gaussian Dropout (Srivastava et al., 2014)[1] that multiplies the outputs of the neurons by Gaussian random noise. Dropout requires specifying the dropout rates which are the probabilities of dropping a neuron. The dropout rates are typically optimized using grid search, but this approach leads to exponential asymptotic when the number of parameters increase. So, we need to find a scalable method which will find the demanded dropout rates.

This Rademacher complexity bound regularizer provides a lot of flexibility and advantage in modeling and optimization. First, it combines the model complexity and the loss function in an unified objective. This offers a viable way to trade-off the model complexity and representation power through the regularizer weighting coefficient. Second, since this bound is a function of dropout probabilities, it is possible to incorporate them explicitly into the computation graph of the optimization procedure. One can then adaptively optimize the objective and adjust the dropout probabilities throughout training in a way similar to ridge regression and the lasso (Hastie et al., 2009) [2]. Third, proposed regularizer assumes a neuron-wise dropout manner and models different neurons to have different retain rates during the optimization. To

the best of our knowledge, the idea behind the idea of adaptive tuning of dropout rates is not the new. More important to go deeper in some sparse systems, as they effectively Recently, Molchanov et al. (2017)[3] extend the variational dropout method with a tighter approximation which subsequently produce more sparse dropout rates.

3 Problem formulation

We will consider the classification task, where k denotes number of classes. Let us denote by $\mathbb{S} = \{(x_i, y_i)|i \in \{1, \dots n), x_i \in \mathbb{R}^d, y_i \in 0, 1^k\}$ the training sample. We consider Neural Network and below some notation, corresponding to the theory of Neural networks.

- k^l - the number of neurons of the l^{th} layer, $W^l \in \mathbb{R}^{k^{l-1} \times k^l}$ - the linear coefficient matrix from $(l-1)^{th}$ layer to the l^{th} layer.
- Forward step: $f^l(x; W^{:l}, r^{:l}) = (r^{l-1} \odot \phi(f^{l-1}(x; W^{:l-1}, r^{:r-1})))W^l$, where $\phi : \mathbb{R} \to \mathbb{R}^+$ is ReLU.
- We use expected value of $f^L(x; W, \theta)$ as the deterministic output: $f^L(x; W, r) = \mathbb{E}_r[f^L(x; W, r)]$
- Loss function: $loss(f^L(x; W, \theta), y)) = -\sum_j y_j \log \frac{e^{f_j^L(x;W,\theta)}}{\sum_j e^{f_j^L(x;W,\theta)}}$

4 Empirical Rademacher Complexity

In this section we will give the brief explanation of the Rademacher Complexity, analyze general idea between Neural Networks and such kind of regularizers.

Definition: The empirical Rademacher complexity of function class $\mathbb{F}$ with respect to the sample $\mathbb{S}$ is

$$R_{\mathbb{S}}(\mathbb{F}) = \frac{1}{n}\mathbb{E}_{\sigma_i}\left[sup_{f\in\mathbb{F}}\sum_{i=1}^{n}\sigma_i f(s_i)\right],$$

where $\sigma_1, \dots \sigma_n$ are independent random variable drawn from the Rademacher distribution:

$$\mathbb{P}(\sigma_i = +1) = \frac{1}{2}, \quad \mathbb{P}(\sigma_i = -1) = \frac{1}{2}$$

Let's go through this abstract definition in more details. One may have the following intuition of Radamecher complexity (RC).

- RC Measures richness of a class of real-valued functions with respect to a probability distribution;
- Maximum covariance of function from the class with the random Rademacher noise.

To better obtain the idea behind this mathematical object, we should consider some special cases of Rademacher complexity. Consider linear class (Shalev-Shwartz and Ben-David, 2014):

$$H_2 = \{x \to \langle w, x\rangle : ||w||_2 \leq B_2\}$$

In this case, the following holds:

$$R_{\mathbb{S}} \leq \max_i ||x_i||_2 B_2/\sqrt{n}$$

We may interpret the regularizer in the ridge regression related an upper bound for the empirical Rademacher complexity of the linear function class. Similarly, we may consider the following class:

$$H_1 = \{x \to \langle w, x\rangle : \|w\|_1 \leq B_1\}$$

In this case, the following holds:

$$R_{\mathbb{S}} \leq \max_i \|x_i\|_\infty B^1 \sqrt{\frac{2\log 2d}{n}}$$

So the lasso problem can also be viewed as adding a Rademacher-related regularization to the empirical loss minimization objective. For further explanations we need to give some additional definitions:

