Second-Order Methods in Boltzmann Learning: An Application to Speechreading

Morten With Pedersen*
Section for Digital Signal Processing
Department of Mathematical Modelling
Technical University of Denmark B321
DK-2800 Lyngby, DENMARK
mwp@imm.dtu.dk

David G. Stork
The MLP Group
Ricoh Silicon Valley
2882 Sand Hill Road Suite 115
Menlo Park, CA 94025-7022 USA
stork@crc.ricoh.com

Abstract

We introduce second-order methods for training and pruning of general Boltzmann networks trained with cross-entropy error. In particular, we derive the second derivatives for the entropic cost function. We illustrate pruning on *Boltzmann zippers*, applied to real-world data — a speechreading (lipreading) problem.

1 INTRODUCTION

Second-order methods for training feed-forward neural networks, such as Gauss-Newton, Levenberg-Marquardt and quasi-Newton algorithms [2], can greatly reduce learning time compared to first-order methods such as backpropagation. Boltzmann networks are among the slowest networks to train, and thus it is natural that we apply second-order methods to them as well. While there are several differences between traditional feed-forward networks and Boltzmann networks, at a sufficiently abstract level these networks are similar enough that the approaches developed for traditional networks can be taken over to Boltzmann networks, as we shall see.

Second-order methods have another important use in neural networks: pruning. There is an enormous body of simulation work demonstrating the value of architecture optimization for networks for pattern classification, and this has properly led to great interest in both theoretical foundations and in new algorithms. There are two basic viewpoints toward this issue: regularization (or penalty based) and sensitivity based. According to the viewpoint of regularization, one seeks to impose

*Corresponding author. Category: Algorithms and Architectures. Presentation: Oral.

some desired property in the final solution, for instance smoothness. Thus in weight decay one penalizes large weights and therefore favors smoother decision boundaries. According to the viewpoint of sensitivity, one seeks to eliminate those parameters (e.g., weights) that have the smallest effect on the training error, thereby restricting the model without severely penalizing the training error. Recent methods (special cases of the Wald statistic) eliminate weights that are predicted to have the least effect on the training error [1, 3]. In this work we introduce these methods (OBD and OBS) in the context of Boltzmann networks.

2 BOLTZMANN NETWORKS

Though typically slower and a bit more difficult to train than feedforward neural networks, Boltzmann networks nevertheless have some desirable properties: natural handling of missing data (during both training and application), pattern completion, and superior avoidance of local energy minima during training. Boltzmann networks are stochastic networks with both visible and hidden units (cf., [4] for an introduction and the notation we use here). We let the subscript α denote the states of the visible units and β the states of the hidden units. The superscript $+$ denotes iterating the network with the visible units clamped to a desired pattern, and $-$ denotes the visible units running freely, or unclamped. The energy function for the Boltzmann network is usually defined as

$$E \;=\; -\frac{1}{2}\sum_{ij} w_{ij} s_i s_j, \tag{1}$$

where w_{ij} is the (bi-directional) weight connecting units i and j, and s_i is the (binary) state of unit i. At *thermal equilibrium* the probability of finding the units in a given state configuration $\alpha\beta$ when the visible units are unclamped is given by the Boltzmann distribution

$$P^-_{\alpha\beta} \;=\; \frac{1}{Z} e^{-E_{\alpha\beta}}, \tag{2}$$

where Z is the normalizing partition function and $E_{\alpha\beta}$ is the energy (dependent on the weights) when the visible units are in states α and the hidden units are in states β; for clarity in the following, we have incorporated the *temperature* into the energy term. Thus, the probability P^-_α of finding the visible units in joint states α is found by summing over the possible hidden unit configurations β.

When training Boltzmann networks we want the probabilities of the freely running network P^-_α to match those of the environment/training examples P^+_α. As a measure of the difference between the two probability distributions we use the *Kullback-Leibler* measure, or relative entropy, as our cost function:

$$H(\mathbf{w}) \;=\; \sum_\alpha P^+_\alpha \ln \frac{P^+_\alpha}{P^-_\alpha} \;=\; -\sum_\alpha P^+_\alpha \ln P^-_\alpha + \mathit{const}, \tag{3}$$

where *const* is a constant determined solely by the environment, and is hence independent of the weights $\mathbf{w}$. When training using gradient descent we need the derivatives of $H(\mathbf{w})$ with respect to the bi-directional weights w_{ij} connecting units i and j:

$$\begin{aligned} \frac{\partial H(\mathbf{w})}{\partial w_{ij}} &= -\sum_\alpha P^+_\alpha \frac{\partial \ln P^-_\alpha}{\partial w_{ij}} = -\sum_\alpha P^+_\alpha \left\langle -\frac{\partial E_{\alpha\beta}}{\partial w_{ij}} \right\rangle^+_\alpha + \left\langle -\frac{\partial E_{\alpha\beta}}{\partial w_{ij}} \right\rangle^- \\ &= \; < s_i s_j >^- - < s_i s_j >^+, \end{aligned} \tag{4}$$

where $< \cdots >^+_\alpha$ is the mean value given that the visible units are clamped in states α, and $< \cdots >^-$ is the mean when all units are free running.

