# A CRITICAL EVALUATION OF STOCHASTIC ALGORITHMS FOR CONVEX OPTIMIZATION

*Simon Wiesler*[1]*, Alexander Richard*[1]*, Ralf Schlüter*[1]*, Hermann Ney*[1,2]

[1]Human Language Technology and Pattern Recognition,
Computer Science Department, RWTH Aachen University, Aachen, Germany
[2]LIMSI CNRS, Spoken Language Processing Group, Paris, France
{wiesler, richard, schlueter, ney}@cs.rwth-aachen.de

## ABSTRACT

Log-linear models find a wide range of applications in pattern recognition. The training of log-linear models is a convex optimization problem. In this work, we compare the performance of stochastic and batch optimization algorithms. Stochastic algorithms are fast on large data sets but can not be parallelized well. In our experiments on a broadcast conversations recognition task, stochastic methods yield competitive results after only a short training period, but when spending enough computational resources for parallelization, batch algorithms are competitive with stochastic algorithms. We obtained slight improvements by using a stochastic second order algorithm. Our best log-linear model outperforms the maximum likelihood trained Gaussian mixture model baseline although being ten times smaller.

***Index Terms***— discriminative models, optimization, speech recognition

## 1. INTRODUCTION

Conventional speech recognition systems follow the generative statistical approach. Typically, hidden Markov models (HMMs) with Gaussian mixture models (GMMs) as emission models are used for modeling the joint probability of the spoken word sequence and the acoustic vector sequence. Such models can be trained efficiently with the expectation maximization (EM) algorithm. Their performance can be improved by a subsequent discriminative training, *e.g.* according to the minimum phone error (MPE) [1] criterion.

In recent years, the interest in *discriminative models* for speech recognition has greatly increased. Strong empirical results have been obtained with hierarchical discriminative models, *e.g.* [2]. Another line of research studies discriminative models with a flat structure [3, 4, 5, 6]. Our interest is in the use of log-linear models, which are attractive, because they are statistical models with a *convex* training criterion. The convexity allows for finding the *global optimum* of the training criterion. In our recent work, we showed that performance competitive to discriminatively trained GMMs can be obtained by using log-linear models [7].

A drawback of all discriminative approaches are the high computational costs required in training. Therefore, the efficiency of optimization algorithms is an important research topic. In general, optimization methods for machine learning can be subdivided into two categories: batch algorithms and stochastic algorithms. In batch algorithms, the statistics which are used for updating the model are computed on the full dataset. In stochastic algorithms, only a small random subset is used. Stochastic algorithms are very promising on large and redundant datasets. However, batch algorithms can be accelerated strongly by using second order information. This is in contrast to the most widely used stochastic optimization algorithm stochastic gradient descent (SGD). Further, batch algorithms can be parallelized straightforwardly. Stochastic algorithms are widely used for training hierarchical models. For convex optimization, typically batch algorithms are employed. However, in recent years, stochastic algorithms have gained a lot of attention for convex models [8], [9], [10], [11].

In this paper, we investigate whether stochastic algorithms are beneficial for optimizing log-linear models. Furthermore, we compare SGD with several stochastic algorithms that make use of second order information. In addition, we compare the numerical robustness of stochastic and batch algorithms. Experiments are performed on a challenging English broadcast conversation task.

## 2. MODEL AND TRAINING CRITERION

Log-linear models are used to model class-posterior probabilities. Let $X \subset \mathbb{R}^D$ denote the observation space and $\mathcal{C} = \{1, \ldots, C\}$ a set of classes. A log-linear model is of the form

$$p_\Lambda(c|x) = \frac{\exp(\sum_{d=1}^{D} \lambda_{c,d} x_d)}{\sum_{c' \in \mathcal{C}} \exp(\sum_{d=1}^{D} \lambda_{c',d} x_d)}, \qquad (1)$$

This work was partly realized under the Quaero Programme, funded by OSEO, French State agency for innovation. The research leading to these results has received funding from the European Union Seventh Framework Programme EU-Bridge (FP7/2007-2013) under grant agreement N287658. H. Ney was partially supported by a senior chair award from DIGITEO, a French research cluster in Ile-de-France.

where $\Lambda = (\lambda_1; \ldots; \lambda_C)^T \in \mathbb{R}^{C \times D}$ are the parameters of the model, $x \in X$ is an observation, and $c \in \mathcal{C}$ is a class. Note that log-linear models correspond to linear classifiers. Non-linear decision boundaries can be achieved by mapping observations into a higher-dimensional space. A generic approach is the use of polynomial features, as *e.g.* in [6].