Theorem. Let $X \in \mathbb{R}^{n\times d}$ be the sample matrix with i^{th} row $x_i \in \mathbb{R}^d, p \geq q, \frac{1}{p}+\frac{1}{q}=1$. If the p-norm of every column of W^l is bounded by a constant B^l, denote $\mathbb{W} = \{W | \max_j ||W_j^l|| \leq B^l, \quad \forall l \in \{1,2,\ldots\}\}$, given θ, the empirical Rademacher complexity of the loss for the dropout neural network defined above is bounded by:

$$R_{\mathbb{S}}\left(loss \circ f^L\right) = \frac{1}{n}\mathbb{E}_{\sigma_i}\left[\sup_{W\in\mathbb{W}}\sum_{i=1}^{n}\sigma_i loss(f^L(x_i;W,\theta),y_i)\right] \leq\leq k\cdot 2^L\sqrt{\frac{2\log 2d}{n}}||X||_{max}\left(\prod_{i=1}^{L}B^l||\theta^{l-1}||_1^{1/q}\right)$$

Now we will discuss how to incorporate in the model this type of regularizer. Previous theorem makes it possible to unify the dropout rates and the network coefficients in one objective:

$$Obj(W,\theta) = Loss\left(\mathbb{S}, f^L(\cdot;W,\theta)\right) + \lambda Reg\left(\mathbb{S},W,\theta\right).$$

$$Loss\left(\mathbb{S}, f^L(\cdot;W,\theta)\right) = \frac{1}{n}\sum_{(x_i,y_i)\in\mathbb{S}} loss(f^L(x_i;W,\theta),y_i)$$

$$Reg\left(\mathbb{S},W,\theta\right) = k\cdot 2^L\sqrt{\frac{\log d}{n}}||X||_{max}\left(\prod_{i=1}^{L}||\theta^{l-1}||_1^{1/q} max_j||W_j^l||_p\right)$$

For stability purposes authors designed some heuristics to scale the regularizer to offset the multipler effects raised from network structure. They suggested to use the following stable version of Radamacher regularizer:

$$k2^L\sqrt{\frac{\log d}{n}}\max_i||x_i||_\infty\left(\prod_{i=1}^{L}||W^l||_{max}||\theta^{l-1}||_1\frac{\sqrt{k^l+k^{l-1}}}{k^l}\right),$$

where k^{l-1} and k^l the number of neurons in $l-1$-th and l-th layers of NN.

5 Problem details

Since we decided to solve the problem of choosing the correct Bernoulli parameters for retain rates via Rademacher regularization, we should discuss what problems we face in this approach. Introducing for each dropout neuron its own Bernouli parameter leads us to probabilistic neural network, i.e. our output is not a number, but a distribution over retain rates. If we want to obtain the exact value we should compute expectation with respect to the distribution's parameters. Unfortunately, this expectation has no closed form formula, thus we need to estimate it using Monte-Carlo. It can lead to problems if the number of parameters if high, what is definitely true for deep neural networks. Authors of the article besides exact Monte-Carlo sampling proposed to assign retain rates as expectation of their parameters. This will give us approximation of expectation less accurate then Monte-Carlo, however the computational speed increases drastically. So, using this approach we try to solve the following problems:

- Since the output of neural network is probabilistic, we estimate predicted target value using expectation. This leads to exponential growth of computations for each input sample with growing number of parameters.
- We cannot compute gradients with respect to distribution parameters directly.

6 Experiments

In this section we discuss experiments conducted to assess the properties of Rademacher regularization.

