Simultaneous Perturbation Stochastic Approximation for Automatic Speech Recognition

Daniel Stein, Jochen Schwenninger, Michael Stadtschnitzer

Fraunhofer IAIS, Schloss Birlinghoven, Germany

name.surname@iais.fraunhofer.de

Abstract

While both the acoustic model and the language model in automatic speech recognition are typically well-trained on the target domain, the free parameters of the decoder itself are often set manually. In this paper, we investigate in how far a stochastic approximation algorithm can be employed to automatically determine the best parameters, especially if additional time-constraints are given on unknown machine architectures. We offer our findings on the German Difficult Speech Corpus, and present significant improvements over both the spontaneous and planned clean speech task.

Index Terms: spsa, free decoding parameters, real-time factor

10.21437/Interspeech.2013-181

1. Introduction

Both the optimization of the acoustic model and the language model in automatic speech recognition for large vocabularies are well-established tasks. Reducing the perplexity of the language model on a withheld development set, for example, is a common way to achieve lower word error rates (cf. [1]). The actual decoding process, however, also uses a large set of free parameters that should be adopted to the given task or domain. While some parameters directly weight the models, others affect the size of the search space, where it is even harder to foresee the effect on the hypothesis quality and on the expected decoding time.

In praxis, these parameters are often set empirically in a rather tedious task, which is even more complex as soon as a real-time factor (RTF) constraint has to be fulfilled. Moreover, they should be adopted to new domains, whenever the training material changes, or when more sophisticated decoding hardware is available that could possibly allow for either faster decoding or better decoding in the same amount of time.

In this paper, we employ Simultaneous Perturbation Stochastic Approximation (SPSA) [2] for optimizing the free decoding parameters and show that it leads to stable and fast results. Further, we show that by extending the loss function with a RTF penalty, arbitrary time constraints can be fulfilled while maintaining a reasonable output quality automatically. We offer our results on the German Difficult Speech Corpus (DiSCo) [3] corpus.

2. Related Work

While recently there has been work on optimizing the free decoding parameters using gradient decent by El Hannani and Hain [4], large-margin iterative linear programming by Mak and Ko [5] [6], or evolutional strategies by Kacur and Korosi [7], we aim at facilitating the optimization process by employing a fast approach and therefore enable this step for a wide range of applications.

In the field of machine translation (MT), the free parameters of recent decoders (e.g., [8, 9]) are typically estimated either with the Downhill Simplex Method [10] or with Och's Minimum Error Rate Training [11]. SPSA has been employed for MT as well and has been shown to be much faster in convergence than downhill simplex, while maintaining a comparable hypothesis quality [12].

3. Simultaneous Perturbation Stochastic Approximation

For the optimization of a tuple of free parameters θ, we employ the SPSA algorithm [2], which works as follows:

Let $\hat{\theta}_k$ denote the estimate for θ in the k-th iteration. Then, for a gain sequence denoted as a_k, and an estimate of the gradient at a certain position denoted as $\hat{g}_k(\cdot)$, the algorithm has the form

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}\left(\hat{\theta}_k\right) \tag{1}$$

In order to estimate $\hat{g}_k(\cdot)$, we perturbate each $\hat{\theta}_k$ with a vector of mutually independent, mean-zero random variables Δ_k, multiplied by a positive scalar c_k, to obtain two new parameter tuples:

$$\hat{\theta}_k^+ = \hat{\theta}_k + c_k \Delta_k \tag{2}$$

$$\hat{\theta}_k^- = \hat{\theta}_k - c_k \Delta_k \tag{3}$$

For a loss function $L(\cdot)$, we then estimate $\hat{g}\left(\hat{\theta}_k\right)$ as:

$$\hat{g}\left(\hat{\theta}_k\right) = \begin{bmatrix} \frac{L\left(\hat{\theta}_k^+\right) - L\left(\hat{\theta}_k^-\right)}{2c_k\Delta_{k1}} \\ \vdots \\ \frac{L\left(\hat{\theta}_k^+\right) - L\left(\hat{\theta}_k^-\right)}{2c_k\Delta_{kp}} \end{bmatrix} \tag{4}$$

