Improved Optimization of Neural Networks

Simon Wiesler

Human Language Technology and Pattern Recognition
Computer Science Department, RWTH Aachen University
D-52056 Aachen, Germany

March 18, 2014

Outline

Outline

Introduction

Mean-Normalized Stochastic Gradient Descent

Experimental Results

Implementation of Neural Networks in RASR

Conclusion

Motivation

- Deep Neural Networks (DNNs) outperform Gaussian mixtures on large-scale speech recognition tasks [Mohammed$^+$, NIPS 2009], [Seide$^+$, Interspeech 2011], [Hinton$^+$, SPM 2012], ...
- The training of DNNs is a difficult non-convex optimization problem. Even with the use of GPUs, training time of DNNs is high.
- Improving optimization algorithms can accelerate training.

Motivation

- Deep Neural Networks (DNNs) outperform Gaussian mixtures on large-scale speech recognition tasks [Mohammed^{+}, NIPS 2009], [Seide^{+}, Interspeech 2011], [Hinton^{+}, SPM 2012], ...
- The training of DNNs is a difficult non-convex optimization problem. Even with the use of GPUs, training time of DNNs is high.
- Improving optimization algorithms can accelerate training.
- For large-scale tasks: limited computation time
  $\Rightarrow$ Better optimization algorithms can improve accuracy of DNNs, see [Bouttou^{+}, NIPS 2007] .

Conventional Training of DNNs

- Typically, DNNs are trained according to the cross-entropy criterion:

$$F(W) = -\sum_{t=1}^{T} \ln p_W(s_t|x_t).$$

Here, $(x_t, s_t)_{t=1,\ldots,T}$ denotes the training sample, x_t an acoustic observation, s_t an HMM state, and $p_W(s_t|x_t)$ the output of the softmax-layer of a DNN with parameters W.

- F is optimized with stochastic gradient descent (SGD):

$$W_n = W_{n-1} - \gamma_n \nabla F_{\mathcal{B}_n}(W_{n-1}) \ .$$

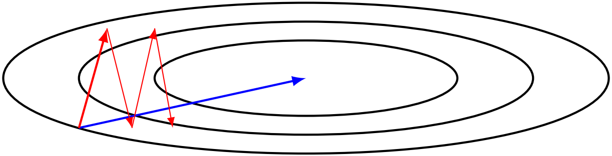
The stochastic gradient $\nabla F_{\mathcal{B}_n}$ is computed on a small random subset of the training data $\mathcal{B}_n$ ("mini-batch").

Second-Order Optimization: The Problem

- SGD scales well to large datasets due to its stochastic nature.

Second-Order Optimization: The Problem

- SGD scales well to large datasets due to its stochastic nature.
- But no use of second-order information
  $\Rightarrow$ May converge very slowly ("zig-zag behavior").



- Second-order algorithms use a better search direction than the steepest descent, typically an approximation to the Newton direction:

$$W_n = W_{n-1} - \gamma_n C_n \nabla F_{\mathcal{B}_n}(W_{n-1}) \, .$$

Second-Order Optimization: Related Work

Batch algorithms:

- L-BFGS [Byrd$^+$, J.Sc.Comp. 1995],
- Rprop [Riedmiller$^+$, ICNN 1993],
- Hessian-Free [Martens, NIPS 2010], [Kingsbury$^+$, Interspeech 2012], [Wiesler$^+$, Interspeech 2013].

Stochastic algorithms:

- diagonal Hessian approximation [LeCun$^+$, 1998],
- online L-BFGS [Schraudolph$^+$, AIStats 2007],
- AdaGrad [Duchi$^+$, JMLR 2010], [Dean$^+$, NIPS 2012], ...

Here: present new stochastic second-order algorithm:

S. Wiesler, A. Richard, R. Schlüter, H. Ney: Mean-Normalized Stochastic Gradient for Large-Scale Deep Learning. In ICASSP 2014 (to appear).