6.1 Experiment 1.

We download MNIST data set from Dropbox storage and parsed it to obtain train, test and validation samples. After that we fitted two neural networks, which were described in article. The architecture of those networks was the following:

1. $784 \overset{Dense}{\to} 1024 \overset{VarDropout}{\to} 1024 \overset{ReLU}{\to} 1024 \overset{Dense}{\to} 10 \overset{Loss}{\to} loss$
2. $784 \overset{Dense}{\to} 800 \overset{VarDropout}{\to} 800 \overset{ReLU}{\to} 800 \overset{Dense}{\to} 800 \overset{VarDropout}{\to} 800 \overset{ReLU}{\to} 800 \overset{Dense}{\to} 10 \overset{Loss}{\to} loss$

Dense means fully connected layer of the correct shape, Var. Dropout means Variational Dropout Layer which is described in the main article. The result of this experiment coincide with the result obtained in the article. You can see the learning curves in the Fig. 1.

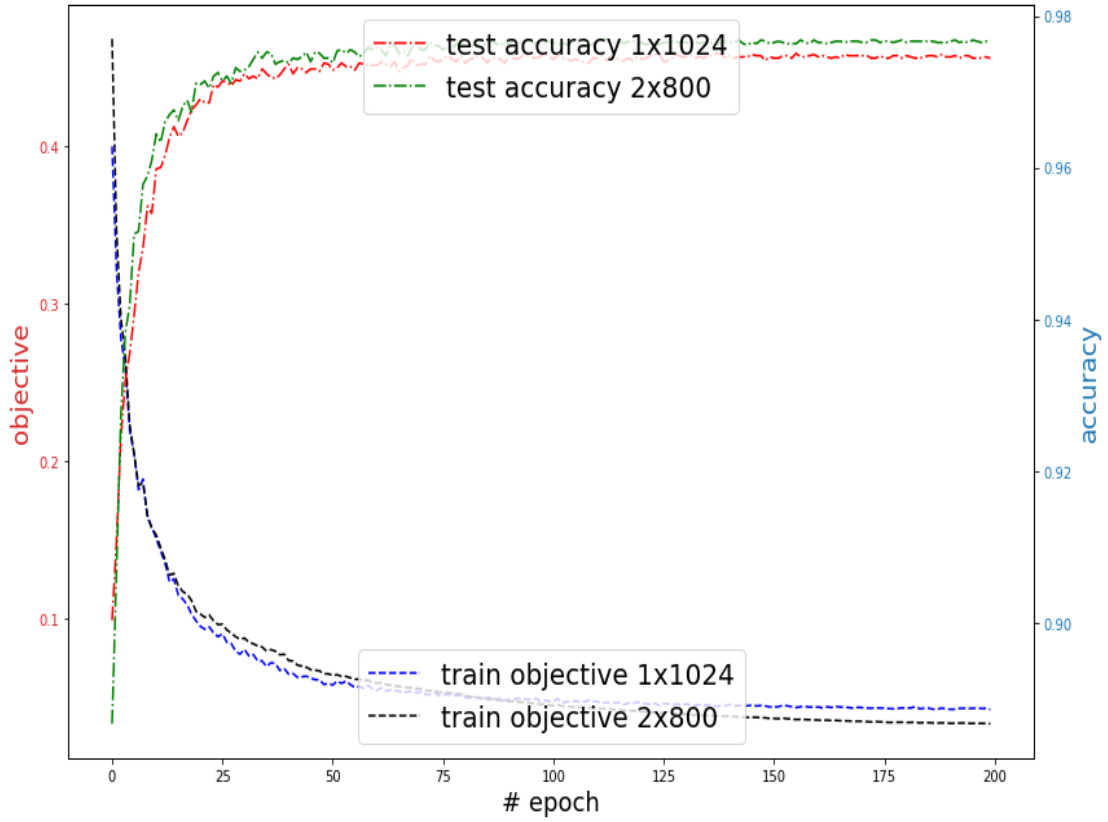

Figure 1: Train objective and validation accuracy on MNIST data

One can see that we obtain very good quality on validation data. Moreover, the results are the same as obtained in article. So, in such simple case the behavior of our implementation and authors' is the same. Now we consider more challenging experiments.

