Investigating experimentaly Deep Learning without poor local minima by Kawaguchi

Laurent Cetinsoy LAURENT.CETINSOY@GMAIL.COM
Ahmed Mazari AHMED.MAZARI@U-PSUD.FR
Hafed Rhouma HAFED.RHOUMA@U-PSUD.FR

Abstract

In a recent work by Kawaguchi (Kawaguchi, 2016), it as been claimed that under several assumptions met in practice, all local minima of the training loss function are equivalent feedforward models. An experimental assessment of such claims is proposed by training pairs of identical networks with different weight initialization. In addition to comparing loss values, several other metrics are considered in order to compare model similarity regarding label classifications and generalization.

1. Introduction

Recently, deep learning models have produced impressive results, pushing state of the art in various domain such as computer vision or natural language processing. Its recent successes is associated to the improvement of algorithms and training techniques and the increase in computing power and data access. Indeed Deep learning models are able to leverage huge amount of data to compute meaningful representations from it.

Although training several neural networks is NP-complete (Blum & Rivest, 1988), it has been shown that they could be successively trained in practice. However little is known about the theoretical properties of such models. And their loss function are notoriously hard to optimize. Various study have tried to give insight on methods for optimizing Deep models (Ngiam et al., 2011; Martens, 2010) and the landscape of the cost functions (Choromanska et al., 2015b;a). Others have tried to address the problem of weight initialization (Sutskever et al., 2013; He et al., 2015), which has a great impact on training performance, without giving more than heuristics.

Recently, (Kawaguchi, 2016) stated that all local minima of the training loss function are equivalent. In other words, finding any local minima is enough. This result could explain why local approaches like stochastic gradient Descent or Momentum mehtods works well in practice.

In this project we investigate such claim experimentally. We first review the hypothesis of the theorem in order to see if they hold in practice. Then conduct several experiments. Finally a more general question is raised: Does the problem answered by the article is well posed ? Indeed it is not because two models have the same training errors (with different parameters) that they will generalize equally.

2. Theorem hypothesis review

We briefly review the hypothesis required by the various theorems. Indeed a theorem with unrealistic hypotheses would have a lower impact. It appears that they are very flexible and seems to hold in practice.

2.1. Deep linear networks

The notations are kept identical as the original article. The author starts with Deep Linear Networks (theorem 2.3).

Let H be the number of layer of the network.

let $p = min(d_H, ..., d_1)$ be the width of the smallest layer.

Let X and Y be the feature matrix and the label matrix respectively. $X \in \mathbb{R}^{m \times d_x}$ where m is the number of sample in the data set. $Y \in \mathbb{R}^{m \times d_y}$

Under such the following hypothesis:

1. $d_y < d_x$
2. XX^T and XY^T are full rank

we have that:

1. the Loss function is non convex and non concave
2. Every local minimum is a global one

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

3. Every critical point that is not a global minimum is a saddle point
4. if $rank(W_H...W_2) = p$, then the Hessian at any saddle point has at least one negative eigenvalue

The hypotheses are often met in practice. Indeed most of the time $d_y = 1$ and $d_x >> 1$. Besides having XX^T being full rank is not restrictive. It seems unlikely that XX^T is not full rank unless two or more columns of X are colinear. This hypothesis can be be easily (although expensively) checked by computing $det(XX^T)$ or the QR decomposition of XX^T.

2.2. Deep non linear networks

The previous result is then extended to Non Linear Deep Network. The hypothesis are identical as the previous section. The deep linear network is written as:

$$h = \sigma_{H+1}(W_{H+1}\sigma_H(...\sigma_2(\sigma_1(W_1X)W_2)...))) \quad (1)$$

with $\sigma(x) = max(0, x)$ being the RELU activation function, which is the most used currently (LeCun et al., 2015).One can note that the softmax activation function is not used at the last layer.