We follow the implementation suggestions in [13] using a ± 1 Bernoulli distribution for Δ_k, and further set:

$$a_k = \frac{a}{(A+k+1)^\alpha} \quad \text{with } a = 2,\ A = 8,\ \alpha = 0.602$$

$$c_k = \frac{c}{(k+1)^\gamma} \quad \text{with } c = 0.25,\ \gamma = 0.101$$

Using these gain sequences a_k and c_k, SPSA normally converges in a similar number of iterations as the classical steepest decent in the Kiefer-Wolfowitz finite-difference stochastic approximation [14], but requires p times fewer measurements of

Table 1: Free parameters of the decoding process. Some parameters are given individually to the 1st pass or 2nd pass of the Julius decoder, and are marked with (2). Continuous parameters are marked by a trailing .0

name	start	min	max
(2) LM weight	10.0/10.0	0.0	20.0
(2) ins. penalty	-7.0/10.0	-20.0	20.0
(2) beam width	250/1 500	700/20	3000/1000
score envelope	80.0	50.0	150.0
stack size	10 000	500	20 000
#expanded hyp.	20 000	2 000	20 000
#sentence hyp.	10	5	1 000

the loss function. α and γ are set to the lowest values satisfying the theoretical conditions for convergence, leading to larger step sizes in the iteration process and therefore faster performance.

Since the measurements of $L(\theta)$ do not contain noise, we choose c to be a small positive number as proposed in [13]. Likewise, we set $A = 8$ to approximately 10% of the expected iterations and finally $a = 2$ so that the steps in the update have a reasonable size for the first iterations. During our experiments, we did not experience a high sensitivity of SPSA for any of its hyperparameters, only the required number of iterations to reach a stable result changes.

4. Setup

The basic architecture of our ASR system has been described in [15]. The training material for German is drawn from transcribed German broadcast news and talkshows, and has been recently extended to roughly 500 h clean speech. The language model data consists of roughly 11 M sentences. We use Julius [16] v 4.2.2 as decoding backend.

For developing, we use a corpus from German broadcast shows, which contains a mix of planned (i.e., read news) and spontaneous (i.e., talk shows) speech, for a total of 2 348 utterances (33 744 words).

For evaluation, we make use of clean speech segments of the DiSCo corpus as described in [3], and use "planned clean speech" (0:55h, 1364 utterances) as well as "spontaneous clean speech" (1:55h, 2861 utterances).

5. Experiments

For optimization, we chose to optimize both parameters that primarily affect the search space as well as those that affect the internal weighting/penalty of the underlying models. On the one hand, some settings might require more internal hypotheses to fully take effect, on the other hand, the search space directly affects the RTF which we also want to optimize. Table 1 lists the Julius parameters, the ranges that we allow for as well as the starting values for optimization. These baseline values are the result of a grid search with the baseline acoustic model. Internally, we map these ranges to $[-15 \cdots +15]$ for SPSA. If the parameters are integers, we store them as floats internally but truncate them for each loss function call.

5.1. WER optimization

First, we optimized the parameters on the word error rate (WER) percentage with a range of 0 to 100, i.e., the number of substitutions, insertions and deletion errors divided by the reference length.

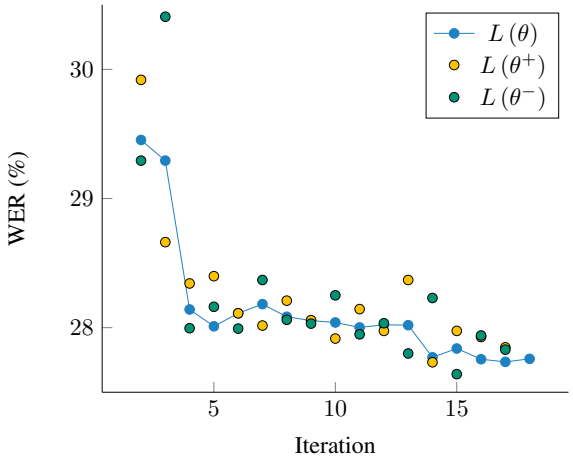


Figure 1: Example run of SPSA and its word error rate progression on the development corpus.