The natural training criterion of log-linear models is the *penalized conditional maximum likelihood criterion*, which minimizes the objective function

$$\mathcal{F}(\Lambda) = -\frac{1}{N} \sum_{n=1}^{N} \log p_\Lambda(c_n|x_n) + \frac{\alpha}{2} \|\Lambda\|^2 \,. \qquad (2)$$

The last term is a regularization term with the *regularization constant* $\alpha \geq 0$. An important property of log-linear models is that their training according to the penalized maximum likelihood criterion is a convex optimization problem.

Given an HMM state alignment, log-linear models can be trained on frame-level, *i.e.,* the classes are HMM states and the the observations are acoustic features. Such log-linear models can be used in HMM speech recognizers via the *hybrid approach* [12].

## 3. OPTIMIZATION ALGORITHMS

We compare two different types of algorithms for convex optimization: batch algorithms and stochastic algorithms. In the following, we give a brief description of the optimization algorithms which we consider here.

### 3.1. LBFGS

The most widely used batch algorithm for log-linear models is LBFGS [13]. LBFGS builds up a model of the inverse Hessian matrix from a limited history of the previous gradients and iterates. There exists a well-founded convergence theory for LBFGS, see *e.g.* [14]. Note that LBFGS has a natural stepsize of $1.0$, which mostly gives a sufficient decrease in the objective function. Therefore, a line-search can be avoided in most iterations, because the objective function can be evaluated together with the gradient of the next iteration without additional costs.

### 3.2. Rprop

Rprop [15] is a batch algorithm well known in the field of neural networks. Rprop uses separate learning rates for all parameters, which are computed from the signs of the gradient. The use of separate learning rates per parameter corresponds to a diagonal second order model of the objective function. There exist slightly different modifications of the basic Rprop algorithm. In our experiments, we used the $\mathrm{iRprop}^+$ variant proposed in [16]. In order to guarantee convergence of Rprop, a line search has to be employed [17]. However, in practice this is not necessary, when a sufficiently small initial learning rate is chosen. Rprop has only few tuning parameters, is simple to implement, and shows good empirical results.

As all batch algorithms, Rprop and LBFGS can be parallelized straightforwardly by distributing the computation of the gradient (*data-parallelism*).

### 3.3. Stochastic Gradient Descent

For stochastic algorithms, the gradient is computed on a random mini-batch only. The most widely used stochastic algorithm is *stochastic gradient descent* (SGD). The update follows a simple iterative rule:

$$\Lambda_{t+1} = \Lambda_t - \eta_t \nabla \mathcal{F}(\Lambda_t, \mathcal{B}_t), \qquad (3)$$

where $\eta_t$ is the learning rate in step $t$, $\mathcal{B}_t$ is a random subset of the training data, and $\nabla \mathcal{F}(\Lambda, \mathcal{B})$ is the gradient of $\mathcal{F}$ at $\Lambda$ on $\mathcal{B}$. It can be shown (see *e.g.* [18]) that SGD converges almost surely towards the optimum, if the learning rates fulfill

$$\sum_{t=1}^{\infty} \eta_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty \,, \qquad (4)$$

and the samples are identically distributed and independent (iid). In our experiments, we use learning rates defined by

$$\eta_t = \frac{\tau}{\tau + t} \eta_0 \,, \qquad (5)$$

where $\eta_0$ is the initial learning rate, and $\tau$ controls how fast the learning rates decrease over time. Iid samples are simulated, by shuffling the dataset. This requires that all samples have to be loaded into memory. Note that this is not required for batch algorithms, because they do not depend on the order of the samples.

A major disadvantage of stochastic algorithms is that they can not be parallelized well. Data-Parallelism is only possible within one mini-batch, which is typically in the size of a few hundred to a few thousand samples.

In contrast to LBFGS and Rprop, SGD does not make use of any second order information. However, the advantage of SGD is that it frequently updates the model. This is in particular useful, when working with very large datasets. In a number of works, stochastic algorithms that make use of second order information have been proposed. In this work, we compare SGD with two stochastic second order algorithms that seemed most promising to us.