6.2 Experiment 2.

For this experiment we consider the following Neural Network architecture

1. $784 \overset{VarDropout}{\rightarrow} 784 \overset{Dense}{\rightarrow} 1024 \overset{ReLU}{\rightarrow} 1024 \overset{VarDropout}{\rightarrow} 1024 \overset{Dense}{\rightarrow} 10 \overset{Loss}{\rightarrow} loss$

Our task was to study behavior of retain rates for input neurons during the optimization procedure. We initialized retain rates as 0.8 and 0.5 respectively. For optimization purposes we set the learning rate as 0.01 and decayed it by half after every 200 epochs. Also, we set regularization weight coefficient as $1e^{-4}$. You may see our results on the 2.

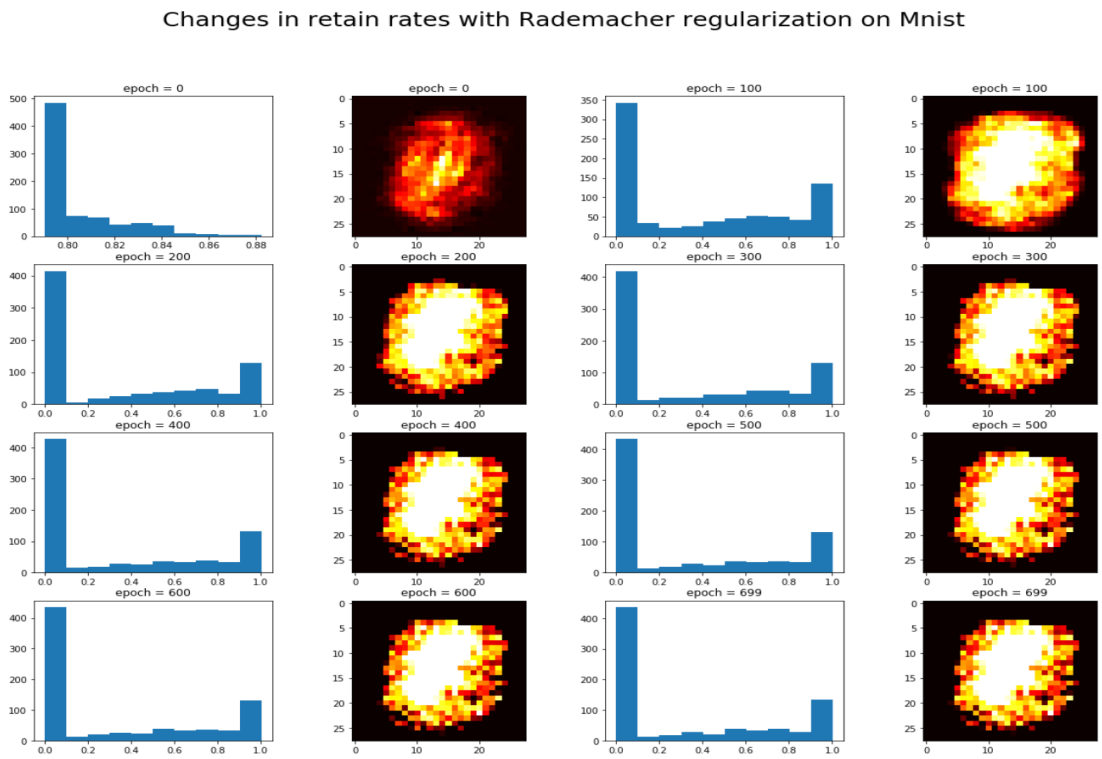

Figure 2: Histograms and retain rates for input neurons during optimization.

As we can see we have a good rate of convergence. Now plot histogram and heatmap for retain probabilities. We can observe that the retain rates converged towards a bimodal distribution. In our case we have one global mode at 0 and one local mode at 1. Empirically, when debugging and testing we observed that for some architectures of NN and initial points and optimization strategies, proportions of modes can be different. Heatmap shows that center pizels often have larger retain rates. This demonstrates that this method is avle to dynamically determine if an input signal is informational or not.