3. Experiments

In order to checker whether all minima are equivalent w.r.t the loss function, tuples of identical feedforward networks are trained with different weight initialization until convergence. As used in the article, the mean squared loss function defined below is used.

$$\mathcal{L} = 0.5 \sum_{i=1}^{m} ||y_i - h(x_i)||_2^2 \quad (2)$$

with $\sigma(x) = max(0, x)$ as used in the article.

Weights were sampled from $\mathcal{N}(0, \sigma_i I)$ with $\sigma_i = i + 2$ and i being the i^{th} model.

The experiments were implemented with Keras on the MNIST and the Boston housing price datasets.

3.1. Metrics

In addition to that, a more general question is raised. Does two models with the same loss function value yield the same behavior. For this purpose, besides training and testing learning curves, the following metrics are proposed.

For a given pair of models, we compute the number of weights whose difference is higher than a given threshold. The idea is to try to quantify the distance between two set of weights and if they are close or not in a parameter space. Indeed seems pair of model have the same architectures weights can be compared neurons by neurons. The computation of the classical euclidian norm over all weights could be also used but might suffer from the curse of dimensionality. However the first proposed approached might also suffer from the same drawback. In a futur work, both approaches will be compared.

In addition to that we consider a analogue of the binary classification loss: The number of training example upon which the two models agree and disagree are computed.

Ideally, two model with the same loss function would classify a set of given example identically.

4. Results

A first set of experiment is done as previously described without softmax activation function. Then the same experiments are carried with it.

4.1. Deep non linear network without softmax

At first, two sets of model were trained with the Adam optimizer: One hidden layer network with 500 units and a two hidden neural network 500 and 300 units respectively.

Figure 1 show the errors of the first set of models. We can see that all models quickly converge towards zero (the global minimum) for training and testing. Accordingly as shown in figure 2, the number of training samples identically labeled by various models increase with epochs. Figure 3 shows the learning curves for the two hidden neural networks.

We can see that all models converge to the global minimum. Thus the question is: is it overfitting ? In order to check that a small model with only (10-10) hidden units was trained (figure 4). It appears that the behaviour is identical. This raises so questions concerning the methodology and the computation of the performance metrics.

4.2. Deep non linear network with softmax

A second set of experiment was conducted by replacing the last layer by a softmax one. Indeed it is often used in classification. The same sets of networks were considered. One can see on figure 5 that convergence is much slower and does not lead to the same values. This behavior can also be seen for two hidden layer networks (fig. 6). Besides model pairs usually disagree on labeling data (fig. 7).

$$h(x_i) = softmax(W_l...\sigma(W_2\sigma(W_1(x_i)))) \quad (3)$$

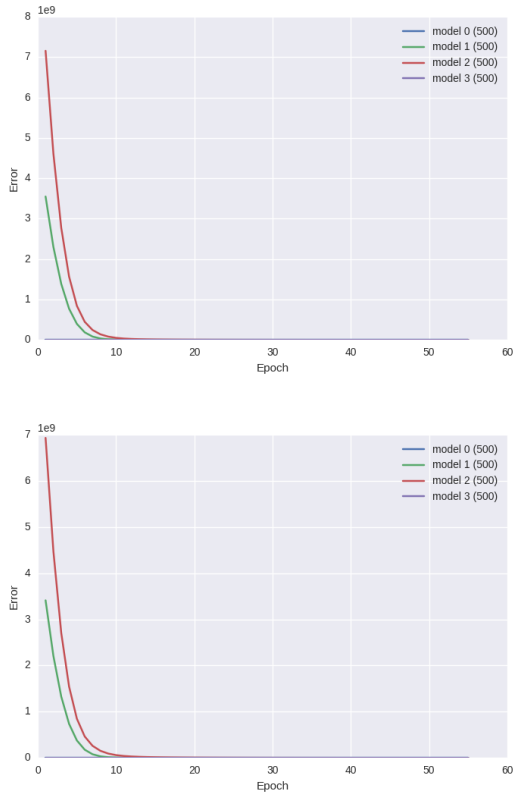

Figure 1. Training (top) and Test error error for one hidden layer neural networks with 500 units.