### 3.4. Online LBFGS

The first one is an online version of LBFGS, which has been proposed in [11]. As in LBFGS, an approximation to the inverse of the Hessian matrix is built up from differences of subsequent gradients and models. In online LBFGS (oLBFGS), the stochastic gradient is used instead of the exact gradient. Online LBFGS requires that the gradients that are used for computing the difference, are computed on the same mini-batch. This means that on each mini-batch, two gradient evaluations have to be performed instead of one as in SGD.

| | train | dev10 | eval10 | eval11 |
|---|---|---|---|---|
| Amount of data | 103h | 3.3h | 3.7h | 3.3h |
| WER ML | - | 25.5 | 25.1 | 32.2 |
| WER MPE | - | 24.0 | 24.0 | 30.6 |
| WER log-linear | - | 24.8 | 24.7 | 31.7 |

**Table 1**. Corpus statistics for the 2010 English Quaero corpus and word error rates (WER) of the ML and MPE baseline systems, and the log-linear system.

### 3.5. Stochastic Levenberg-Marquardt

The second algorithm is known as the *stochastic Levenberg-Marquardt algorithm* [19] (SLM) and is widely used for the optimization of convolutional neural networks. Analogously to the Levenberg-Marquardt algorithm, the update is

$$\Lambda_{t+1} = \Lambda_t - \eta_t B_t \nabla \mathcal{F}(\Lambda_t, \mathcal{B}_t) \ , \tag{6}$$

where $B_t$ is an approximation to the inverse of the Hessian matrix.

In SLM, only a diagonal approximation of the Hessian matrix is used. For log-linear models, the diagonal of the Hessian matrix on a mini-batch can be computed exactly with only small additional costs. In order to compensate for stochastic noise, the diagonal stochastic Hessian matrix is smoothed using the estimate of the previous iteration and interpolated with the identity matrix.

## 4. EXPERIMENTAL RESULTS

In this section, we describe experiments with log-linear models on the 2010 English Quaero broadcast conversations recognition corpus.

### 4.1. Experimental Setup

Our GHMM baseline system is a simplified version of our evaluation system [20]. In contrast to our evaluation system, the baseline system is only a single one-pass system without multi-layer perceptron (MLP) features. See Table 4.1 for corpus statistics and baseline results.

The baseline system uses MFCC features with vocal tract length normalization and a voicedness feature. Context is incorporated by concatenating features from a window of nine frames. The dimension of the resulting feature vector is reduced to 45 by means of an LDA. The GMM has a pooled, diagonal covariance matrix and models 4500 generalized triphones. An ML model is trained with the EM algorithm with a splitting procedure. The GMM has roughly one million densities. The ML model is used as initialization for an MPE training. The recognition lexicon consists of 150k words with multiple pronunciations. The language model is a smoothed 4-gram, trained on roughly three billion words.

The log-linear system uses the same baseline features and the same state tying as the GHMM system. We use polynomial features of second order for the log-linear model, which are computed from the baseline features. We applied a mean and variance normalization to the features in order to improve the condition of the optimization problem [21]. The dimension of the second order features is $1080$. The number of parameters of the log-linear model is $4.86$ million, and thus only about one tenth of the GMM models. The small size of the log-linear models is beneficial in applications where memory usage is an issue. Furthermore, the decoding speed directly depends on the model size.

For LBFGS, we chose a history size of $20$. For all experiments with stochastic algorithms, mini-batches of size $4500$ are used. We tested smaller mini-batch sizes as well, but the results were slightly worse. The parameters of the learning rate (5) were optimized using roughly one million frames. In our experiments, we observed that it is very important to use a smaller learning rate for the bias parameters than for the other parameters ($0.1$ times the standard learning rate in our experiments).

### 4.2. Comparison of Batch Algorithms with SGD

In our first set of experiments, we compared SGD with the batch algorithms. We distributed the computation of the batch gradient on one hundred CPUs. For SGD, we used multi-threading with eight threads.

Rprop takes about 38 iterations to reach the objective function value of SGD after only a single data sweep. However, in the late phase of convergence, SGD is quite slow. Interestingly, Rprop is roughly three times faster than the widely used LBFGS. The overall computation time of SGD is much lower than that of Rprop. In wall clock time, Rprop is competitive to SGD because of the better parallelizability. Which algorithm is preferable depends on the amount of resources that one is willing to spend.

An important issue for the practical application of batch and stochastic algorithms is their sensitivity to rounding errors. Surprisingly, this issue did not receive much attention in the comparison of stochastic and batch algorithms. Stochastic algorithms are inherently robust to rounding errors. Since their search direction is computed on a random sample, the sampling noise will typically be much higher than the effect of rounding errors. For batch algorithms, significant rounding errors occur in the accumulation of the gradient, because a summation over the complete dataset is performed. In practice, this means that for batch algorithms, double precision is required at least for the data structure that holds the accumulated gradient. When using single precision for Rprop, the training diverged after $25$ iterations, and only reached a WER of $56.1\%$. For stochastic algorithms, we did not observe any difference in performance when using only single precision.

