ADAPTIVE DROPOUT WITH RADEMACHER COMPLEXITY REGULARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We 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, we show the network Rademacher complexity is bounded by a function related to the dropout rate vectors and the weight coefficient matrices. Subsequently, we impose this bound as a regularizer and provide a theoretical justified way to trade-off between model complexity and representation power. Therefore, the dropout rates and the empirical loss are unified into the same objective function, which is then optimized using the block coordinate descent algorithm. We discover that the adaptively adjusted dropout rates converge to some interesting distributions that reveal meaningful patterns. Experiments on the task of image and document classification also show our method achieves better performance compared to the state-of-the-art dropout algorithms.

1 INTRODUCTION

Dropout training (Srivastava et al., 2014) has been proposed to regularize deep neural networks for classification tasks. It has been shown to work well in reducing co-adaptation of neurons—and hence, preventing model overfitting. The idea of dropout is to stochastically set a neuron’s output to zero according to Bernoulli random variables. It has been a crucial component in the winning solution to visual object recognition on ImageNet (Krizhevsky et al., 2012). Ever since, there have been many follow-ups on novel learning algorithms (Goodfellow et al., 2013; Baldi & Sadowski, 2013), regularization techniques (Wager et al., 2013), and fast approximations (Wang & Manning, 2013).

However, the classical dropout model has two limitations. First, the model requires to specify the retain rates, i.e., the probabilities of keeping a neuron’s output, a priori to model training. It is often not clear how to choose the retain rates in an optimal way. They are usually set via grid-search over hyper-parameter space or simply according to some rule-of-thumb, and kept consistent throughout the training process thereafter. Another limitation is that all neurons in the same layer share the same retain rate. This exponentially reduces the search space of hyper-parameter optimization. For example, Srivastava et al. (2014) use a fixed retain probability throughout training for all dropout variables in each layer.

In this paper, we propose a novel regularizer based on the Rademacher complexity of a neural network (Shalev-Shwartz & Ben-David, 2014). Without loss of generality, we use multilayer perceptron with dropout as our example and prove its Rademacher complexity is bounded by a term related to the dropout probabilities. This enables us to explicitly incorporate the model complexity term as a regularizer into the objective function.

This Rademacher complexity bound regularizer provides us 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, we are able to incorporate them explicitly into the computation graph of the optimization procedure. We 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). Third, our
proposed regularizer assumes a neuron-wise dropout manner and models different neurons to have different retain rates during the optimization. Our empirical results demonstrate interesting trend on the changes in histograms of dropout probabilities for both hidden and input layers. We also discover that the distribution over retain rates upon model convergence reveals meaningful pattern on the input features.

To the best of our knowledge, this is the first ever effort of using the Rademacher complexity bound to adaptively adjust the dropout probabilities for the neural networks. We organize the rest of the paper as following. Section 2 reviews some past approaches well aligned with our motivation, and highlight some major difference to our proposed approach. We subsequently detail our proposed approach in Section 3. In Section 4, we present our thorough empirical evaluations on the task of image and document classification on several benchmark datasets. Finally, Section 5 concludes this paper and summarizes some possible future research ideas.

2 Related Works

There are several prior works well aligned with our motivation and addressing similar problems, but significantly different from our method. For example, the standout network (Ba & Frey, 2013) extends dropout network into a complex network structure, by interleaving a binary belief network with a regular deep neural network. The binary belief network controls the dropout rate for each neuron, backward propagates classification error and adaptively adjust according to training data. Zhuo et al. (2015) realize the dropout training via the concept of Bayesian feature noising, which incorporates dimension-specific or group-specific noise to adaptively learn the dropout rates.

In addition to these approaches, one other family of solution is via the concept of regularizer. Wang & Manning (2013) propose fast approximation methods to marginalize the dropout layer and show that the classical dropout can be approximated by a Gaussian distribution. Later, Wager et al. (2013) show that the dropout training on generalized linear models can be viewed as a form of adaptive regularization technique. Gal & Ghahramani (2016) develop a new theoretical framework casting dropout training as approximation to Bayesian inference in deep Gaussian processes. It also provides a theoretical justification and formulates dropout into a special case of Bayesian regularization. In the mean time, Maeda (2014) discuss a Bayesian perspective on dropout focusing on the binary variant, and also demonstrate encourage experimental results. Generalized dropout (Srinivas & Babu, 2016) further unifies the dropout model into a rich family of regularizers and propose a Bayesian approach to update dropout rates.