The results on the development set are shown in Figure 1. In total, the hypothesis quality improved by 1.9 WER absolute (6.4 rel.). In a second run (s. Table 2), the improvement was similar and converged after 10 iteration runs already. The results on the test sets are presented in Figure 2. It can be seen that the optimization generalizes nicely on both DiSCo corpora: 1.4% WER absolute improvements on the planned speech task, and 2.7% WER absolute improvement on the spontaneous speech task (s. Fig.2(a)), over a strong baseline surpassing the results given in the original corpus paper [3]. However, these improvements come with a rather high price in terms of RTF (s. Fig. 2(b)), especially for the second run. While for many settings this might not pose a problem, in time-crucial applications this is not desireable. Thus, in a second set of experiments, we try to take the RTF into account and optimize WER and RTF jointly.

5.2. Time-constrained WER optimization

In these sets of experiments, we penalize the loss function by a RTF dependent term μ:

$$L\left(\hat{\theta}_k\right) = \text{WER}\left(\hat{\theta}_k\right) + \mu\left(\hat{\theta}_k\right) \tag{5}$$

It soon became apparent that careful planning is needed in order to obtain the desired result. Intuitively, we penalized RTFs above a given threshold t exponentially. This turned out to deteriorate the parameters too much when the initial RTF was already substantially slower than this given threshold t. This was especially a problem for optimization on a slow machine, where the WER dropped 30% absolute due to a severe gradient misjudgement in the first iteration.

Using the delta of the actual RTF instead with

$$\mu\left(\hat{\theta}_k\right) = \begin{cases} \text{RTF}\left(\hat{\theta}_k\right) - t, & \text{for RTF}\left(\hat{\theta}_k\right) > t \\ 0, & \text{else} \end{cases} \tag{6}$$

lead to an equilibrium (s. Figure 3(a)), a trend that was reproducible on a second optimization run (s. Table 2). In general, the RTF appeared more stabilized, with no loss in WER visible.

In a final experiment, we penalized the RTF increasingly with each iteration:

$$\mu\left(\hat{\theta}_k\right) = \left(\text{RTF}\left(\hat{\theta}_k\right) - t\right) \cdot \tilde{k}\,, \text{for RTF}\left(\hat{\theta}_k\right) > t \tag{7}$$

Table 2: WER and RTF results on all corpora, for the SPSA iterations and their respective loss functions. Each optimization on a given loss function has been executed two times from scratch to check for convergence. The unconstrained runs use the WER directly as loss function, delta uses eq. 5 and increasing uses eq. 6

		dev			test planned			test spontaneous		
loss function	**#iter**	**WER**	**RTF** 1.6GHz	**ΔRTF**	**WER**	**RTF** 2.6GHz	**ΔRTF**	**WER**	**RTF** 2.6GHz	**ΔRTF**
baseline	0	29.6	5.3	1.00	24.0	4.6	1.00	31.1	4.0	1.00
unconstrained	18	27.7	7.0	1.32	22.8	5.4	1.17	28.4	5.9	1.48
unconstrained	18	27.7	7.3	1.38	22.6	6.1	1.33	28.4	6.1	1.53
delta	18	27.6	5.3	1.00	22.2	4.5	0.98	27.7	4.8	1.20
delta	18	27.6	5.1	0.96	22.5	4.2	0.91	27.9	4.4	1.10
increasing	14	32.5	3.0	0.57	26.1	2.2	0.48	31.9	2.3	0.58
	+28	31.6	2.9	0.55	25.3	2.5	0.54	30.0	2.6	0.65
increasing	12	31.2	3.0	0.57	25.5	2.2	0.48	31.0	2.1	0.53
	+28	33.6	2.9	0.55	27.5	2.3	0.50	32.1	2.4	0.60