### 4.3. Stochastic Second Order Algorithms

Since batch algorithms benefit strongly from incorporating second order information, it is promising to incorporate second order information into stochastic algorithms. We compared the performance of SGD with the oLBFGS and SLM. In our experiments, the investigated stochastic second order algorithms perform slightly better than SGD, see Table 2. In the first iterations, both, oLBFGS and SLM, perform worse than SGD. But in the vicinity of the solution, they converge faster than SGD, see Figure 1. Note that the second order algorithms require additional computational effort. In particular, oLBFGS requires two gradient evaluations per batch, hence doubling the computational demands. In contrast, the additional computational costs of SLM are low. We also tested SGD-QN [10], a stochastic second order algorithm which received a lot of attention in literature. But in our experiments, SGD-QN failed to converge, presumably because it does not deal well with small regularization constants.

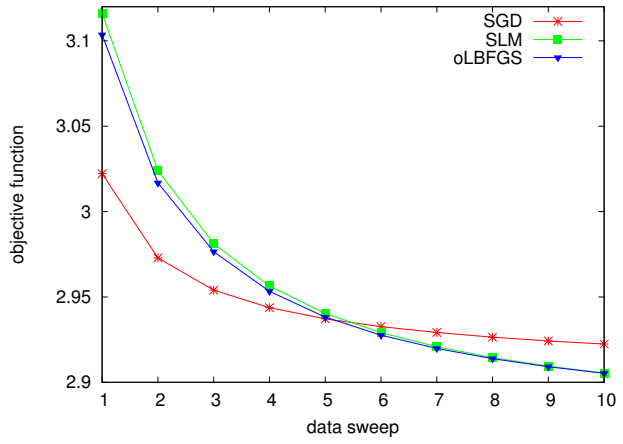


**Fig. 1**. Objective function of SGD, SLM, and oLBFGS.

### 4.4. Effect of Feature Normalization

The possible gains from using second order algorithms might be limited, because the mean and variance normalization of the features already captures an important part of the second order information of the convex objective function, *cf.* [21]. We examined this hypothesis by using the stochastic algorithms for training with unnormalized features. All algorithms performed worse without normalization of the features, see Table 2. While the performance of SGD and SLM degraded only by about one percent WER, oLBFGS was affected much stronger. This result is surprising, because the goal of incorporating second order information into stochastic algorithms, is to better cope with such ill-conditioned optimization problems than SGD.

| | data sweep | normalized | | unnormalized | |
|---|---|---|---|---|---|
| | | WER | $\mathcal{F}$ | WER | $\mathcal{F}$ |
| SGD | 5 | 25.3 | 2.937 | 26.6 | 3.050 |
| SGD | 10 | 25.2 | 2.922 | 26.2 | 3.017 |
| oLBFGS | 5 | 25.1 | 2.938 | 33.1 | 4.109 |
| oLBFGS | 10 | 24.9 | 2.905 | 32.3 | 4.055 |
| SLM | 5 | 25.1 | 2.940 | 25.7 | 3.020 |
| **SLM** | **10** | **24.8** | **2.905** | 25.8 | 3.007 |
| Rprop | 50 | 25.6 | 2.985 | 26.8 | 3.138 |
| Rprop | 100 | 25.3 | 2.935 | – | – |
| LBFGS | 50 | 27.8 | 3.004 | – | – |
| LBFGS | 100 | 26.1 | 3.194 | – | – |

**Table 2**. WERs on quaero-dev10 and objective function values on the training data for SGD, oLBFGS, SLM, Rprop, and LBFGS with and without feature normalization.

## 5. DISCUSSION

In literature, stochastic algorithms are typically seen as superior to batch algorithms for the optimization of discriminative models. This point of view can be supported by theoretical arguments [22]. However, these considerations do not take the possibility of parallel computation into account. Furthermore, batch algorithms require much less tuning than stochastic algorithms.

In our work, we empirically compared different stochastic and batch algorithms on a large-scale task. Our results show that batch algorithms are a valid choice when enough computational resources for parallelization are available. In particular, Rprop performs similar to stochastic gradient descent in terms of wall clock time, depending on the degree of parallelization. Interestingly, LBFGS, which is the most-widely used batch algorithm for the optimization of discriminative models, is much slower than Rprop.