IBM Research Spoken Language Processing Student Paper Award

Outline

Introduction

Mean-Normalized Stochastic Gradient Descent

Experimental Results

Implementation of Neural Networks in RASR

Conclusion

Mean-Normalized SGD: Idea

- It is known that the convergence behavior of NN training depends on mean, variance, and correlation of the features. [LeCun^{+}, NIPS 1991], [Wiesler^{+}, NIPS 2011]
- Feature normalization makes the gradient closer to the optimal search direction.

Mean-Normalized SGD: Idea

- It is known that the convergence behavior of NN training depends on mean, variance, and correlation of the features. [LeCun$^+$, NIPS 1991], [Wiesler$^+$, NIPS 2011]
- Feature normalization makes the gradient closer to the optimal search direction.
- For deep networks: Impact of feature normalization is limited.
- Basic idea: Apply mean normalization to input of all layers.
- But: Normalization can not be performed beforehand.

Mean-Normalized SGD: Idea

- It is known that the convergence behavior of NN training depends on mean, variance, and correlation of the features. [LeCun$^+$, NIPS 1991], [Wiesler$^+$, NIPS 2011]
- Feature normalization makes the gradient closer to the optimal search direction.
- For deep networks: Impact of feature normalization is limited.
- Basic idea: Apply mean normalization to input of all layers.
- But: Normalization can not be performed beforehand.
- Solution: Perform normalization implicitly in optimization algorithm.
- Mean statistics are computed as running averages.

Mean-Normalized SGD: Derivation

- Consider a single layer of a NN for simplicity.
- Let $W \in \mathbf{R}^{D' \times D}$ be the weight matrix, $a \in \mathbf{R}^{D'}$ the bias, and $y \in \mathbf{R}^{D}$ an input to the layer. Then:

$$F : \mathbf{R}^{D' \times (D+1)} \to \mathbf{R}, \quad (W, a) \mapsto \mathcal{G}(W\,y + a)\ .$$

- Mean normalization is a shift by a vector b:

$$\begin{aligned}\tilde{F} : \mathbf{R}^{D' \times (D+1)} \to \mathbf{R}, \quad (W, a) &\mapsto \mathcal{G}(W(y + b) + a) \\ &= \mathcal{G}(W\,y + W\,b + a) = F(W, W\,b + a)\ .\end{aligned}$$

$\Rightarrow$ Normalization can be performed on model side.

- Mapping from original to new parameter space:

$$\phi(W, a) := (W, a - W\,b)\ ,$$
$$F(W, a) = (\tilde{F} \circ \phi)(W, a)\ .$$

Mean-Normalized SGD: Derivation

- Define the MN-SGD update rule as:

$$(\hat{W}, \hat{a}) = \phi^{-1}\Big(\phi(W, a) - \gamma \cdot (\nabla \tilde{F})\big(\phi(W, a)\big)\Big) \ .$$

- Application of chain rule yields ...

$$\begin{aligned}\hat{W} &= W - \gamma \cdot \left(\nabla_W F(W, a) + b \cdot \nabla_a F(W, a)^T\right) \\ \hat{a} &= a - \gamma \cdot \left(\nabla_W F(W, a) b + (1 + b^T b) \nabla_a F(W, a)\right) \ .\end{aligned}$$

Mean-Normalized SGD: Derivation

- Define the MN-SGD update rule as:

$$(\hat{W}, \hat{a}) = \phi^{-1}\Big(\phi(W, a) - \gamma \cdot (\nabla \tilde{F})\big(\phi(W, a)\big)\Big) \ .$$

- Application of chain rule yields ...

$$\boxed{\begin{aligned} \hat{W} &= W - \gamma \cdot \left(\nabla_W F(W, a) + b \cdot \nabla_a F(W, a)^T\right) \\ \hat{a} &= a - \gamma \cdot \left(\nabla_W F(W, a) b + (1 + b^T b) \nabla_a F(W, a)\right) \ . \end{aligned}}$$