6.3 Experiment 3.

In this experiment we examine the properties of Rademacher regularization on CIFAR-10 data set. The core idea of experiments if the following: we should learn neural network with given architecture and check that retain rates will not nullify. Similar to MNIST dataset, we observe the retain rates for all layers are diffused throughout training process, and finally converged towards a unimodal distribution. However, unlike MNIST dataset, we do not see similar pattern for retain rates of input layer. This is mainly due to the nature of dataset, such that CIFAR-10 images spread over the entire range, hence all pixels are potentially informational to the classification process. This demonstrates that the Rademacher regularizer is able to distinguish the informational pixels and retain them during training. The result of the experiment can be found in Fig. 3

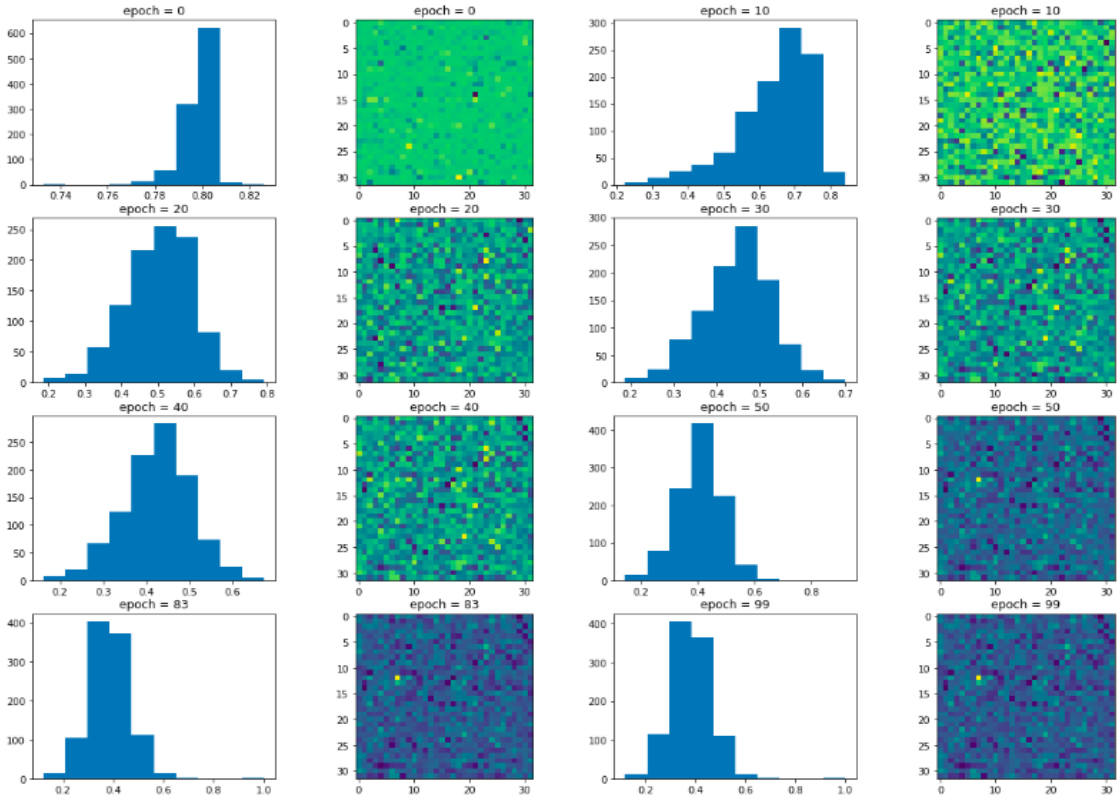

Figure 3: Retain rates diffusion through optimization

7 Overcame Problems

During the process of implementation of Rademacher Variational Dropout we faced different problems: starting from misunderstanding of article and ending with problems with PyTorch, the package used to implement this algorithm. The first problem was with understanding of different learning modes: authors in their article told about stochastic and deterministic mode, however exact proofs and hints were given only for deterministic model. For the stochastic mode it was not stated how we should update parameters of Bernoulli distribution, however this is the core of the article.

8 P.S.

All implemented results can be found on our GitHub

References

[1] Dropout: A simple way to prevent neural networks from overfitting, author=Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, booktitle=The Journal of Machine Learning Research, volume=15(1), year=2014,.

[2] Robert Tibshirani Trevor Hastie and Jerome Friedman. The elements of statistical learning: data mining, inference and prediction. In *Springer*, volume 2 edition, 2009.

[3] Arsenii Ashukha Dmitry Molchanov and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. *Proceedings of the 34th International Conference on Machine Learning*, 2017.