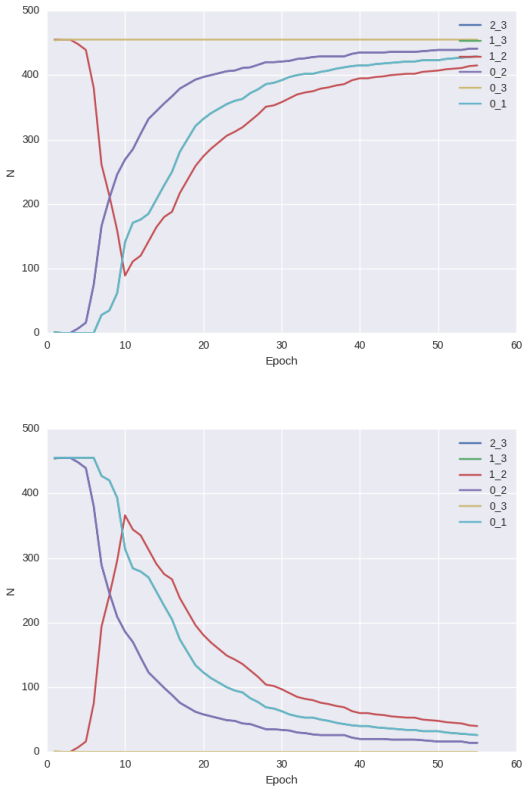

Figure 2. Number of training samples identically (top) classified and not identically classified (bottom) versus epochs on training set. The Red lines shows that at the begining two models disagree more and more on labeling data but then the dynamic is broken and they start to agree more and more with epochs

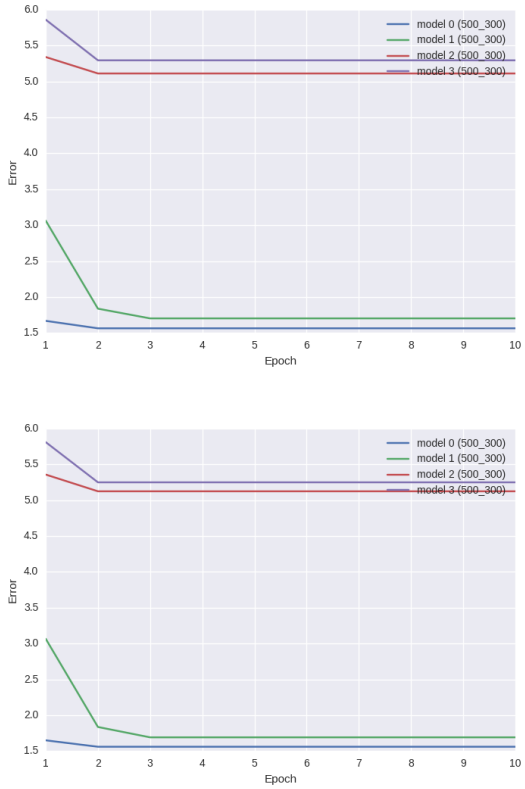

Figure 3. Training (top) and Test error error for two hidden layer neural networks with 500 and 300 units.

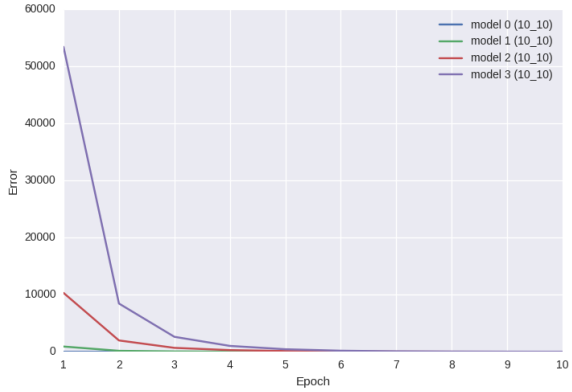

Figure 4. Train of 10-10 hidden unit networks.