with an increasing $\tilde{k} = k$ as long as a RTF threshold is not reached. For the first iteration where the RTF factor is equal to the threshold, $\tilde{k}$ is fixed in order to give the optimization the ability to converge, thus stabilizing the WER. In our experiments, we arbitrarily set the RTF threshold $t = 3$, which was reached in iteration 12 and 14, respectively. After this, the WER decreased in the first run another 0.9% absolute on the development set while maintaining the desired RTF (s. Figure 3(b)), with the result generalizing well to the unseen DiSCo planned and spontaneous test sets. In the second run, SPSA got stuck in a local optimum, leading to faster decoding but with lower quality.

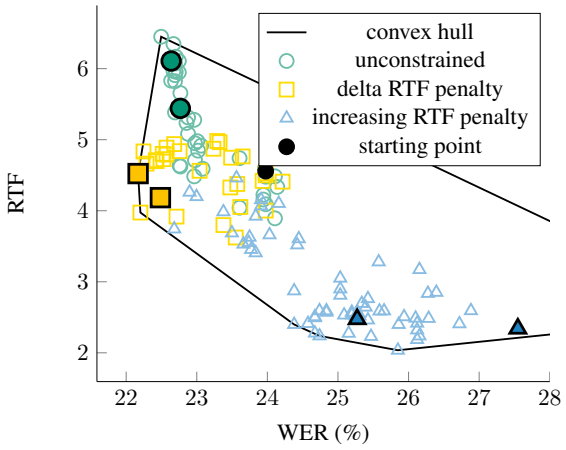
In order to see whether our optimization is a reasonable trade-of between RTF and WER, we collected all results from the iterations and computed their convex hull (Figure 4(a)). It can be seen that the final SPSA iteration for each optimization run is typically part of the convex hull or very near to its border. From our optimization runs, we could see no gain for the RTF-unconstrained loss function. A delta RTF penalized loss function could result in a configuration that performs better in terms of WER and is generally faster. If the RTF is penalized increasingly in each step, the WER rate is still within reasonable range for a much more comfortable RTF.

6. Conclusion

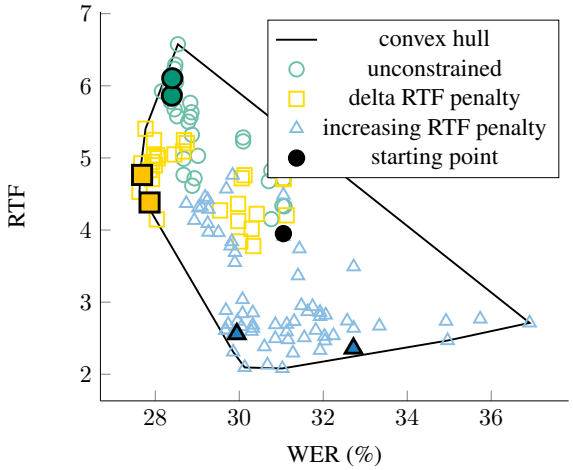
In this paper, we have shown that SPSA is an efficient means to optimize free parameters of an ASR decoder. In an unconstrained setting, the WER improves rapidly, but the RTF also increases in an undesirable way. By adding the RTF to the loss function, one is able to stabilize the increase in time requirements. Overall, we have achieved an improvement of 1.8 absolute WER on the DiSCo planned clean task and an improvement of 3.4 absolute WER on the DiSCo spontaneous task, over an already improved baseline, while still having a reasonable RTF. When a specific RTF is required for an application scenario, using the "increasing" penalty function for SPSA allows reaching reasonable performance, given this constraint. The risk for local optima seems higher in this setting, though, which is why we would recommend multiple optimization runs.