One popular method along with these works is the variational dropout method (Kingma et al., 2015), which provides an elegant interpretation of Gaussian dropout as a special case of Bayesian regularization. It also proposes a Bayesian inference method using a local reparameterization technique and translates uncertainty of global parameters into local noise. Hence, it allows inference on the parameterized Bayesian posteriors for dropout rates. This allows us to adaptively tune individual dropout rates on layer, neuron or even weight level in a Bayesian manner. Recently, Molchanov et al. (2017) extend the variational dropout method with a tighter approximation which subsequently produce more sparse dropout rates. However, these models are fundamentally different than our proposed approach. They directly operates on the Gaussian approximation of dropout models rather than the canonical multiplicative dropout model, whereas our proposed method directly bounds the model complexity of classical dropout model.

Meanwhile, the model complexity and the generalization capability of deep neural networks have been well studied in theoretical perspective. Wan et al. (2013) prove the generalization bound for the DropConnect neural networks—a weight-wise variant of dropout model. Later, Gao & Zhou (2016) extend the work and derive a Rademacher complexity bound for deep neural networks with dropout. These works provide a theoretical guarantee and mathematical justification on the effectiveness of dropout method in general. However, they both assume that all input and hidden layers have the same dropout rates. Thus their bound can not be applied to our algorithm.
3 Regularization with Rademacher Complexity

We would like to focus on the classification problem and use multilayer perceptron as our example. However, note that the similar idea could be easily extended to general feedforward networks. Let us assume a labeled dataset $S = \{(x_i, y_i) | i \in \{1, 2, \ldots, n\}, x_i \in \mathbb{R}^d, y_i \in \{0, 1\}^k\}$, where $x_i$ is the feature of the $i$th sample, $y_i$ is the one-hot class label for the $i$th sample, and $k$ is the number of classes in prediction. Without loss of generality, an $L$-layer multilayer perceptron with dropout can be modeled as a series of recursive function compositions. Let $k^l$ be the number of neurons of the $l$th layer. In particular, the first layer takes sample features as input, i.e., $k^0 = d$, and the last layer outputs the prediction, i.e., $k^L = k$.

We denote $W^l \in \mathbb{R}^{k^{l-1} \times k^l}$ as the linear coefficient matrix from the $(l-1)$th layer to the $l$th layer, and $W^l_{(r)}$ be the $r$th column of $W^l$. For dropout, we denote $\theta^l \in [0, 1]^{k^l}$ as the vector of retain rates for the $l$th layer. We also define $r^l \in \{0, 1\}^{k^l}$ as a binary vector formed by concatenating $k^l$ independent Bernoulli dropout random variables, i.e., $r^l_j \sim \text{Bernoulli}(\theta^l_j)$. To simplify our notation, we further refer $W^l = \{W^1, \ldots, W^L\}$, $r^l = \{r^0, \ldots, r^L\}$, $\theta^l = \{\theta^0, \ldots, \theta^L\}$, $W = W^L$, $r = r^{(L-1)}$, and $\theta = \theta^{(L-1)}$.

For an input sample feature vector $x \in \mathbb{R}^d$, the function before the activation of the $j$th neuron in the $l$th layer $f^l_j$ is

$$f^l_j(x; W^l, r^l) = \sum_r W^l_{(r)} f^{l-1}(x; W^{l-1}, r^{l-1}), \forall l \in \{2, 3, \ldots, L\}$$

where $\phi: \mathbb{R} \rightarrow \mathbb{R}^+$ is the rectified linear activation function ([Nair & Hinton, 2010] ReLU). In vector form, if we denote $\odot$ as the Hadamard product, we could write the output of the $l$th layer as

$$f^l(x; W, r) = (r^{l-1} \odot \phi(f^{l-1}(x; W^{l-1}, r^{l-1}))) W^l.$$  

Without loss of generality, we also apply Bernoulli dropout to the input layer parameter $\theta^0 \in \mathbb{R}^d$, i.e., $f^1(x; W, r^0) = (r^0 \odot x) W^1$. Note that the output of the neural network $f^L(x; W, r) \in \mathbb{R}^k$ is a random vector due to the Bernoulli random variables $r$. We use the expected value of $f^L(x; W, r)$ as the deterministic output

$$f^L(x; W, \theta) = \mathbb{E}_r[f^L(x; W, r)].$$

The final predictions are made through a softmax function, and we use the cross-entropy loss as our optimization objective. To simplify our analysis, we follow [Wan et al., 2013] and reformulate the cross-entropy loss on top of the softmax into a single logistic function

$$\text{loss}(f^L(x; W, \theta), y) = - \sum_{j} y_j \log \frac{e^{f^L_j(x; W, \theta)}}{\sum_{j'} e^{f^L_{j'}(x; W, \theta)}}.$$  

