Stochastic Function Norm Regularization of DNNs

Amal Rannen Triki*
Dept. of Computational Science and Engineering
Yonsei University
Seoul, South Korea
amal.rannen@yonsei.ac.kr

Matthew B. Blaschko
Center for Processing Speech and Images
Departement Elektrotechniek
KU Leuven, Belgium
matthew.blaschko@esat.kuleuven.be

Abstract

Deep neural networks (DNNs) have had an enormous impact on image analysis. State-of-the-art training methods, based on weight decay and DropOut, result in impressive performance when a very large training set is available. However, they tend to have large problems overfitting to small data sets. Indeed, the available regularization methods deal with the complexity of the network function only indirectly. In this paper, we study the feasibility of directly using L_2 function norms for regularization. Two methods to integrate this new regularization in a stochastic backpropagation framework are proposed. Moreover, the convergence of these new algorithms is studied. We finally show that they outperform the state-of-the-art methods in the low sample regime on benchmark datasets (MNIST and CIFAR10). The obtained results demonstrate very clear improvement, especially in the context of small sample regimes with data laying in a low dimensional manifold. Source code of the method is made available at `https://github.com/AmalRT/DNN_Reg`.

1 Introduction

Deep Neural Networks (DNNs) have been shown to be a powerful tool in learning from large datasets in many domains. However, they require a large number of training samples in order to have reasonable generalization power. Indeed, this fact tends to limit their application to the type of images where it is easy to construct a large labeled database, such as natural images where collection can be done from image sharing websites and labels obtained at low cost via crowdsourcing. When only few labeled samples are available, such as in medical imaging, DNNs tend to overfit to the data and may not be competitive with methods for which known tractable methods for stronger regularization are available, e.g. kernel methods.

Generally in learning theory, one wants to minimize the statistical risk $\mathcal{R}(W) = \int \ell(f_W(x), y) dP(x, y)$, where ℓ is a loss a function and $P(x, y)$ is the data distribution of inputs and outputs, respectively. However, as P is generally unknown, only the empirical risk $\mathcal{R}_N(W) = \frac{1}{N}\sum_{i=1}^{N} \ell(f_W(x_i), y_i)$ is accessible. This approximation is accurate when a large database is used, but when only few samples are available, an appropriate regularization is required. Training a DNN with regularization is solving $\arg\min_W \mathcal{R}_N(W) + \lambda\Omega(f_W)$. Much research has approached the idea of regularized DNN training. However, until now, this regularization tends to be poor compared, for example, to regularization in kernel methods [11]. Two categories of regularization can be distinguished in the existing literature: (i) regularizing by controlling the magnitude of the weights W (ℓ_1 and ℓ_2 regularization [4, 8]); and (ii) regularizing by controlling the network architecture (DropOut [6] and DropConnect [13]). In order to make the regularization stronger, we study the feasibility of using L_2 function norms directly. The canonical L_2 function norm is:

$$\Omega(f_W) = \|f_W\|_2^2 = \int \langle f_W(x), f_W(x) \rangle dx. \tag{1}$$

*ART is also affiliated with KU Leuven.

As the exact value of this norm, and more importantly of its gradient, are not accessible, we study two closely-related weighted norms. For both cases, we reweight the norm using a probability distribution (we suppose for simplicity that all the distributions admit density functions).

2 Methods and algorithms : Function norm regularization

Data distribution: The first idea is to consider the function norm with respect to the data distribution :

$$\Omega(f_W) = \int \langle f_W(x), f_W(x) \rangle dP(x), \tag{2}$$

where P is the marginal data distribution $P(x) := \int P(x, y)dy$.

One can easily see that this is a valid function norm [3], and furthermore is just a weighted canonical function norm such that the regularization has a higher effect on the data that has the higher probability to be observed.

