AN ADAPTIVE GRADIENT-SEARCH BASED ALGORITHM FOR DISCRIMINATIVE TRAINING OF HMM'S

*Albino Nogueiras-Rodríguez** *José B. Mariño* *Enric Monte*

Universitat Politècnica de Catalunya. Barcelona, SPAIN.

{albino,canton,enric}@gps.tsc.upc.es

ABSTRACT

Although having revealed to be a very powerful tool in acoustic modelling, discriminative training presents a major drawback: the lack of a formulation guaranteeing convergence in no matter which initial conditions, such as the Baum-Welch algorithm in maximum likelihood training. For this reason, a gradient descent search is usually used in this kind of problem. Unfortunately, standard gradient descent algorithms rely heavily on the election of the learning rates. This dependence is specially cumbersome because it represents that, at each run of the discriminative training procedure, a search should be carried out over the parameters ruling the algorithm. In this paper we describe an adaptive procedure for determining the optimal value of the step size at each iteration. While the calculus and memory overhead of the algorithm is negligible, results show less dependence on the initial learning rate than standard gradient descent and, using the same idea in order to apply self-scaling, it clearly outperforms it.

1 INTRODUCTION

During the last years, discriminative training (**DT**) has proved to be a powerful tool in acoustic modeling for automatic speech recognition. Most discriminative training frameworks are based on the use of an auxiliary function that somehow characterises the behaviour of the system under real conditions. We shall refer to this function as *loss function*, $\mathcal{LF}$, although the results herein presented can be also applied to quality measures such as the *mutual information*. The objective of DT is to obtain the set of parameters (Λ^*) that minimises $\mathcal{LF}(\Lambda)$. As a multi-dimensional non-linear problem, DT presents several characteristics that make most classical algorithms either unaffordable or unreliable. For instance, they involve a great number of parameters—more than 150,000 in the examples considered in this paper. As a result, most work done so far makes use of any kind of gradient search relying on the fact that a *small enough* movement in the direction of the negative gradient always leads to an improvement in the loss function.

In this paper we propose the use of a simple adaptive gradient search algorithm (**AGS**) that provides a way to overcome most of the problems associated with the election of the step sizes, ε_t, in gradient descent search. We shall compare the results obtained using AGS with those obtained with standard gradient descent (**GD**) in the minimisation of the task independent confusibility of a phone HMM based continuous speech recognition system. The details of the system are explained in a companion paper [3].

*This paper was supported by the CICYT, Spanish government, under contract TIC95–0884–C04–02, and the CIRIT, Generalitat de Catalunya, through the CREL.

2 GRADIENT DESCENT IN DT

GD is one of the most widely used strategies in order to cope with the optimisation of loss functions in DT [1]. Being $\mathcal{LF}(\Lambda_t)$ the loss function for system Λ at time t, the following actualisation of the parameters of the system

$$\Lambda_{t+1} = \Lambda_t - \varepsilon_t \nabla \mathcal{LF}(\Lambda_t) \tag{1}$$

ensures that $\mathcal{LF}(\Lambda_{t+1}) \leq \mathcal{LF}(\Lambda_t)$, as long as $\mathcal{LF}(\Lambda)$ is continuous and ε_t is a positive quantity small enough to ensure that the first order approximation applies. This actualisation procedure leads to an iterative algorithm that, for continuous functions, is guaranteed to converge, at least, to a local minimum. In its simplest implementation, the step size, ε_t, is kept constant during the whole optimisation procedure. In the following, we shall refer to this method as *gradient descent* (**GD**).

The main drawback of gradient descent methods is that they rely heavily on the election of the step size sequence because the convergence to a solution depends on the form of the landscape of the function $\mathcal{LF}(\Lambda)$. If the values are chosen too big, the first order approximation will not apply and the system will probably not converge. If too small, convergence will be slow and the probability of being caught in a local minimum is higher. As a result, the step sizes themselves need to be optimised each time a discriminative training is scheduled. As the step size sequence is completely defined by the election of ε_0, the step sizes may be optimised via a simple line search over this value. Unfortunately, this optimisation may be extremely costly because, at each iteration of the line search over ε, a complete optimisation of the system parameters Λ must be carried out. The situation is specially serious if we consider that the optimal values of the step size series may depend on other parameters affecting either the system or the optimisation algorithm. Thus, whenever a change is done in any of these parameters, a new optimisation of the step sizes must be done in order to guarantee that the convergence conditions are consistently met. This is the case, for instance, if we want to evaluate the effect of changing the number of states of the HMM's or the number of hypotheses considered in DT.