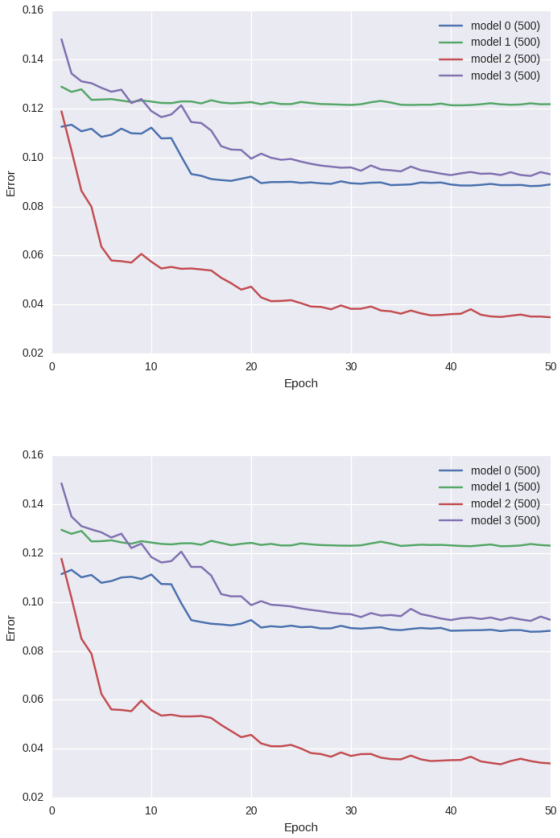

Figure 5. Train (top) and Test (bottom) error versus epochs of four one hidden layer (500 units) networks. The various models does not converge to the same loss.

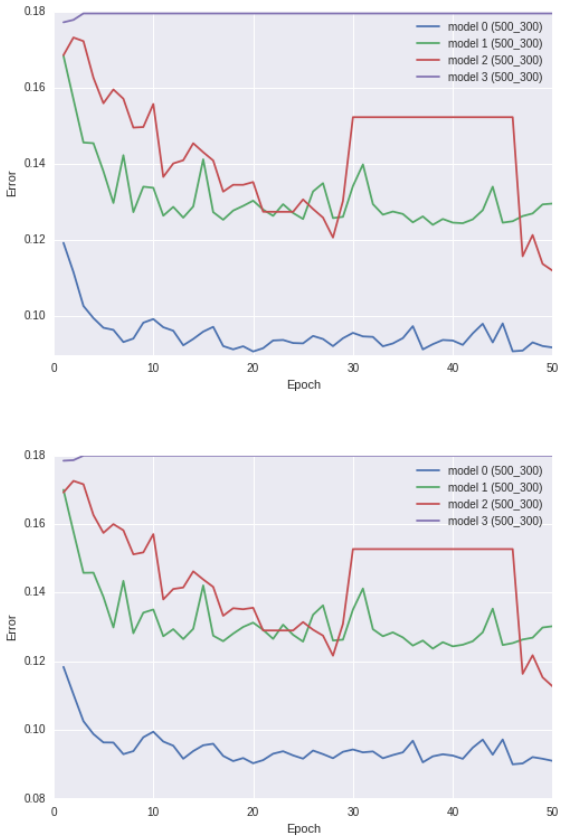

Figure 6. Training (top) and Test (bottom) error versus epochs for two hidden layer networks.