For this first method, our regularization term, written another way is $\mathbb{E}_{x \sim P(x)} [\langle f_W(x), f_W(x) \rangle]$. For some set $\{x_1, \ldots, x_m\}$ drawn i.i.d. from $P(x)$

$$\frac{1}{m} \sum_{i=1}^{m} \langle f_W(x_i), f_W(x_i) \rangle \tag{3}$$

is an unbiased estimate. For samples outside the training set, this is a U-statistic of order 1, and has an asymptotic Gaussian distribution for which finite sample estimates converge quickly [7]. **Only the images of these samples under f_W are needed for regularization and not their labels, which is particularly interesting for applications such as medical imaging where labeling the samples require expert intervention and is usually highly expensive.**

Slice sampling: The second idea is to use a probability density proportional to the integrand for the weighted norm. The regularization term is then:

$$\Omega(f_W) = \int \langle f_W(x), f_W(x) \rangle d\tilde{P}(x), \tag{4}$$

where $\tilde{P}(x) \propto \|f_W(x)\|_2^2$.

This idea is inspired by stochastic integration methods. A simple way to apply these methods is to draw samples in the domain of the function and to approximate the integral by the average value of the function in these points in a Quasi-Monte-Carlo fashion [1].

However, as we are working in a very high dimension, this type of numerical integration is not appropriate because it suffers from the curse of dimensionality and results in a sample concentration independent of the characteristics of the function we want to integrate, i.e. $\|f_W(x)\|_2^2$. As this function is real and non-negative, we can consider sampling from a distribution for which the density is proportional to the function itself, using Markov-chain methods such as Gibbs [2] and Metropolis–Hastings [5] sampling. As Gibbs sampling requires the approximation of the marginal distribution (which is not easily accessible in our case and most importantly non-standard), and Metropolis-Hastings sampling is depending on the choice of the proposal distribution, we generate our samples using slice sampling [9].

This method draws samples uniformly from the hypervolume under the n-dimensional graph of the function by iterating the following steps (to sample from a function f): (i) Choose an initial sample x_0; (ii) Draw a uniformly distributed sample $y \in (0, f(x_0))$; (iii) Find an hyperrectangle, around x_0, that is contained in the slice $S = \{x : f(x) > f(x_0)\}$; (iv) Draw the new samples x_1 uniformly from this hyperrectangle; (v) Go back to the second step and iterate until collecting the desired number of samples. In [9], many methods to define the hyperrectangle in the third step are proposed. Here we used the method called "stepping-out and shrinking-in". This method consists of finding a neighborhood of x_0, increasing it progressively until being outside the slice, and drawing samples from this increased neighborhood. If the drawn sample is outside the slice, it is used to shrink the neighborhood.

The samples that are drawn using this procedure will have a concentration that depends on the variation of $\|f_W(x)\|_2^2$. They will be concentrated around the regions where its values are high. The effect of the curse of dimensionality is then limited. Moreover, this means that the regularization will have a higher effect on the regions where the samples are more concentrated, i.e. the regions

where the values of the function are high, resulting in a less complex function. **This method does not require new images, which is interesting when we have access to only a small number of samples (labeled or unlabeled).**

An extended version of this paper is available on arXiv, in which we prove the convergence of stochastic gradient descent when applied to the regularized objectives, and provide an algorithm to integrate the regularization in a classical training procedure [10].

3 Experiments and results

To test the proposed methods, two series of experiments using two different databases are considered: MNIST and CIFAR10. The experiments are conducted using MatConvNet [12]. For both of the databases, the convolutional neural network LeNet is used. For each we compare our regularization methods to (i) training with only weight decay and (ii) training with DropOut (with different rates) + weight decay.

MNIST: We report the results of the tests conducted with our algorithms on MNIST (60,000 samples for training and 10,000 samples for test). These experiments where conducted using only a 2.4 GHz Intel Core 2 Duo processor.