3.1 Rademacher Complexity of Dropout Neural Network

Define loss $\circ f^L$ as the composition of the logistic loss function loss and the neural function $f^L$ returned from the $L$th (last) layer, i.e.,

$$\text{loss} \circ f^L = \{(x, y) \rightarrow \text{loss}(f^L(x; W, \theta), y)\}.$$  

Theorem 3.1. Let $X \in \mathbb{R}^{n \times d}$ be the sample matrix with the $i$th row $x_i \in \mathbb{R}^d$, $\frac{p}{p} + \frac{q}{q} = 1$. If the $p$-norm of every column of $W^l$ is bounded by a constant $B^l$, denote $W = \{W | \max_j \|W^l_j\|_p \leq B^l, \forall l \in \{1, 2, \ldots, L\}\}$, given $\theta$, the empirical Rademacher complexity of the loss for the dropout neural network defined above is bounded by

$$R_\mathbb{S}(\text{loss} \circ f^L) = \frac{1}{n} \mathbb{E}_{\sigma} \left[ \sup_{W \in \mathbb{W}} \sum_{i=1}^{n} \sigma_i \text{loss}(f^L(x_i; W, \theta), y_i) \right]$$

$$\leq k \sqrt{\frac{2 \log(2d)}{n}} \|X\|_{\max} (\prod_{l=1}^{L} B^l \|\theta^{l-1}\|_q),$$

where $k$ is the number of classes to predict, $\theta^l$ is the $k^l$-dimensional vector of Bernoulli parameters for the dropout random variables in the $l$th layer, $\sigma_i$ are i.i.d. Rademacher random variables, and $\|\cdot\|_{\max}$ is the matrix max norm defined as $\|A\|_{\max} = \max_{ij} |A_{ij}|$.

Please refer to the appendix for the proof.
3.2 Regularize Dropout Neural Network with Rademacher Complexity

We have shown that the Rademacher complexity of a neural network is bounded by a function of the dropout rates, i.e., Bernoulli parameters $\theta$. This makes it possible to unify the dropout rates and the network coefficients $W$ in one objective. By imposing our upper bound of Rademacher complexity to the loss function as a regularizer, we have

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

(2)

where the variable $\lambda \in \mathbb{R}^+$ is a weighting coefficient to trade off the training loss and the generalization capability. The empirical loss $\text{Loss}(S, f^L(\cdot; W, \theta))$ and regularizer function $\text{Reg}(S, W, \theta)$ are defined as

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

$$\text{Reg}(S, W, \theta) = k \sqrt{\frac{\log d}{n}} \| X \|_{\max} \left( \prod_{l=1}^L \| \theta^{l-1} \|_q \max_j \| W^l_j \|_p \right),$$

where $W^l_j$ is the $j$th column of $W^l$ and $\theta^l$ is the retain rate vector for the $l$th layer. The variable $k$ is the number of classes to predict and $X \in \mathbb{R}^{n \times d}$ is the sample matrix.

In addition to the Rademacher regularizer $\text{Reg}(S, W, \theta)$, the empirical loss term $\text{Loss}(S, f^L(\cdot; W, \theta))$ also depends on the dropout Bernoulli parameters $\theta$. Intuitively, when $\theta$ becomes smaller, the loss term $\text{Loss}(S, f^L(\cdot; W, \theta))$ becomes larger, since the model is less capable to fit the training samples (i.e., less representational power), the empirical Rademacher complexity bound becomes smaller (i.e., more generalizable), and vice versa. Figure 1 plots the values of the cross-entropy loss function and empirical Rademacher $p = q = 2$ regularizer upon model convergence under different fixed settings of retain rates. In the extreme case, when all $\theta^l_i$ become zeros, the model always makes random guess for prediction, leading to a large fitness error $\text{Loss}(S, f^L(\cdot; W, \theta))$, and the Rademacher complexity $R_S(\text{loss} \circ f^L)$ approaches $0$.

3.3 Optimize Dropout Rates

We now incorporate the Bernoulli parameters $\theta$ into the optimization objective as in Eqn. (2), i.e., the objective is a function of both weight coefficient matrices $W$ and retain rate vectors $\theta$. In particular, the model parameters and the dropout out rates are optimized using a block coordinate descent algorithm. We start with an initial setting of $W$ and $\theta$, and optimize $W$ and $\theta$ in an alternating fashion. During the optimization of $\theta$, we use the expected value of the dropout layer to rescale the output from each layer. It significantly speeds up the forward propagation process, as we do not need

\footnote{To balance out the regularizer and the loss function so that they scale similarly as the sample size and internal layer nodes grow, we add some scaling factors to $\lambda$, which is discussed in Section 4.}
to iteratively sample the dropout variables. Note that this is an approximation to the true \( f^k(x; W, \theta) \), however, in practice, we find the difference are negligible on each minibatch. Essentially, it makes the layer output deterministic and the underlying network operates as if without dropout, i.e., similar to the approximation used in Srivastava et al. [2014] during testing time.