2.1 Experimentation Using Gradient Descent

We have applied GD to the minimisation of the task independent confusibility on a phone based CSR system. 2608 utterances from the train male corpus of TIMIT were used to train phone HMM's. The parameters of the system were adapted every 400 utterances, i.e. some 6.5 times per complete cycle through the whole training corpus. Five different step sizes, spaced by a factor of 10, were tried in order to show the margin of values for which the algorithm converges, as well as what happens if this value is multiplied or divided by 10. Figure 1 plots the evolution of the phone error rate in the phone recognition of the test male corpus of TIMIT during 10 complete cycles of GD. There are two remarkable things:

- All five ε's improve the baseline result. Nevertheless, over-shooting—an excessive step size—is noticeable for the biggest one, and under-shooting for the smallest.
- Even for the two mid values, those that present best convergence, the difference in performance is almost one point in the error rate.

3 ADAPTIVE GRADIENT SEARCH ALGORITHM

One way to overcome the difficulties in the election of the step sizes is using the method of *steepest descent* (**SD**). Steepest descent is one of the most widely used methods for minimising non-linear functions of several variables. It works by applying Equation 1 with a succession of step sizes ε_t defined by

$$\varepsilon_t = \arg\min_{0\leq\varepsilon<\infty} \mathcal{LF}(\Lambda_t - \varepsilon\nabla\mathcal{LF}(\Lambda_t)). \qquad (2)$$

This expression not only avoids the ambiguity in the step size values but also presents better convergence properties than simple GD. The problem now is how to evaluate the ε_t that satisfies Equation 2. Ideally, a line search over ε should be carried out at each iteration of Equation 1, but this line search would be as expensive, or more, as the evaluation of the gradient, so the computational cost will even be higher than just optimising ε_0 in GD. Nevertheless, and unlike other more sophisticated methods as Davidon-Fletcher-Powell's one, it has been frequently observed that SD is not radically affected by inaccuracy in the line search. This is so because SD is strictly a gradient descent method. This means that, at each time t, the direction of update of the parameters of the system, Λ, is the gradient of $\mathcal{LF}(\Lambda)$, which, in turn, represents the direction over which maximum improvement in the loss function is achieved with the minimum perturbation of the parameters. Even if Equation 2 is not verified, a closed enough approximation will lead to a solution that will be, at least, as good as if we took a small enough step to ensure that the first order approximation holds. Yet, it is arguable if an exact solution to Equation 2 is needed. Although it would guarantee good convergence properties for pure quadratic functions, we do know that this will not be the case for the kind of functions being optimised in DT. So accurateness in the line search will be voided by the errors in the quadratic assumption.

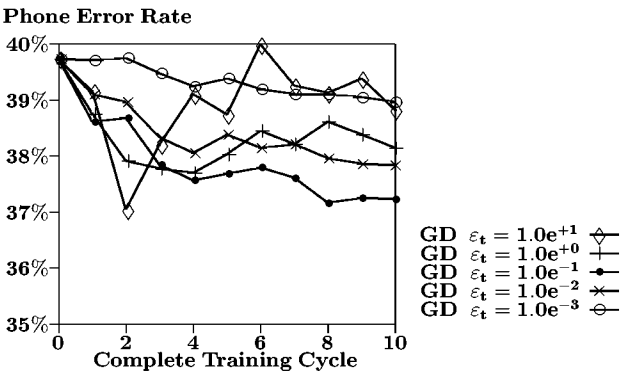

Figure 1: Phone error rate evolution using GD

We will use this flexibility in the line search of SD in order to develop a very simple method of estimating convenient values for ε_t. Lets suppose that $\mathcal{LF}(\Lambda)$ is a pure quadratic form with positive definite Hessian, Q, i.e.:

$$\mathcal{LF}(\Lambda) = \Lambda^T Q\Lambda - \Lambda^T b. \qquad (3)$$

In this case, the line search carried over the direction of the gradient becomes a simple parabolic optimisation problem because the projection of a quadratic form over any direction is a parabola. One convenient way of performing line search over a parabola is the so-called *method of false position*. It is based on the fact that the derivative of a parabola is a straight line. As the minimum of the parabola corresponds to the point where this line equals zero, two different values of the derivative are enough to determine where this happens.