7. Acknowledgements

This work has been partly funded by the European Community's Seventh Framework Programme (FP7-ICT) under grant agreement n° 287911 LinkedTV and n° 269980 AXES.

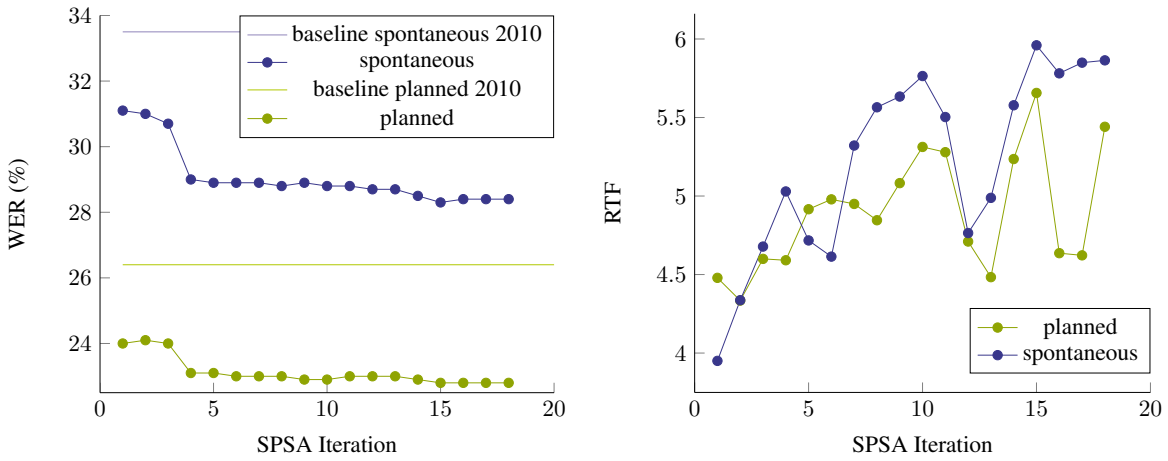


(a) Results for DiSCo "planned clean".



(b) Results for DiSCo "spontaneous clean".

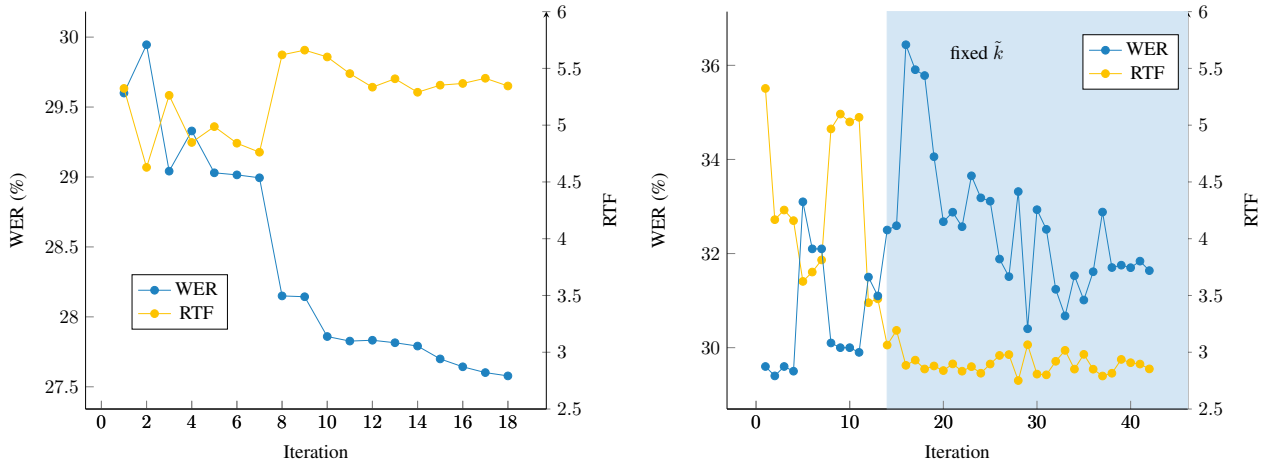
Figure 4: Scatter plot with all configurations, on the DiSCo test corpora. The final optimization iteration is marked by filled-out symbols.