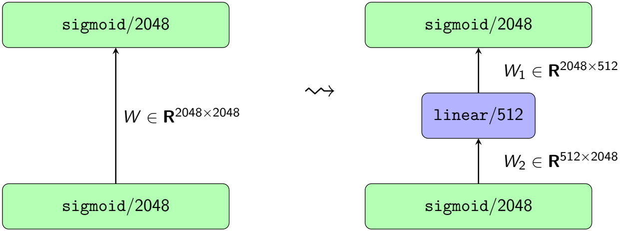
- Rule is expressed in terms of the gradient of F.
- Computing the update is cheap.
- Can be written in the form of a second-order algorithm.
- Convergence proof in [Wiesler$^+$, ICASSP 2014].

Second-Order Optimization and Regularization

- Problem: Improved optimization can lead to overfitting.
- Conventional ℓ_2- or ℓ_1-regularization is typically not very effective for NN acoustic models.
- Alternative: Limiting number of parameters.
- Additional positive effect: Reduces time and memory requirements in training and recognition.

Linear Bottlenecks

- Specific network structure proposed by [Sainath⁺, ICASSP 2013] for speeding up NN training.
- Example:



- Question: Consequence for performance?
- Note: Equivalent to a low rank approximation: $W \approx W_1 W_2$.

Linear Bottlenecks: Problems

- Observation of [Sainath$^+$, ICASSP 2013]: Training NNs with linear bottlenecks from scratch: works only for output layer, performance degradation for hidden layers.
- [Xue$^+$, Interspeech 2013] showed: linear bottlenecks are possible between all layers when
  1. training a full network,
  2. factorizing the weight matrices with SVD and reducing their rank,
  3. fine-tuning the smaller network.
- Only beneficial for recognition.
- Conclusion: optimization problem!

Outline

Experimental Setup

- Corpus: Quaero English (English broadcast conversations)
- Experiments on 50h subset and full corpus.

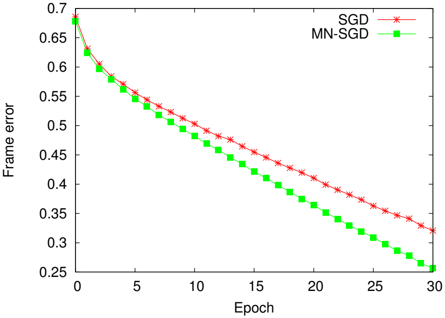
	#Running Words	Duration
Train50h	300k	50h
TrainComplete	1.7M	268h
dev (quaero-eval10)	41k	3.7h
test (quaero-eval11)	35k	3.3h

- Lexicon size: 150k words.
- OOV rate:
  - 0.39% on dev,
  - 0.44% on test.
- 4-gram LM with perplexity
  - 123.0 on dev,
  - 135.6 on test.

Neural Network Training

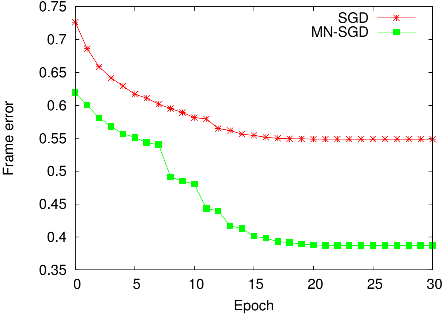
- Feature extraction: 16-dim MFCC vector, sliding window of size 17, $\Delta + \Delta\Delta \Rightarrow$ 493-dim feature vector, global mean and variance normalization.
- Model: six hidden layers of size 2048 with sigmoid activation, softmax-output layer of size 4498.
- Training criterion: cross-entropy.
- Optimization: SGD vs. MN-SGD with mini-batch size 512, supervised pretraining [Seide$^+$, ASRU 2011].
- Initial baseline with learning rate schedule Newbob: 19.8% WER on dev, 25.7% WER on test.
- Improved baseline with better learning rate strategy: optimize on train, use early stopping for regularization: 18.7% WER on dev, 24.7% WER on test.