Lets consider now, with the above assumptions, how we can get to the optimal value Λ_t^* that optimises $\mathcal{LF}(\Lambda)$ along the line $\Lambda_t - \mu\nabla\mathcal{LF}(\Lambda_t)$, where μ is any real non-zero value. We define $\mathbf{g}_t = -\nabla\mathcal{LF}(\Lambda_t)$, $\Lambda'_t = \Lambda_t + \mu\mathbf{g}_t$ and $\mathbf{g}'_t = -\nabla\mathcal{LF}(\Lambda'_t)$. At Λ'_t, the partial derivative of the loss function in the direction of $\mathbf{g}_t$ is the projection of the gradient at this point, $\mathbf{g}'_t$, over $\mathbf{g}_t/\|\mathbf{g}_t\|$. Then

$$\begin{aligned}\Lambda_t^* &= \Lambda_t + \frac{\mathbf{g}_t^T\mathbf{g}_t}{\mathbf{g}'^T_t\mathbf{g}_t - \mathbf{g}_t^T\mathbf{g}_t}(\Lambda'_t - \Lambda_t)\\ &= \Lambda_t - \mu\frac{\mathbf{g}_t^T\mathbf{g}_t}{\mathbf{g}_t^T\mathbf{g}_t - \mathbf{g}'^T_t\mathbf{g}_t}\mathbf{g}_t\\ &= \Lambda_t - \varepsilon_t^*\mathbf{g}_t \qquad (4)\end{aligned}$$

Where $\varepsilon_t^* = \mu\frac{\mathbf{g}_t^T\mathbf{g}_t}{\mathbf{g}_t^T\mathbf{g}_t - \mathbf{g}'^T_t\mathbf{g}_t}$ is the optimal step size that leads to Λ_t^* applying Equation 1. Two things are remarkable about Equation 4:

- It leads to Λ_t^* with independence of μ, so one posible—and probably convenient—choice for μ is ε_{t-1}.
- Due to the parabolic approximation, the value obtained for ε_t^* is also optimal if Equation 1 is applied from Λ'_t substituting the gradient at this point, $\mathbf{g}'_t$, by its projection over $\mathbf{g}_t$.

These two properties enable a short cut in the SD procedure: we move from Λ_t to Λ_{t+1} with the value of ε_t that would have led to Λ^*_{t-1} if this step size were only applied to the component of the gradient at Λ_t parallel to the gradient at Λ_{t-1} Instead of applying it to this only component—as would require strict SD—we apply it to the whole gradient. We cannot ensure that this value will verify Equation 2 at any time t, but it would be equivalent to getting to Λ^*_{t-1} —thus optimising the line search for time $t-1$—and using this same step size as an estimate of the optimal one in the direction of the orthogonal component of the gradient—in a way performing an additional line-search [1]. Any way, the algorithm ensures that any movement done during the iterative search will be optimised in the following iteration. As a result, the step size taken at time t, ε_t, is the step size taken in the previous iteration multiplied by an actualisation term $\hat{\varepsilon}_t$:

$$\left.\begin{array}{lcl} \mathbf{g}_t & = & -\nabla\mathcal{LF}(\Lambda_t) \\ \hat{\varepsilon}_t & = & \frac{\mathbf{g}_{t-1}^T\mathbf{g}_{t-1}}{\mathbf{g}_{t-1}^T\mathbf{g}_{t-1}-\mathbf{g}_t^T\mathbf{g}_{t-1}} \\ \varepsilon_t & = & \hat{\varepsilon}_t\varepsilon_{t-1} \\ \Lambda_{t+1} & = & \Lambda_t - \varepsilon_t\mathbf{g}_t \end{array}\right\} \text{for } t>0 \qquad (5)$$

3.1 Positive Definiteness of the Hessian

Equation 5 is only valid for loss functions with positive definite Hessian. If this is not the case, and in general it will not, then the second derivative along any given direction may be negative. In this situation, actually a line search, the negativeness is not as serious as in the case of multi dimensional optimisation. Yet, it is easily detected—the value $\mathbf{g}_t^T\mathbf{g}_t/\mathbf{g}_t\mathbf{g}_t$ will be greater than one—, and may be easily corrected. A negative value of the second derivative simply indicates the presence of a maximum, instead of a minimum, in the second order approximation. So its value has little significance in a minimisation problem. As a matter of fact, the problems with the value of $\mathbf{g}_t^T\mathbf{g}_{t-1}/\mathbf{g}_{t-1}\mathbf{g}_{t-1}$ do start when it gets close to one. This is because, for values greater than zero, the method of false position becomes an extrapolation instead of an interpolation. As it gets close to one, $\hat{\varepsilon}_{t-1}$ grows until reaching infinity at this value, and becoming negative for values above it. Nevertheless, the direction of the gradient is always the maximum benefit one. Moreover, assuming the parabolic approximation, a positive value of $\mathbf{g}_t^T\mathbf{g}_{t-1}/\mathbf{g}_{t-1}\mathbf{g}_{t-1}$ means that moving along the direction of the gradient, $\mathbf{g}_{t-1}$, not only is beneficial at time $t-1$, but will also be at time t. Thus it seams reasonable to think that a greater value of the step size at time $t-1$ could have been enough to get to that same point in one iteration less. This leads to an iterative procedure similar to Equation 5, but with a different formulation for the evaluation of ε_t when $\frac{\mathbf{g}_t^T\mathbf{g}_{t-1}}{\mathbf{g}_{t-1}^T\mathbf{g}_{t-1}} \geq a$ being $0 \geq a \geq 1$ —we use a value $a = 0.75$ in our experiments—.