Since batch algorithms make efficient use of second-order information, it is tempting to exploit second-order information in stochastic algorithms as well. We empirically compared two stochastic second-order algorithms which seemed most promising to us: oLBFGS [11], which previously has only been applied to small-scale tasks, and stochastic Levenberg-Marquardt [19], which is widely used for convolutional neural networks. The second-order algorithms continue improving at a point where SGD already begins to stagnate. Still, only the use of SLM is recommendable, because oLBFGS causes a large computational overhead.

Our best system, trained with SLM, yields a relative improvement of 3% WER over the ML trained GMM baseline. The MPE trained GMM performs better than the log-linear model, but at the cost of ten times more parameters.

## 6. REFERENCES


[1] D. Povey and P.C. Woodland, "Minimum phone error and I-smoothing for improved discriminative training," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Orlando, USA, 2002.

[2] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Interspeech*, 2011, pp. 437–440.

[3] A. Ragni and MJF Gales, "Structured discriminative models for noise robust continuous speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4788–4791.

[4] J. Morris and E. Fosler-Lussier, "Conditional random fields for integrating local discriminative classifiers," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 3, pp. 617–628, 2008.

[5] S. Watanabe, T. Hori, and A. Nakamura, "Large vocabulary continuous speech recognition using wfst-based linear classifier for structured data," in *Proc. Interspeech*, 2010, pp. 346–349.

[6] Simon Wiesler, Markus Nußbaum, Georg Heigold, Ralf Schlüter, and Hermann Ney, "Investigations on features for log-linear acoustic models in continuous speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2009 IEEE Workshop on*, 2009, pp. 52–57.

[7] Simon Wiesler, Ralf Schlüter, and Hermann Ney, "Accelerated batch learning of convex log-linear models for lvcsr," in *Interspeech*, 2009, pp. 52–57.

[8] SVN Vishwanathan, N.N. Schraudolph, M.W. Schmidt, and K.P. Murphy, "Accelerated training of conditional random fields with stochastic gradient methods," in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, USA, June 2006, pp. 969–976.

[9] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for svm," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 807–814.

[10] A. Bordes, L. Bottou, and P. Gallinari, "Sgd-qn: Careful quasi-newton stochastic gradient descent," *Journal of Machine Learning Research*, vol. 10, pp. 1737–1754, 2009.

[11] N. Schraudolph, J. Yu, and S. Günter, "A stochastic quasi-newton method for online convex optimization," in *International Conference on Artificial Intelligence and Statistics*, San Juan, Puerto Rico, March 2007, pp. 436–443.

[12] H. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*, vol. 247, Springer, 1994.

[13] D.C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, pp. 503–528, 1989.

[14] Jorge Nocedal and Stephen Wright, *Numerical Optimization*, Springer, 1999.

[15] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *Proceedings of the International Conference on Neural Networks*, March 1993, pp. 586–591.

[16] C. Igel and M. Hüsken, "Empirical evaluation of the improved rprop learning algorithms," *Neurocomputing*, vol. 50, pp. 105–123, 2003.

[17] A.D. Anastasiadis, G.D. Magoulas, and M.N. Vrahatis, "New globally convergent training scheme based on the resilient propagation algorithm," *Neurocomputing*, vol. 64, pp. 253–270, 2005.

[18] Léon Bottou, "Online learning and stochastic approximations," in *Online Learning in Neural Networks*, David Saad, Ed., pp. 9–43. Cambridge University Press, Cambridge, UK, 1998.

[19] Y. LeCun and L. Bottou, "Efficient backprop," in *Neural networks: Tricks of the trade*, G. Orr and K. Müller, Eds., pp. 546–546. Springer, New York, USA, 1998.

[20] M. Sundermeyer, M. Nußbaum-Thom, S. Wiesler, C. Plahl, A.E.D. Mousa, S. Hahn, D. Nolden, R. Schlüter, and H. Ney, "The rwth 2010 quaero asr evaluation system for english, french, and german," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Prague, Czech, May 2011, pp. 2212–2215.

[21] S. Wiesler, R. Schlüter, and H. Ney, "A convergence analysis of log-linear training and its application to speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 1–6.

[22] Léon Bottou and Olivier Bousquet, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems*, J.C. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., vol. 20, pp. 161–168. NIPS Foundation, 2008.