SELF-STABILIZED DEEP NEURAL NETWORK

Pegah Ghahremani, Johns Hopkins University and Jasha Droppo, Microsoft Research

Introduction

Deep neural network models have been successfully applied to many tasks such as image labeling and speech recognition.

Mini-batch stochastic gradient descent is the most prevalent method for training these models.

A critical part of successfully applying Mini-batch SGD
- choosing appropriate initial values
- local and global learning rate scheduling algorithms.

In this paper, we present a method which:
- Makes training converge faster,
- Makes training less sensitive to global learning rate, and
- Produces better models.

Learning Rate Scheduling

The goal of training a DNN is to update the set of parameters in order to optimize objective function F. Gradient descent algorithms accomplish this by following steps against the gradient's direction. The size of the step is controlled by the learning rate. In practice, a good learning rate schedule can result in faster convergence to a better local optimum.

Global learning rates are applied to all parameters
- Exponential or power scheduling
- Algorithmic scheduling

Local learning rates are customized for each parameter
- Natural Gradient
- Natural Newton
- AdaGrad and DeltaGrad

These methods introduce hyper-parameters that should be tuned on development data to give the best performance.

Self-Stabilized Parameters are Local Learning Rate Adjustment

The self-stabilizing parameter are a scalar parameter that augments each matrix parameter of the DNN model.

Compare how this changes the derivation of parameter update equations, both without (left) and with (right) the self-stabilizing parameter.

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} \qquad \boldsymbol{y} = e^{\beta}\boldsymbol{W}\boldsymbol{x}$$

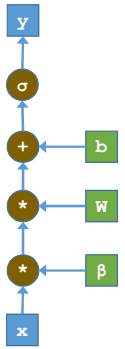
If the gradient of the objective function F with respect to $\boldsymbol{y}$ is known, then the gradient of F with respect to the parameter $\boldsymbol{W}$ is given by:

$$\frac{dF}{d\boldsymbol{W}} = \frac{dF}{d\boldsymbol{y}}\boldsymbol{x}^T \qquad \frac{dF}{d\boldsymbol{W}} = e^{\beta}\frac{dF}{d\boldsymbol{y}}\boldsymbol{x}^T$$

The parameter update equation multiplies this gradient by a learning rate η.

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta\frac{dF}{d\boldsymbol{y}}\boldsymbol{x}^T \quad \boldsymbol{W} \leftarrow \boldsymbol{W} - \eta e^{\beta}\frac{dF}{d\boldsymbol{y}}\boldsymbol{x}^T$$

With $\beta = 0$, the two updates are equivalent. **Other values of β act as a local modification of the learning rate applied to updating the parameter $\boldsymbol{W}$.**



Self-Stabilized Parameter Update

What controls the stabilizing parameter? Observe the derivation of its update equation.

$$\boldsymbol{y} = e^{\beta}\boldsymbol{W}\boldsymbol{x}$$

The gradient with respect to the input is given by,

$$\frac{dF}{d\boldsymbol{x}} = e^{\beta}(W)^T\frac{dF}{d\boldsymbol{y}}$$

And the gradient with respect to β is given by,

$$\frac{dF}{d\boldsymbol{\beta}} = e^{\beta}\left(\frac{dF}{d\boldsymbol{y}}\right)^T\boldsymbol{W}\boldsymbol{x} = \left(\frac{dF}{d\boldsymbol{x}}\right)^T\boldsymbol{x}$$

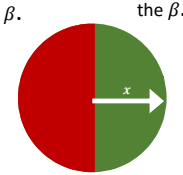
Yielding the update rule,

$$\beta \leftarrow \beta - \eta\left(\frac{dF}{d\boldsymbol{x}}\right)^T\boldsymbol{x}$$

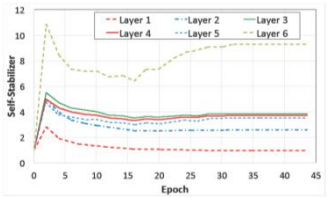
Clearly, when increasing the magnitude of $\boldsymbol{x}$ will improve the objective function, β will increase.

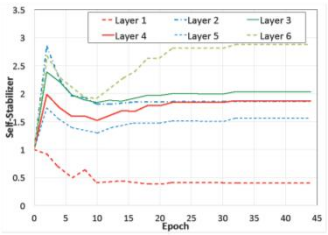
Gradients opposing the direction of $\boldsymbol{x}$ increase the β.

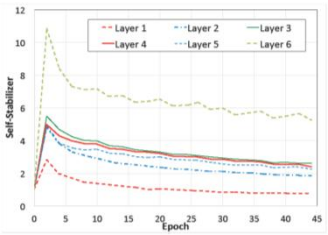
Gradients in the direction of $\boldsymbol{x}$ decrease the β.



Process is stable when gradients and $\boldsymbol{x}$ are uncorrelated.







The self-stabilizing parameter adapts to different global learning rates.
- The top figure uses a learning rate 1/8 the value of the learning rate in the middle figure.
- The learned parameters, which effectively adjust the local learning rate, are two to three times bigger.

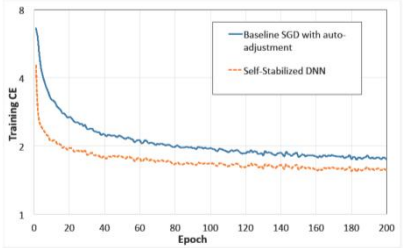
The self-stabilizing parameter adopts a diminishing learning rate schedule over time.
- The top and bottom figures use the same initial learning rate.
- The top figure uses the CNTK-AA algorithm to decrease the global learning rate over time.
- The bottom figure holds the global learning rate fixed, but the self-stabilizing parameters decrease over time.

Self Stabilizer	Learning Rate Scheduler	Initial Learning Rate	Training Frame Error	Validation Frame Error	Validation Word Error
No	CNTK-AA	0.1	53.1%	57.3%	40.2%
No	CNTK-AA	0.8	43.5%	51.0%	32.4%
Yes	CNTK-AA	0.1	38.7%	49.8%	32.5%
Yes	CNTK-AA	0.8	39.8%	49.7%	**32.0%**
No	AdaDelta	0.8	44.6%	52.0%	33.5%
Yes	AdaDelta	0.8	39.2%	51.3%	32.2%
Yes	None	0.1	**36.3%**	51.0%	32.1%
Yes	None	0.1	**37.3%**	50.7%	**31.8%**

The self-stabilizing parameter produces better models, more quickly.

The figure on the right shows training cross entropy dropping more quickly, and producing a better final value.

Switchboard + Fisher training data with 9000 class labels, six layer Sigmoid DNN.



Results on AMI Using Various Training Methods

- SS makes training more robust to initial learning rate. (lines 1+2, lines 3+4).
- SS Improves training for CNTK-AA or AdaDelta (lines 1+3, lines 2+4, lines 5+6).
- SS gives nearly the best results, even without a global learning rate schedule (line 7)
- CNTK-AA is an auto-adjust learning rate algorithm "adjust after epoch" , which reduces global learning rate by some factor if the cross entropy objective degrades on validation set.
- L2 regularization technique on parameters used in last experiments to reduce overtraining to training data.
- The overtraining is no longer a problem on large training sets (e.g. SWBD + Fisher)

Conclusion

The self-stabilizing parameters control the activation distribution throughout training. In our paper, we show these parameter interact positively with stochastic gradient training, yielding models that:

- Are less sensitive to initial learning rate
- Are less sensitive to parameter initialization
- Are still compatible with learning rate scheduling algorithms, and
- **Converge more quickly to better local optimum.**