### 4 Experiments

We apply our proposed approach with different network architectures, on the task of image and text classification using several public available benchmark datasets. All hidden neurons and convolutional filters are rectified linear units (Nair & Hinton [2010], ReLU). We found that our approach achieves superior performance against strong baselines on all datasets. For all datasets, we hold out 20% of the training data as validation set for parameter tuning and model selection. After then, we combine both of these two sets to train the model and report the classification error rate on test set. We optimize categorical cross-entropy loss on predicted class labels with Nesterov momentum of 0.95 (Sutskever et al. [2013]).

For Rademacher complexity term, we perform a grid search on the regularization weight \( \lambda \in \{0.5, 0.1, 0.05, 0.01, 0.005, 0.001\} \), and update the dropout rates after every \( f \in \{1, 5, 10, 50, 100\} \) minibatches. For variational dropout method (Kingma et al. [2015], \textsc{VarDrop}), we examine the both Type-A and Type-B variational dropout with per-layer, per-neuron or per-weight adaptive dropout rate. We found the neuron-wise adaptive regularization on Type-A variational dropout layer often reports the best performance under most cases. We also perform a grid search on the regularization noise parameter in \( \{0.1, 0.01, 0.001, 1e^{-4}, 1e^{-5}, 1e^{-6}\} \). For sparse variational dropout method (Molchanov et al. [2017], \textsc{SparseVarDrop}), we find the model is much more sensitive to regularization weights, and often gets diverged. We examine different regularization weight in \( \{1e^{-3}, 1e^{-4}, 1e^{-5}\} \). We follow similar weight adjustment scheme and scale it up by 10 after first \( \{100, 200, 300\} \) epochs, then further scale up by 5 and 2 after same number of epoch.

#### Scales of Regularization

In practice, we want to stabilize regularization term within some manageable variance, so its value does not vary significantly upon difference structure of the underlying neural networks. Hence, we design some heuristics to scale the regularizer to offset the multiplier effects raised from network structure. For instance, recall the neural network defined in Section 3 the Rademacher complexity regularizer with \( p = q = 2 \) after scaling is

\[
    k \sqrt{\frac{\log d}{n}} \max_i \|X_i\|_{\infty} \left( \prod_{l=1}^{L} \frac{\max_j \|W^l_j\|_2 \|\theta^l_j\|_2}{k^l} \sqrt{\frac{k^{l} - 1 + k^l}{\log k^l}} \right),
\]

where \( W^l_j \) is the \( j \)th column of the weight coefficient matrix \( W^l \) and \( \theta^l \) is the retain rate vector for the \( l \)th layer. The variable \( k \) is the number of classes to predict and \( X \in \mathbb{R}^{n \times d} \) is the sample matrix. Similarly, we could rescale the Rademacher complexity regularizers under other settings of \( p = 1, q = \infty \) and \( p = \infty, q = 1 \). Please refer to the appendix for the scaled Rademacher complexity bound regularizers and detailed derivations.

#### 4.1 MNIST

The MNIST dataset is a collection of 28 × 28 pixel hand-written digit images in grayscale, containing 60K for training and 10K for testing. The task is to classify the images into 10 digit classes from 0 to 9. All images are flattened into 784 dimension vectors, and all pixel values are rescaled to gray scale. We examine several different network structures, including architectures of 1 hidden layer with 1024 units, 2 hidden layers with 800 neurons each, as well as 3 hidden layers with 1024 units each.

<table>
<thead>
<tr>
<th>Model</th>
<th>1024</th>
<th>800 × 2</th>
<th>1024 × 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multilayer Perceptron</td>
<td>1.69</td>
<td>1.62</td>
<td>1.61</td>
</tr>
<tr>
<td>+ Dropout</td>
<td>1.22</td>
<td>1.28</td>
<td>1.25</td>
</tr>
<tr>
<td>+ \textsc{VarDrop}</td>
<td>1.20</td>
<td>1.16</td>
<td>1.07</td>
</tr>
<tr>
<td>+ \textsc{SparseVarDrop}</td>
<td>1.34</td>
<td>1.30</td>
<td>1.27</td>
</tr>
<tr>
<td>+ Rademacher ( p = q = 2 )</td>
<td>1.14</td>
<td>1.05</td>
<td>0.95</td>
</tr>
<tr>
<td>+ Rademacher ( p = 1, q = \infty )</td>
<td>1.13</td>
<td>1.04</td>
<td>0.96</td>
</tr>
<tr>
<td>+ Rademacher ( p = \infty, q = 1 )</td>
<td>1.11</td>
<td>1.08</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 1: Classification error on MNIST dataset.
We use a learning whitening as suggested in Srivastava et al. (2014). No data augmentation is used. The neural network has a \( 5 \times 96 \) architecture we evaluate on uses three convolutional layers, each of which followed by a max-pooling layer. We preprocess all images by subtracting the per-pixel mean computed over all training set, then with ZCA whitening as suggested in Srivastava et al. (2014). No data augmentation is used. The neural network architecture we evaluate on uses three convolutional layers, each of which followed by a max-pooling layer. The convolutional layers have 96, 128, and 256 filters respectively. Each convolutional layer has a \( 5 \times 5 \) receptive field applied with a stride of 1 pixel, and each max-pooling layer pools from \( 3 \times 3 \) to \( 2 \times 2 \) receptive field.