(a) WER progression on the baseline given has been extracted from [3].

(b) RTF development on the DiSCo corpora "clean planned" and "clean spontaneous", for the first optimization run.

Figure 2: WER and RTF results on the DiSCo corpora "clean planned" and "clean spontaneous", using the unconstrained optimization criterion.



(a) Delta RTF penalty (Eqn. 6).

(b) Increasing RTF penalty (Eqn. 7).

Figure 3: Optimization runs on the development set, with different RTF-penalized loss functions.

8. References

[1] D. Klakow and J. Peters, "Testing the correlation of word error rate and perplexity," *Speech Communication*, vol. 38, no. 1–2, pp. 19–28, 2002.

[2] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37:3, Mar. 1992.

[3] D. Baum, D. Schneider, R. Bardeli, J. Schwenninger, B. Samlowski, T. Winkler, and J. Köhler, "DiSCo — A German Evaluation Corpus for Challenging Problems in the Broadcast Domain," in *Proc. Seventh conference on International Language Resources and Evaluation (LREC)*, Valletta, Malta, may 2010.

[4] A. El Hannani and T. Hain, "Automatic optimization of speech decoder parameters," *Signal Processing Letters, IEEE*, vol. 17, no. 1, pp. 95–98, 2010.

[5] B. Mak and T. Ko, "Automatic estimation of decoding parameters using large-margin iterative linear programming," in *Proc. Interspeech*, 2009, pp. 1219–1222.

[6] ——, "Min-max discriminative training of decoding parameters using iterative linear programming," in *Proc. Interspeech*, 2008, pp. 915–918.

[7] J. Kacur and J. Korosi, "An accuracy optimization of a dialog asr system utilizing evolutional strategies," in *Image and Signal Processing and Analysis, 2007. ISPA 2007. 5th International Symposium on*. IEEE, 2007, pp. 180–184.

[8] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens *et al.*, "Moses: Open source toolkit for statistical machine translation," in *Annual meeting-association for computational linguistics*, vol. 45, no. 2, 2007, p. 2.

[9] D. Vilar, D. Stein, M. Huck, and H. Ney, "Jane: an advanced freely available hierarchical machine translation toolkit," *Machine Translation*, vol. 26, no. 3, pp. 197–216, Sep. 2012. [Online]. Available: http://dx.doi.org/10.1007/s10590-011-9120-y

[10] J. Nelder and R. Mead, "The Downhill Simplex Method," *Computer Journal*, vol. 7, p. 308, 1965.

[11] F. Och, "Minimum error rate training in statistical machine translation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2003, pp. 160–167.

[12] P. Lambert and R. Banchs, "Tuning machine translation parameters with SPSA," in *Proc. IWSLT*, 2006, pp. 190–196.

[13] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34:3, Jul. 1998.

[14] J. Kiefer and J. Wolfowitz, "Stochastic estimation of a regesssion function," *Ann. Math. Stat.*, no. 23, pp. 462–466, 1952.

[15] D. Schneider, J. Schon, and S. Eickeler, "Towards Large Scale Vocabulary Independent Spoken Term Detection: Advances in the Fraunhofer IAIS Audiomining System," in *Proc. Association for Computing Machinery's Special Interest Group Information Retrieval (ACM SIGIR)*, Singapore, 2008.

[16] A. Lee, T. Kawahara, and K. Shikano, "Julius – an Open Source Real-Time Large Vocabulary Recognition Engine," in *Proceedings of Eurospeech*, Aalborg, Denmark, 2001, pp. 1691–1694.