$$\hat{\varepsilon}_t = 1 + (1-a)^{-1}\frac{\mathbf{g}_t^T\mathbf{g}_{t-1}}{\mathbf{g}_{t-1}^T\mathbf{g}_{t-1}} \quad \text{for} \quad \frac{\mathbf{g}_t^T\mathbf{g}_{t-1}}{\mathbf{g}_{t-1}^T\mathbf{g}_{t-1}} \geq a \qquad (6)$$

[1] Notice that $\Lambda_t = \Lambda'_{t-1}$ for all t

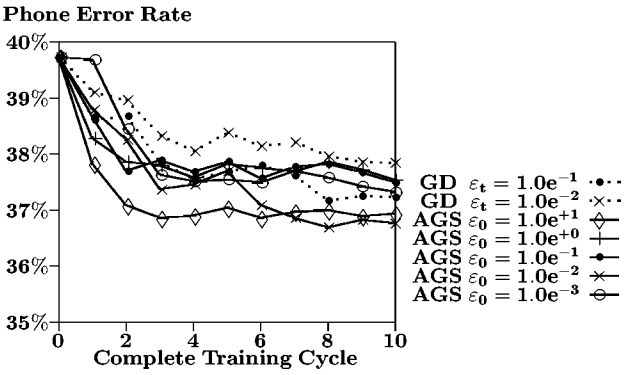

Figure 2: Phone error rate evolution using AGS

3.2 Experimentation Using Adaptive Gradient Search

We have applied this algorithm to the same task of Section 2.1. Figure 2 plots the evolution of the error rate during ten iterations of the method. The same five initial ε_0's used in GD are used now. The evolution of the error rate for the two best runs of GD is also plotted—with dotted lines—for comparison purposes.

It stands out that, for all five initial step sizes, performance achieved using AGS is very similar to that achieved with the best run of GD, being, in all cases, superior to the second best run. Moreover, the variance in the results is now much smaller than in the case of GD. Yet, the variance in the results is now similar to the confidence margin—with a total number of 4284 phones, the 95% confidence margin in the error rate is some $\pm 0.7\%$—so we believe that the final discrepancy in the results is more dependent on their estimation than on the magnitude of the initial step-size. It is significant that, now, the evolution in the error rate is very similar in all cases. This result confirms that AGS is much less sensitive to the initial step size chosen. As a matter of fact, the algorithm neutralises the effects of this election after the first few iterations—employed in removing both under and

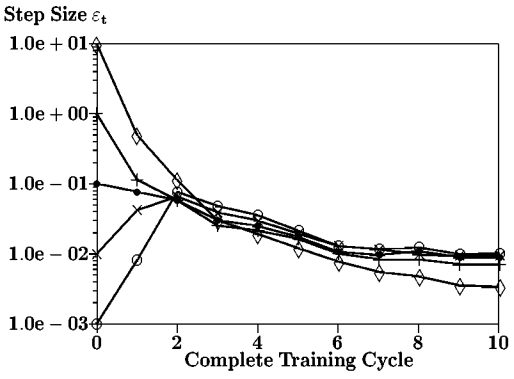

Figure 3: Evolution of ε_t in AGS

over-shooting—, and then ε_t evolves similarly in all cases. This behaviour is depicted in Figure 3.

3.3 Adaptive Interpretation

The algorithm in Equation 6 has an adaptive interpretation. Following this interpretation, the movement done at time t, $d_t = -\varepsilon_t \mathbf{g}_t$ is decomposed into two components d_t^P and d_t^N, being the first parallel to the direction of $\mathbf{g}_{t-1}$, and the second its conjugate. The parallel one, d_t^P, can be seen as a correction to the movement done at time $t-1$. If d_{t-1} was too small in magnitude, d_t^P will grow in the same direction as d_{t-1} and Equation 6 will make bigger the step size. If it is too big d_t^P will decrease in the direction of d_{t-1} and the step size will be made smaller. The value of ε_t is then adapted at each iteration until $\hat{\varepsilon}_t$ equals one, in which case successive movements in the gradient descent procedure are taken in orthogonal directions. Convergence of the algorithm is ensured at time t whenever the Hessian of $\mathcal{LF}$ is semi positive definite inside the region $\{\Lambda_\mu : \Lambda_{t-1} + \mu \nabla \Lambda_{t-1} \quad | \quad 0 \leq \mu \leq \varepsilon_{t-1}\}$. This region of convergence extends the one of GD to all convex regions.