Table 1 compares the performance of our proposed models against other techniques. We use a learning rate of 0.01 and decay it by 0.5 after every \{300, 400, 500\} epochs. We let all models run sufficiently long with 100K updates. For all models, we also explore different initialization for neuron retaining rates, including \{0.8, 1.0\} for input layers, \{0.5, 0.8, 1.0\} for hidden layers. In practice, we find initializing the retaining rates to 0.8 for input layer and 0.5 for hidden layer yields better performance for all models, except for SPARSEVARDROP model, initializing retaining rate to 1.0 for input layer seems to give better result.

Figure 2 illustrates the changes in retain rates for both input and hidden layers under Rademacher regularization \( p = q = 2 \) with 0.01 regularization weight. The network contains one hidden layer of 1024 ReLU units. The retain rates were initialized to 0.8 for input layer and 0.5 for hidden layer. The learning rate is 0.01 and decayed by 0.5 after every 300 epochs. We observe the retain rates for all layers are diffused throughout training process, and finally converged towards a unimodal distribution. We also notice that the retain rates for input layer upon model convergence demonstrate interesting feature pattern of the dataset. For example, the pixels in surrounding margins yield smaller retain rates, and the center pixels often have larger retain rates. This is because the digits in MNIST dataset are often centered in the image, hence all the surrounding pixels are not predictive at all when classifying an instance. This demonstrates that our proposed method is able to dynamically determine if an input signal is informational or not, and subsequently gives higher retain rate if it is, otherwise reduce the retain rate over time.

\[ \text{Figure 2: Changes in retain rates with Rademacher } p = q = 2 \text{ regularization on MNIST dataset. Top-Left: changes in retain rate histograms for input layer (784 gray scale pixels) through training. The retain rates get diffused over time, and only a handful of pixels have retain rates close to 1. Top-Right: changes in retain rate histograms for hidden layer (1024 ReLU units) through training process. Bottom-Left: sample images from MNIST dataset. Bottom-Right: retain rates for corresponding input pixels upon model convergence. The surrounding pixels of input image yield smaller retain rates (corresponds to the dark background area), and the center ones have significantly larger retain rates (corresponds to the number pixels).} \]

4.2 CIFAR

CIFAR10 and CIFAR100 datasets are collections of 50K training and 10K testing RGB images from 10 and 100 different image categories. Every instance consists of \( 32 \times 32 \) RGB pixels. We preprocess all images by subtracting the per-pixel mean computed over all training set, then with ZCA whitening as suggested in Srivastava et al. (2014). No data augmentation is used. The neural network architecture we evaluate on uses three convolutional layers, each of which followed by a max-pooling layer. The convolutional layers have 96, 128, and 256 filters respectively. Each convolutional layer has a \( 5 \times 5 \) receptive field applied with a stride of 1 pixel, and each max-pooling layer pools from \( 3 \times 3 \) to \( 2 \times 2 \) receptive field.
pixel region with strides of 2 pixels. These convolutional layers are followed by two fully-connected layers having 2048 hidden units each.

Table 2 summarizes the performance of our proposed models against other baselines. We initialize dropout rates settings with \( \{0.9, 1.0\} \) for input layers, \( \{0.75, 1.0\} \) for convolutional layers and \( \{0.5, 1.0\} \) for fully-connected layers. Similarly to the MNIST evaluation, we find setting the corresponding retaining probabilities for input layers, convolutional layers and fully-connected layers to \( 0.9, 0.75 \) and \( 0.5 \) respectively yields best performance under all models. We initialize the learning rate to 0.001 and decay it exponentially every \{200, 300, 400\} epochs.