Training Error of SGD and MN-SGD



- Training frame error of SGD and MN-SGD for full-sized models.

Training Error of SGD and MN-SGD



- Training frame error of SGD and MN-SGD for models with bottlenecks of size 256.
- Note: Already the pre-training results of SGD and MN-SGD differ strongly.

Recognition Results: 50h

Algorithm	Bottleneck	#Parameters	WER (dev)	WER (test)
SGD	-	31.2M	18.7	24.7
	512	14.8M (47.7%)	18.9	25.0
	256	7.9M (25.5%)	19.7	25.7
MN-SGD	-	31.2M	19.0	25.5
	512	14.8M (47.7%)	18.7	24.7
	256	7.9M (25.5%)	18.4	24.2
	128	4.9M (15.7%)	18.6	24.2
	64	2.8M (8.9%)	19.3	25.3

- 0.5% WER improvement with only 25.5% parameters
- Even with only 15.7% parameters an improvement in WER is observed.

Recognition Results: Full Training Set

Algorithm	Bottleneck	#Parameters	WER (dev)	WER (test)
SGD	-	31.2M	15.1	20.3
	512	14.8M (47.7%)	15.8	21.0
MN-SGD	-	31.2M	15.0	20.2
	512	14.8M (47.7%)	14.9	19.8
	256 (depth 8)	10.0M (32.0%)	14.9	19.9

- 0.4% WER improvement with only 32.0% parameters.

Outline

The RWTH NN Toolkit for Speech Recognition

- RWTH RASR is the open source version of the RWTH speech recognition toolkit.
- Development started by Max Bisani and Stephan Kanthak in 2001, and further developed since then.
- At RWTH, RASR is used for all project evaluations and research.

The RWTH NN Toolkit for Speech Recognition

- RASR is the open source version of the RWTH speech recognition toolkit.
- Development started by Max Bisani and Stephan Kanthak in 2001, and further developed since then.
- At RWTH, RASR is used for all project evaluations and research.
- Previous versions of RASR did not have support for NNs.
- In collaboration with Alexander Richard and Pavel Golik: Wrote the RASR NN module.
- Part of the latest version of RASR, available at `http://www-i6.informatik.rwth-aachen.de/rwth-asr`
- S. Wiesler, A. Richard, P. Golik, R. Schlüter, H. Ney: RASR/NN : The RWTH Neural Network Toolkit for Speech Recognition. In ICASSP 2014 (to appear).

RASR/NN: Features

- Highly efficient GPU and CPU-multithreading implementation.
- Directly integrated in speech recognition software.
- Generic code basis, clean interfaces.

Support for

- feed-forward NNs of a very general form: must be representable by a DAG,
- many activation functions: sigmoid, tanh, softmax, RLU,
- various training criteria: cross-entropy, squared-error, binary divergence,
- regularization: ℓ_1, ℓ_2, dropout,
- different optimization algorithms: SGD, MN-SGD, momentum, Rprop.
- ...

All components can be configured independently.

Experimental Comparison with QuickNet

- Experiments with a DNN with 493 inputs, six 2048-dim. hidden layers, 4498 outputs. Word error rates in %.

	Learning rate schedule			
	QuickNet		**RASR**	
Training using	Dev	Test	Dev	Test
QuickNet	19.6	26.2	19.4	25.9
RASR	19.8	25.7	19.2	25.4

- RASR uses an improved version of the Newbob learning rate schedule.
- With the same learning rates: comparable results.

Runtime Analysis of RASR and QuickNet

- Runtime was measured on an Nvidia Tesla K20c (GPU) and a 12-core AMD processor (CPU).

Model	Hardware	Implementation	Time / Epoch	Speedup
6x2048	GPU	QuickNet	58.2m	
		RASR	37.1m	1.8
	CPU	QuickNet	1773.3m	
		RASR	549.3m	3.2

- Even larger difference for shallow networks.