3.4 Self-Scaling of the Variables

SD is very sensitive to the eigenvalue structure of the Hessian of the loss function [2]. Specifically, convergence of SD degrades fast when the ratio between the highest and the lowest eigenvalues grows. On the contrary, if this ratio is kept close to the unity, convergence is ensured in just one step of SD for quadratic forms. One way to modify the structure of eigenvalues of the Hessian, while keeping the location of the minima unaltered, is substituting $\nabla \mathcal{LF}(\Lambda)$ with $\mathbf{U} \nabla \mathcal{LF}(\Lambda)$, where $\mathbf{U}$ is a suitable definite positive matrix. If $\mathbf{U}$ is chosen to be diagonal, this procedure is equivalent to a linear scaling of the variables, and can be seen as using a different step-size for each variable.

In the case that the Hessian of the loss function is diagonal by blocks, the projections of the gradient over each of the hyperplanes that define the different blocks will bring a set of directions that will be both decoupled and orthogonal. Each of the hyperplanes will define a sub-space where the values of the gradient do not depend on the values of the variables lying in any other hyperplane. The decouplement between the different projections of the gradient enables the so-called *conjugate directions* method to be implemented in a concurrent way: for each of the N conjugate components in which the gradient at time t is decomposed, $\mathbf{g}_t^n$, we estimate the value of the different ε_t^n using Equation 6 independently on each of them, and actualise the parameters of Λ with the composite movement $\Lambda_{t+1} = \Lambda_t - \sum_n \varepsilon_t^n \mathbf{g}_t^n$. Although the Hessian will not be, in general, diagonal by blocks, it seems reasonable to consider that parameters corresponding to a same kind of parameter—transition or emission probabilities, mean or variance of the codebook, etc.—, and a same unit will be more tightly coupled than parameters corresponding to different kinds or units. In this way, the problem of determining a convenient scaling can be reduced to applying AGS independently on each kind of parameter and each unit.

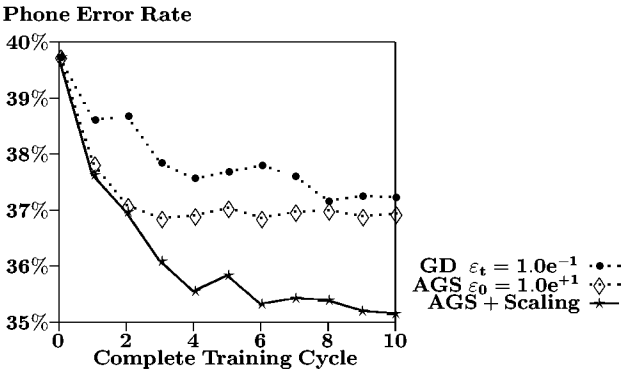

Figure 4: Error rate of AGS with self-scaling

In our implementation of this algorithm, we perform AGS independently for each kind of parameter at odd iterations, and independently for each unit at even ones, in this way we arrive to a different value of ε_t^n for each kind of parameter and unit, while relaxing the diagonal by blocks assumption at each iteration. Even in the case that the different hyperplanes are coupled, the convergence of the method will not be too much affected because AGS will force convergence in all of the hyperplanes independently. Yet, as neither over or under-shooting will be allowed for any of the directions chosen, this procedure will be more controlled than just estimating one common step size.

Figure 4 shows the evolution of the performance of the algorithm when this scaling is performed, as well as the best result achieved with GD and AGS without scaling. It stands out that applying of self-scaling not only improves convergence but also leads to much better results than either of the two other methods. This result was also previously stated using the *gradient probabilistic descent* algorithm [1].

ACKNOWLEDGMENTS

The authors wish to thank Prof. Juan Fernández Rubio for his valuable suggestions to this work and Mr. Christian Pomar Berry for his determinant collaboration in the preparation of this paper.

REFERENCES

[1] B.-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Trans. on Signal Proc.*, pages 3043–3054, December 1992.

[2] D. G. Luenberger. *Linear and Nonlinear Programming (2nd. Edition).* Addison–Wesley, 1984.

[3] A. Nogueiras and J.B. Mariño. Task adaptation of sub-lexical unit models using the minimum confusibility criterion on task independent databases. In *Proc. of ICLSP'98*, 1998.