Figure 3 illustrates the changes in retain rates for both input and hidden layers under Rademacher regularization \( (p = q = 2) \) with 0.01 regularization weight. The network contains two convolution layers with 32 and 64 convolutional filters followed by one fully-connected layer with 1024 neurons. All hidden units use ReLU activation functions. The retain rates were initialized to 0.9 for input layer, 0.75 for convolutional layer and 0.5 for fully-connected layer. The learning rate is 0.001 and exponentially decayed by half after every 300 epochs. 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 CIFAR10 images spread over the entire range, hence all pixels are potentially informational to the classification process. This again demonstrates that the Rademacher regularizer is able to distinguish the informational pixels and retain them during training.

### 4.3 Text Classification

In addition, we also compare our proposed approach on text classification datasets—SUBJ and IMDB. SUBJ is a dataset containing \( 10K \) subjective and objective sentences (Pang & Lee 2004) with nearly \( 14.5K \) vocabulary after stemming. All subjective comments come from movie reviews expressing writer’s opinion, whereas objective sentences are from movie plots expressing purely facts. We randomly sample 20\% from the collections as test data, and use other 80\% for training and validation. IMDB is a collection of movie reviews from IMDB website, with 25\% for training and another 25\% for test (Maas et al. 2011), containing more than 50\% vocabulary after stemming. It contains an even number of positive (i.e., with a review score of 7 or more out of a scale of 10) and negative (i.e., with a review score of 4 or less out of 10) reviews. The dataset has a good movie diversity coverage with less than 30 reviews per movie. For each sentence or document in these datasets, we normalize it into a vector of probability distribution over all vocabulary.

Table 3 summarizes the performance of our proposed models against other baselines. We initialize dropout rates settings with \( \{0.8, 1.0\} \) for input layers and \( \{0.5, 1.0\} \) for fully-connected layers. Similarly, by setting the corresponding retaining probabilities for input layers and fully-connected layers to 0.8 and 0.5 respectively, the model often yields the best performance. We use a constant learning rate of 0.001, as well as an initialization learning rate of 0.01 and decay it by half every \{200, 300, 400\} epochs. We notice that overall the improvement of dropout is not as significant as MNIST or CIFAR datasets.

Figure 4 illustrates the changes in retain rates for both input and hidden layers under Rademacher regularization \( (p = q = 2) \) with 0.005 regularization weight on IMDB dataset. The network contains one hidden layer of 1024 ReLU units. The retain rates were initialized to 0.8 for input layer and 0.5 for hidden layer. The learning rate is 0.001 and decayed by 0.5 after every 200 epochs. Similar to
MNIST datasets, we observe the retain rates for all layers are diffused slightly, and the retain rates for input layer upon model convergence demonstrate interesting feature pattern of the dataset.

Recall that the task for IMDb dataset is to classify movie reviews into negative or positive labels. Generically speaking, adjectives are more expressive than nouns or verbs in this scenario, and our findings seems to be consistent with this intuition. From our model, words like “lousi(y)”, “flawless”, “obnoxious”, “finest” and “unwatchable” yield large retain rates and hence indicative feature to predicaiton. We also notice that “baldwin” and “kurosawa” are also very informative features. On the other hand, words like “young”, “review”, “role”, “animation” and “year” have near zero retain rates upon model convergence, which are less informative. One other interesting observation is that the word “oscar” also yields near zero retain rate, which implies the positivity or negativity of a movie review is not necessarily correlated with the mention of Academy Awards.

5 Conclusion

Imposing regularizaiton for a better model generalization is not a new topic. However we tackle the problem for the dropout neural network regularization in a different way. The theoretical upper bound we proved on the Rademacher complexity facilitates us to directly incorporate the dropout rates into the objective function. In this way the dropout rate can be optimized by block coordinate
In the future, we would like to investigate the sparsity property of the learnt retain rates to encourage a sparse representation of the data and the neural network structure (Wen et al., 2016), similar to the sparse Bayesian models and relevance vector machine (Tipping, 2001). We would also like to explore the applications of deep network compression (Han et al., 2015a; Iandola et al., 2016; Ullrich et al., 2017; Molchanov et al., 2017; Louizos et al., 2017). In addition, one other possible research direction is to dynamically adjust the architecture of the deep neural networks (Srinivas & Babu, 2015; Han et al., 2015b; Guo et al., 2016), and hence reduce the model complexity via dropout rates.

ACKNOWLEDGMENTS

Available after blind review.

REFERENCES


6 APPENDIX

6.1 PROOF OF THEOREM 3.1

Proof. In the analysis of Rademacher complexity, we treat the functions fed into the neurons of the $l$th layer as one function class $F_l = \mathbf{f}_l(x; w^l)$. Here again we are using the notation $w^l = \{w_1, \ldots, w^l\}$, and $w = w^L$. As a consequence $\forall j, f_j(x; w^l) \in F_l$.