In a first set of experiments, we test our first idea and the two reference methods listed above using a decreasing number of samples of MNIST in the training data set. The remaining training samples are used for regularization using data distribution. For slice sampling, we generate 6000 samples in the case of 600 training samples, and 4000 samples in the case of 100 samples. Figure 1 shows the evolution of the test top-1-error during training (the x-axis represents the number of epochs) for weight decay, weight decay+DropOut with the rate 50%, weight decay+DropOut with the rate 25% and approximate function norm regularization using the data distribution and slice sampling.

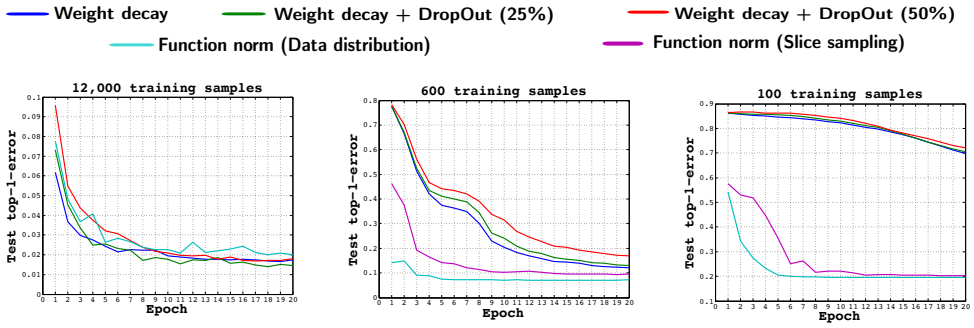

Figure 1: MNIST - Left: 12,000 samples - Center: 600 samples - Right: 100 samples

In Figure 3-left, we display the test Top-1 error evolution as a function of the training time using 100 samples for training. The number (1:N) indicates the ratio (Training samples:Regularization samples). Table 1 shows the accuracy obtained on the test set at the end of the training. As a reference, note that the best accuracy obtained when all the training data is used is 99.06%.

Training	Regularization	Weight decay	WD + DO(25%)	WD + DO(50%)	Function norm (data dist.)	Function norm (slice sampling)
12,000	48,000	98.26	97.65	95.82	97.99	97.99 (1:4)
1,200	58,800	92.60	89.93	85.52	95.15	92.70 (1:4)
600	59,400	88	83.12	73.97	92.84	90.47 (1:10)
100	59,900	30.27	18.75	13.36	80.53	79.74 (1:40)

Table 1: MNIST - Correct classification rate (%) for different methods

CIFAR: In this paragraph, we report the results of our algorithms when applied to CIFAR 10. The database is composed of 50,000 samples for training and 10,000 samples for test. These experiments where conducted on a machine equipped with a 4 core CPU (4.00 GHz) and a GPU GeForce GTX 750 Ti. In this set of experiments, we test our two algorithms, weight decay and weight decay + DropOut (10% and 50%) with a decreasing number of samples of CIFAR in the training data set. The

remaining training samples are used for regularization using data distribution. For the slice sampling method, we generate 5000 samples in the case of 500 training samples, and 4000 samples in the case of 100 samples. Figure 2 shows the Top-1 error for the test set as a function of the number of epochs using respectively 10,000, 500, and 100 training samples. Figure 3-right shows the evolution of the test Top-1 error while using 500 samples for training. Table 2 shows the accuracy obtained on the test set at the end of the training. As a reference, note that the best accuracy obtained when all the training data is used is 80.38%.