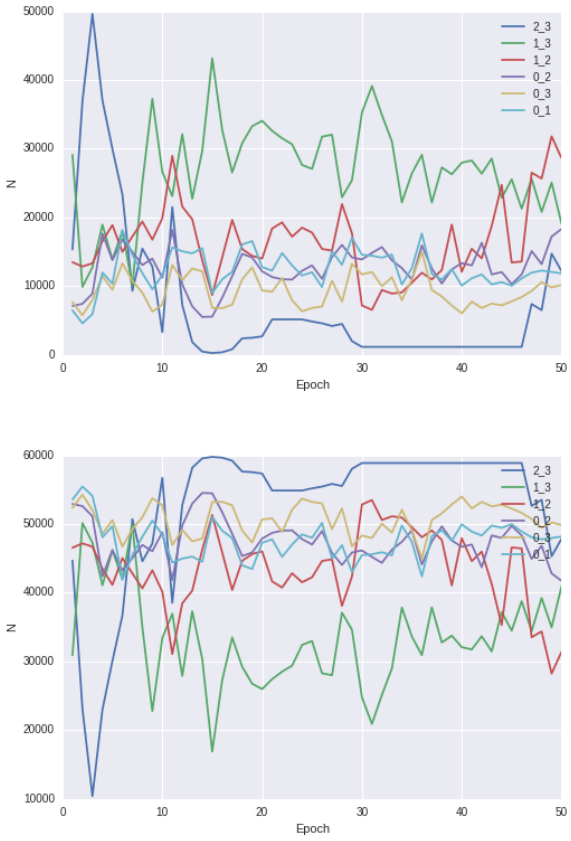

Figure 7. Number of samples identically (top) and not identically (bottom) labeled by two hidden layer network pairs (500 and 300 hidden units) using softmax output. No model agrees besides models 2 and 3 (blue line)

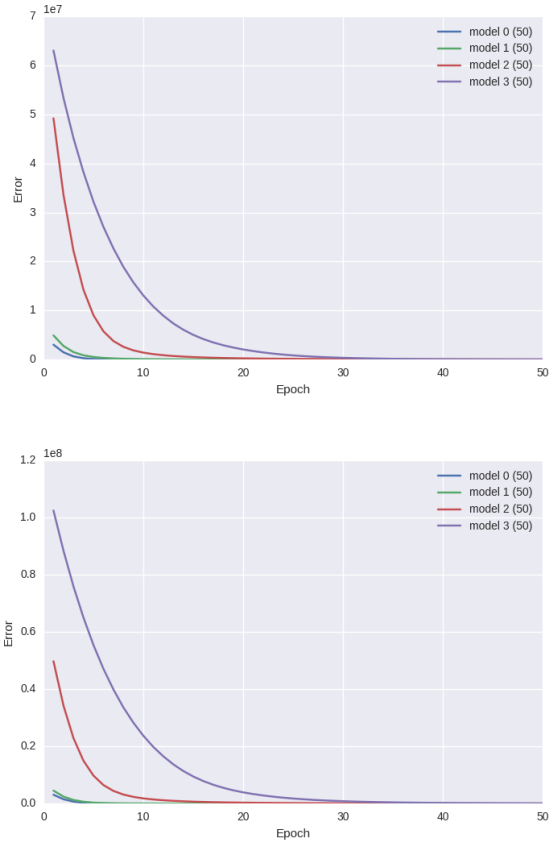

Figure 8. Training (top) and Test (bottom) error versus epochs on the Boston dataset with a 50 hidden neural network layer without softmax output.