To simplify our analysis, we follow [Wan et al., 2013] and reformulate the cross-entropy loss on top of the softmax into a single logistic function

$$\text{loss}(f^L(x; W), y) = -\sum_j y_j \log \frac{e^{f_j^L(x; w)}}{\sum_j e^{f_j^L(x; w)}}.$$ 

The function class fed into the neurons of the $l$th layer $f^l(x; w^l)$ admits a recursive expression

$$f^l(x; w^l, r^{(l-1)}) = \sum_k \phi(f_{k}^{l-1}(x; w^{l-1}, r^{(l-2)}))r_{k}^{l-1}w^l_k \quad (3)$$

$$f^l(x; w^l, \theta^{(l-1)}) = E_{r^{(l-1)}} f^l(x; w^l, r^{(l-1)}) \quad (4)$$

Given the neural network function (1) and the logistic loss function $l$ is 1 Lipschitz, by Contraction lemma (Shalev-Shwartz & Ben-David, 2014), the empirical Rademacher complexity of the loss function is bounded by

$$R_\phi(l \circ f^L) = \frac{1}{n} E_{\sigma \in \Sigma} \max_{w} \sum_{i} \sigma_i l(f^L(x_i; w), y_i) \leq \frac{k}{n} E_{\sigma \in \Sigma} \max_{w} \sum_{i} \sigma_i f^L(x_i; w) = kR_\phi(f^L) \quad (5)$$

Note the empirical Rademacher complexity of the function class of the $L$th layer, i.e., the last output layer, is

$$R_\phi(f^L) = \frac{1}{n} E_{\sigma_1} \left[ \sup_{w} \sum_{i} \sigma_i f^L(x_i; w) \right]$$ 

11
To prove the bound in a recursive way, let’s also define a variant of the Rademacher complexity with absolute value inside the supremum:

\[
\tilde{R}_0(f) = \frac{1}{n} E_{(\varepsilon_i)} \sup_{w} \left| \sum_i \sigma_i f(x_i; w) \right|
\]  

(7)

Note here \(\tilde{R}_0(f)\) is not exactly the same as the Rademacher complexity defined before in this paper. And we have

\[
R_0(f^L) \leq \tilde{R}_0(f^L)
\]

(8)

Now we start the recursive proof. The empirical Rademacher complexity (with absolute value inside supremum) of the function class of the \(l^{th}\) layer is

\[
\tilde{R}_0(f^l) = \frac{1}{n} E_{(\varepsilon_i)} \sup_{w} \left| \sum_i \sigma_i f^l(x_i; w) \right|
\]  

(9)

Let \(g_{j}^{l-1}(x; w^{l-1}, \theta^{(l-2)}) = E_{\varepsilon_i} f_{j}^{l-1}(x; w^{l-1}, \theta^{(l-2)})\), then

\[
\tilde{R}_0(f^l) = \frac{1}{n} E_{(\varepsilon_i)} \sup_{w: l-1} \left| \sum_i \sigma_i \sum_j w_j^l \theta_j^{l-1} g_{j}^{l-1}(x_i; w^{l-1}, \theta^{(l-2)}) \right|
\]  

(10)

According to the assumption, \(p \geq 1, q = p/(p - 1)\), and \(\|w^l\|_p \leq B_l\), from equation (10), by Holder’s inequality, we have \(\forall l \in \{1, \ldots, L\}\)

\[
\tilde{R}_0(g^l(x; w^l)) \leq \tilde{R}_0(f^l(x; w^l))
\]

(12)

For the first layer, i.e., the feature layer with dropout but without activation function, if \(\|w^1\|_p \leq B^1\), then

\[
\tilde{R}_0(f^1(x; w^1)) = \frac{1}{n} E_{(\varepsilon_i)} \left[ \sup_{\|w^1\|_p \leq B^1} \sum_{i=1}^n \sigma_i E_{\varepsilon_i} w_i < w^1, x_i \otimes r^0 > \right]
\]

(13)
By Lemma 26.11 in Shalev-Shwartz & Ben-David (2014), we know the last term in (13) is bounded by
\[ \frac{1}{n} \mathop{E}_{(\sigma_i)} \left[ \left\| \sum_{i=1}^{n} \sigma_i x_i \right\| \right] \leq \max_i \|x_i\|_\infty \sqrt{2 \log(2d)/n} \]
Thus we get
\[ \tilde{R}_0(f^1(x; w^1)) \leq \max_i \|x_i\|_\infty B^1 \|\theta^1\| q \sqrt{2 \log(2d)/n} \]
(14)
Combining the inequalities (5), (8), (11), (12), and (14), we have
\[ R_S(l \circ f^L) \leq k \sqrt{2 \log(2d)/n} \max_i \|x_i\|_\infty \left( \prod_{l=1}^{L} B^l \|\theta^{l-1}\| q \right) \]
(15)
6.2 Generalization Bound on the Dropout Neural Networks
Here we need to define truncated cross entropy loss function:
\[ \tilde{l} \circ f^L = \tilde{l}(f^L(x, \theta), y) = \min(\text{loss}(f^L(x; W, \theta), y), C_l) \]
where \( C_l \) is a constant. Note with the truncation, the cross entropy loss is still 1-Lipschitz so the empirical Rademacher complexity bound still holds for the truncated loss \( \tilde{l}(f^L(x; W, \theta), y) \).