Outline

Conclusion

- Developed MN-SGD, a new stochastic second-order optimization algorithm.
- Linear bottlenecks act as a regularization method, but make optimization more difficult.
- MN-SGD + linear bottlenecks outperforms baseline with 85% less parameters.
- Efficient & generic NN implementation, publicly available as part of the latest RASR release.

Thank you for your attention

Simon Wiesler

wiesler@cs.rwth-aachen.de

http://www-i6.informatik.rwth-aachen.de

References I

L. Bottou, O. Bousquet.
The tradeoffs of large scale learning.
In *Advances in Neural Information Processing Systems 21*, Vol. 4, 2, Dec. 2007.

R. Byrd, P. Lu, J. Nocedal, C. Zhu.
A limited memory algorithm for bound constrained optimization.
SIAM Journal on Scientific Computing, Vol. 16, pp. 1190–1208, 1995.

J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. W. Senior, P. A. Tucker, K. Yang, A. Y. Ng.
Large scale distributed deep networks.
In *Advances in Neural Information Processing Systems 25*, pp. 1232–1240, Dec. 2012.

J. Duchi, E. Hazan, Y. Singer.
Adaptive subgradient methods for online learning and stochastic optimization.
J. Mach. Learn. Res., Vol. 12, pp. 2121–2159, 2010.

G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, B. Kingsbury.
Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups.
IEEE Signal Processing Magazine, Vol. 29, No. 6, pp. 82–97, Nov 2012.

B. Kingsbury, T. N. Sainath, H. Soltau.
Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization.
In *Proc. Interspeech*, Sept. 2012.

References II

Y. LeCun, L. Bottou, G. B. Orr, K.-R. Müller.
Efficient backprop.
In *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 1998.

Y. LeCun, I. Kanter, S. Solla.
Second-order properties of error surfaces: learning time and generalization.
In *Advances in Neural Information Processing Systems 4*, Vol. 4, pp. 918–924, Denver, CO, April 1991.

J. Martens.
Deep learning via hessian-free optimization.
In *Proc. of the 27th Int. Conf. on Machine Learning*, Vol. 951, 2010, June 2010.

A.-r. Mohamed, G. Dahl, G. Hinton.
Deep belief networks for phone recognition.
In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, Dec. 2009.

M. Riedmiller, H. Braun.
A direct adaptive method for faster backpropagation learning: The Rprop algorithm.
In *Proc. of the Int. Conf. on Neural Networks*, pp. 586–591, March 1993.

T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, B. Ramabhadran.
Low-rank matrix factorization for deep neural network training with high-dimensional output targets.
In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 6655–6659. IEEE, May 2013.

N. Schraudolph, J. Yu, S. Günter.
A stochastic quasi-newton method for online convex optimization.
In *Proc. Int. Conf. on Artificial Intelligence and Statistics*, pp. 436–443, San Juan, Puerto Rico, March 2007.

References III

F. Seide, G. Li, X. Chen, D. Yu.
Feature engineering in context-dependent deep neural networks for conversational speech transcription.
In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pp. 24–29, Waikoloa, Hawaii, USA, dec 2011.

F. Seide, G. Li, D. Yu.
Conversational speech transcription using context-dependent deep neural networks.
In *Proc. Interspeech*, pp. 437–440, Aug. 2011.

S. Wiesler, J. Li, J. Xue.
Investigations on hessian-free optimization for cross-entropy training of deep neural networks.
In *Proc. Interspeech*, pp. 3317–3321, Lyon, France, Aug. 2013.