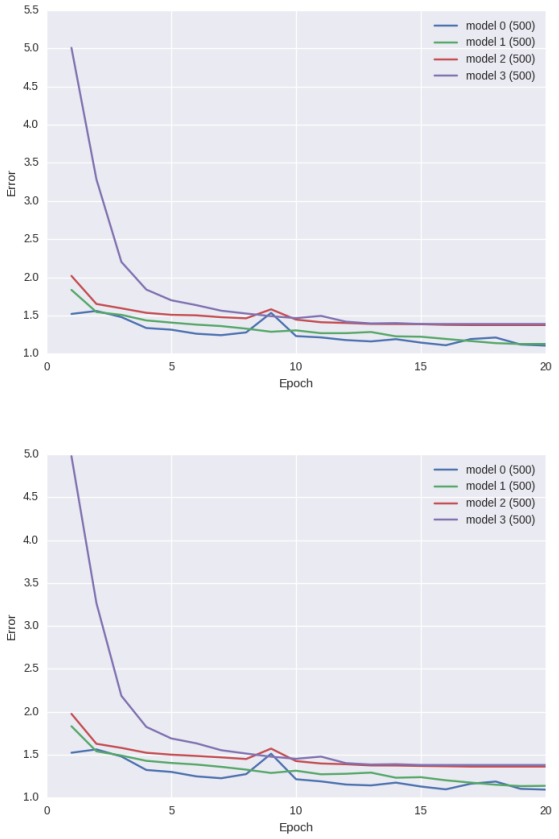

Figure 9. Training (top) and Test (bottom) error versus epochs on MNIST trained with binary cross entropy loss without softmax output. It appears that all models converge to the same minimum

For both architectures, it can be shown that the model losses does not reach the same values.

4.3. Other experiments

Several quick experiments were conducted. First on the Boston Housing data in order to run a regression model. Then a set of model was trained on MNIST using the cross entropy loss.

4.4. Boston Housing dataset

On figure 8 we can see that the loss function also converges toward zero for a one hidden layer network (50 hidden units) without softmax activations.

4.5. Cross entropy loss

Finally the cross entropy loss was used to train several models. Figure 9 shows the learning curves. Two sets behaviours can be noted: models 2 and 3 and models 0 and 1. However further training should be done in order to draw conclusions: they may all converge with more iterations.

5. Discussion and further works

Networks without the softmax activation converge much quicker than models with softmax activations functions. It seems that they are always able to converge to zero. If it is indeed the case, then it would explain and generalize the results of Kawaguchi: all global minimum are 0. However before being able to make such claim the methodology must be assessed.

The second goal of the project was also to question if the problem addressed by Kawaguchi is well posed. Indeed two models having the same loss on a training set can have different performance in generalizing. Moreover two identical loss functions does not necessarily mean that the training examples will be label identically. It appears that for models without softmax, they both have the same performance on training and generalization set.

Due to a lack of time, the analysis of the converge with Hessians or weight pertubations were not carried out. The following experiments are considered as future works :

- Verify the methodology around training models without softmax activation
- Training model with L1 or L2 regularization.
- Dropout regularization and batch normalization.
- Conducting theses experiments on Convolutional Networks and Residual ones.
- Same for Recurrent neural networks and GAN. The latter are considered as much harder to train.
- Conducting experiments to explore the landscape of the loss function

As conclusion the following questions are asked. First, is the softmax approach justified ? It is widely used as it allows to interpret model ouputs as probability. However it seems to reduce the performance of models a lot. Besides recent bayesian approaches (Gal & Ghahramani, 2015) have criticized the probability interpretation. Finally, the choice of the loss function seems to have a non trivial impact on the landscape of optimization problem. The mean square loss function seems to behave nicely as two models havings same loss value will label training points identically.

References

Blum, Avrim and Rivest, Ronald L. Training a 3-node neural network is np-complete. In *Proceedings of the 1st International Conference on Neural Information Processing Systems*, pp. 494–501. MIT Press, 1988.

Choromanska, Anna, Henaff, Mikael, Mathieu, Michael, Arous, Gérard Ben, and LeCun, Yann. The loss surfaces of multilayer networks. In *AISTATS*, 2015a.

Choromanska, Anna, LeCun, Yann, and Arous, Gérard Ben. Open problem: The landscape of the loss surfaces of multilayer networks. In *COLT*, pp. 1756–1760, 2015b.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2, 2015.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kawaguchi, Kenji. Deep learning without poor local minima. In *Advances In Neural Information Processing Systems*, pp. 586–594, 2016.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521(7553):436–444, 2015.

Martens, James. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 735–742, 2010.

Ngiam, Jiquan, Coates, Adam, Lahiri, Ahbik, Prochnow, Bobby, Le, Quoc V, and Ng, Andrew Y. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 265–272, 2011.

Sutskever, Ilya, Martens, James, Dahl, George E, and Hinton, Geoffrey E. On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147, 2013.