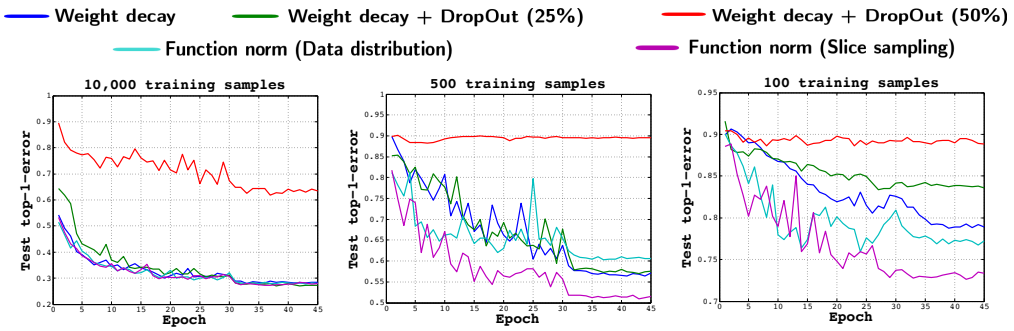

Figure 2: CIFAR - Left: 10,000 samples - Center: 500 samples - Right: 100 samples

Training	Regularization	Weight decay	WD + DO(10%)	WD + DO(50%)	Function norm (data dist.)	Function norm (slice sampling)
10,000	40,000	71.53	69.35	40.79	71.90	72.02 (1:4)
5,000	45,000	67.27	64.59	28.08	68.26	68.21 (1:4)
1,000	49,000	52.57	47.99	12.03	50.75	54.38 (1:10)
500	49,500	43.64	37.16	11.57	39.96	48.47 (1:10)

Table 2: CIFAR- Correct classification rate (%) for different methods

4 Conclusion

In this work, two methods to introduce function norm regularization in neural network training have been developed, one of which does not require any additional training data. We have demonstrated both theoretically and empirically that their integration into stochastic backpropagation converges. Their efficiency have been shown on the MNIST and CIFAR10 data sets. Our experiments suggest that the developed methods are of high quality on small databases that lie in low dimensional manifolds. For more complicated data with a small number of samples, the results of our methods outperform the state-of-the-art regularization methods available in the literature. Moreover, we have shown that both of the sampling methods yield similar results, which suggests that they are equally good approximations of the function norm.

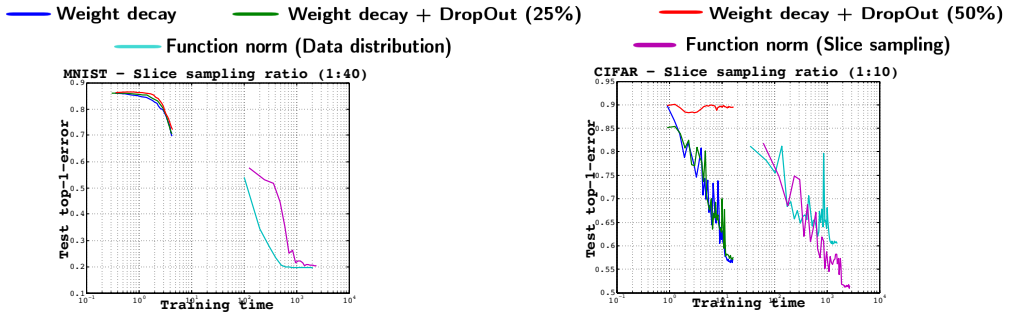

Figure 3: Sampling method comparison - Test Top1-error evolution in time - Left : MNIST, 100 samples/ Right: CIFAR, 500 samples

References

[1] R. E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7:1–49, 1998.

[2] G. Casella and E. I. George. Explaining the Gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[3] J. Galambos. *Advanced Probability Theory*. CRC Press, 1995.

[4] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Book in preparation for MIT Press, 2016.

[5] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

[7] A. J. Lee. *U-Statistics: Theory and Practice*. CRC Press, 1990.

[8] J. Moody, S. Hanson, A. Krogh, and J. A. Hertz. A simple weight decay can improve generalization. *Advances in Neural Information Processing Systems*, 4:950–957, 1995.

[9] R. M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 06 2003.

[10] A. Rannen Triki and M. B. Blaschko. Stochastic function norm regularization of deep networks. *CoRR*, abs/1605.09085, 2016.

[11] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2001.

[12] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for MATLAB. In *Proceedings of the ACM International Conference on Multimedia*, 2015.

[13] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1058–1066, 2013.