**Theorem 6.1.** For the dropout neural network defined in section 3, if truncated cross entropy loss \( \tilde{l} \) is used, then \( \forall \delta \geq 0 \), with probability at least \( 1 - 2\delta \), \( \forall l \circ f^L : \)
\[ E_{(x,y) \sim D}(\tilde{l}(f^L(x; W, \theta), y)) \leq \frac{1}{n} \sum_i \tilde{l}(f^L(x_i; W, \theta), y_i) + R_0(\tilde{l} \circ f^L) + 3C_l \sqrt{1/(2n\delta)} \]
(17)
Note here the empirical Rademacher complexity for the bounded loss function \( R_0(\tilde{l} \circ f^L) \) admits the same bound as the empirical Rademacher complexity for the unbounded cross entropy loss \( R_0(l \circ f^L) \).

6.3 Towards an Unified View over Rademacher Regularization

In fact, adding a Rademacher related regularizer, though not investigated much, is not new at least for linear functions.

It is well known (Shalev-Shwartz & Ben-David 2014) that the empirical Rademacher complexity of the linear class
\[ H_2 = \{ x \to \langle w, x \rangle : \|w\|_2 \leq B_2 \} \]
is bounded by
\[ R_0 \leq \max_i \|x_i\|_2 B_2 / \sqrt{n} \]
Note the \( l_2 \) loss function is 2-Lipschitz. In this way, we may interpret the regularizer in the ridge regression related an upper bound for the empirical Rademacher complexity of the linear function class.

Similarly for the linear class
\[ H_1 = \{ x \to \langle w, x \rangle : \|w\|_1 \leq B_1 \} \]
the empirical Rademacher complexity is bounded by
\[ R_0 \leq \max_i \|x_i\|_\infty B_1 \sqrt{2 \log(2d)/n} \]
So the lasso problem can also be viewed as adding a Rademacher-related regularization to the empirical loss minimization objective.

\*Note that the content in Section 6.3 and 6.4 is based purely on heuristics, and derived on an ad hoc basis.
6.4 Scales of Regularization

In application we do not want the regularization term to vary too much when the neural network has different number of internal neurons. To overcome that we design some heuristics to add to the regularizer. Note here all the scales mentioned in this section are added in a heuristic fashion. It is purely empirical.

When \( p = q = 2 \), the regularizer is bounded by

\[
R_S(l \circ f^L) \leq k \sqrt{\frac{\log(d)}{n}} \max_i \|x_i\|_\infty \left( \Pi_{l=1}^L \max_j \|W^l_j\|_2 \|\theta^{l-1}\|_2 \right).
\]

(18)

where \( W^l_j \in \mathbb{R}^{k^l-1} \) is the \( j \)-th column of \( W^l \). Suppose we use i.i.d. uniform random variable to initialize \( W^l \) such that \( \forall l \ \forall_{l,j} w^l_{ij} \sim U[-\sqrt{6}/(k^l + k^{l+1}), \sqrt{6}/(k^l + k^{l+1})] \). Considering the scales of the maximum among the 2-norms of \( W^l_j \), we use a scaled regularizer instead:

\[
k \sqrt{\frac{\log d}{n}} \max_i \|x_i\|_\infty \left( \Pi_{l=1}^L \max_j \|W^l_j\|_2 \|\theta^{l-1}\|_2 \frac{\|\theta^{l-1}\|_2}{k^l + k^{l+1}} \right).
\]

Similarly, when \( p = \infty \) and \( q = 1 \), the scaled regularizer we used is

\[
k \sqrt{\frac{\log d}{n}} \max_i \|x_i\|_\infty \left( \Pi_{l=1}^L \|W^l\|_{\max} \|\theta^{l-1}\|_1 \frac{\|\theta^{l-1}\|_1}{k^l + k^{l+1}} \right).
\]

When \( p = 1 \) and \( q = \infty \), we used the scaled regularizer

\[
k \sqrt{\frac{\log d}{n}} \max_i \|x_i\|_\infty \left( \Pi_{l=1}^L \frac{\|W^l\|_1 \|\theta^{l-1}\|_\infty}{k^l - 1 \sqrt{\log k^l/(k^l + k^{l+1})}} \right).
\]