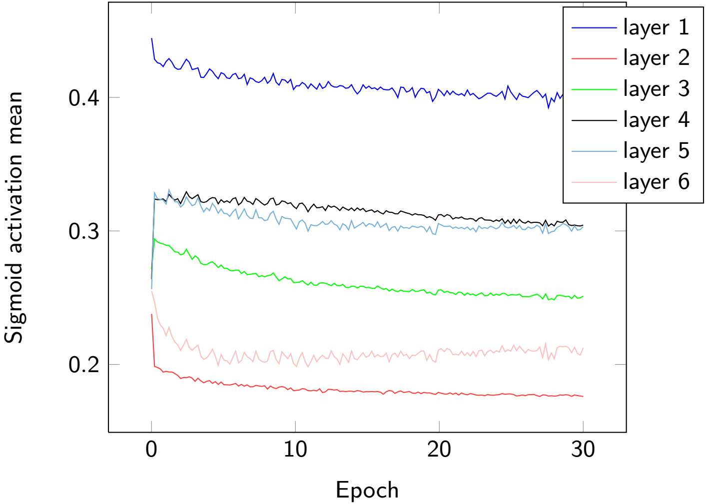
S. Wiesler, H. Ney.
A convergence analysis of log-linear training.
In *Advances in Neural Information Processing Systems 24*, pp. 657–665, Dec. 2011.

S. Wiesler, A. Richard, R. Schlüter, H. Ney.
Mean-normalized stochastic gradient for large-scale deep learning.
In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (accepted for publication)*, Florence, Italy, May 2014.

J. Xue, J. Li, Y. Gong.
Restructuring of deep neural network acoustic models with singular value decomposition.
In *Proc. Interspeech*, pp. 2365–2368, Lyon, France, Aug. 2013.

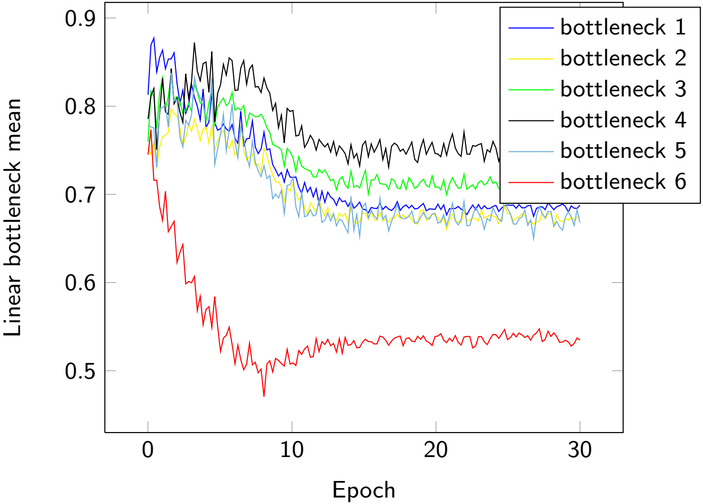
Activation Mean: Sigmoid

- Average magnitude of the activation mean per sigmoid layer (index from input to output layer).



Activation Mean: Sigmoid + Bottlenecks

- Average magnitude of the activation mean per linear bottleneck.



Results with RLUs

- Use Rectified linear units instead of sigmoid activations.
- Additional ℓ_2-regularization required.

Algorithm	#Parameters	Bottleneck	WER (dev)	WER (test)
SGD	31.2M	-	18.1	24.1
	14.8M (47.7%)	512	18.1	23.9
	7.9M (25.5%)	256	18.0	24.1
MN-SGD	31.2M	-	18.8	25.0
	14.8M (47.7%)	512	17.8	23.8
	7.9M (25.5%)	256	17.7	23.3
	4.9M (15.7%)	128	18.0	23.8
	2.8M (8.9%)	64	19.0	24.9

Low-Rank Factorization

Example:

- $W \in \mathbf{R}^{2048\times2048}$, $W_1 \in \mathbf{R}^{2048\times512}$, $W_2 \in \mathbf{R}^{512\times2048}$.
- Number of parameters: $2048 * 2048$ vs $2 * 2048 * 512$.
  $\Rightarrow$ Parameter reduction by a factor of 2.