3 SECOND-ORDER METHODS

The first derivatives lead to the traditional first-order training methods [4]. However, for faster training methods and for pruning algorithms we need the second derivatives of the entropic cost. These second derivatives are calculated as:

$$\begin{aligned}\frac{\partial^2 H(\mathbf{w})}{\partial w_{ij}\partial w_{pq}} &= -\left[\sum_\alpha \frac{P_\alpha^+}{P_\alpha^-}\cdot\frac{\partial^2 P_\alpha^-}{\partial w_{ij}\partial w_{pq}} - \sum_\alpha \frac{\partial P_\alpha^-}{\partial w_{ij}}\cdot\frac{P_\alpha^+}{\left(P_\alpha^-\right)^2}\cdot\frac{\partial P_\alpha^-}{\partial w_{pq}}\right] \\ &\approx \sum_\alpha \frac{\partial \ln P_\alpha^-}{\partial w_{ij}}\cdot P_\alpha^+\cdot\frac{\partial \ln P_\alpha^-}{\partial w_{pq}}. \end{aligned} \tag{5}$$

The approximation in Eq. 5 is equivalent to that made in *Fisher's method of scoring* [7] and also corresponds to the Gauss-Newton approximation to the Hessian of a quadratic cost in which the term with second derivatives is ignored. As is the case for the quadratic cost function, the approximation becomes exact in the limit of infinitely many examples, provided the network is not underparametrized. In that limit, the parameters that minimize the entropic cost function will converge towards a set of *optimal* weights $\mathbf{w}^*$ for which $P_\alpha^- = P_\alpha^+$, $\forall\alpha$. For these weights the term in Eq. 5 involving second derivatives of the unclamped probabilities P_α^- reads

$$\sum_\alpha \frac{P_\alpha^+}{P_\alpha^-}\cdot\frac{\partial^2 P_\alpha^-}{\partial w_{ij}\partial w_{pq}} = \sum_\alpha \frac{\partial^2 P_\alpha^-}{\partial w_{ij}\partial w_{pq}} = \frac{\partial^2}{\partial w_{ij}\partial w_{pq}}\left(\sum_\alpha P_\alpha^-\right) = 0, \tag{6}$$

where in the first step we used the fact that $P_\alpha^+ = P_\alpha^-$ at $\mathbf{w} = \mathbf{w}^*$, and in the last step that $\sum_\alpha P_\alpha^- = 1$. Thus for weights sufficiently "close" to the optimal, $\mathbf{w}^*$, the term in Eq. 5 involving second derivatives will be "small" and the rest involves terms of a form calculated from Eq. 4:

$$\frac{\partial \ln P_\alpha^-}{\partial w_{ij}} = \left\langle -\frac{\partial E_{\alpha\beta}}{\partial w_{ij}}\right\rangle_\alpha^+ - \left\langle -\frac{\partial E_{\alpha\beta}}{\partial w_{ij}}\right\rangle^- = < s_i s_j >_\alpha^+ - < s_i s_j >^- . \tag{7}$$

Thus we obtain the simplified form of the second derivatives:

$$\begin{aligned}\frac{\partial^2 H(\mathbf{w})}{\partial w_{ij}\partial w_{pq}} &\approx \sum_\alpha \frac{\partial \ln P_\alpha^-}{\partial w_{ij}}\cdot P_\alpha^+\cdot\frac{\partial \ln P_\alpha^-}{\partial w_{pq}} \qquad (8)\\ &= \sum_\alpha P_\alpha^+ < s_i s_j >_\alpha^+ < s_p s_q >_\alpha^+ - < s_p s_q >^- \sum_\alpha P_\alpha^+ < s_i s_j >_\alpha^+ \\ &\quad - < s_i s_j >^- \sum_\alpha P_\alpha^+ < s_p s_q >_\alpha^+ + < s_i s_j >^- < s_p s_q >^- . \end{aligned}$$

We note that the approximation to the second derivatives is positive definite which is an advantage when using the second derivatives for second-order training. The approximation involves only terms already computed when calculating the gradient, and thus implementation is straightforward, requiring little computation beyond that needed for computing the gradient.

3.1 Learning: The damped Gauss-Newton method

Learning in Boltzmann networks can be sped up if a more efficient optimization technique than gradient descent is used. Here we suggest the damped Gauss-Newton method,[2] in which the direction of weight changes in iteration k is computed as

$$\Delta\mathbf{w}_k = -\eta[H''(\mathbf{w}_k)]^{-1}H'(\mathbf{w}_k). \tag{9}$$

This corresponds to the direction towards the minimum of a second-order expansion of the entropic cost function around the current iteration point. The term "damped" refers to the use of a line search such as simple bisection in order to determine the step size η. The line search makes the method globally convergent and is simple if the Boltzmann network has a structure that allows for exact computation of likelihoods.

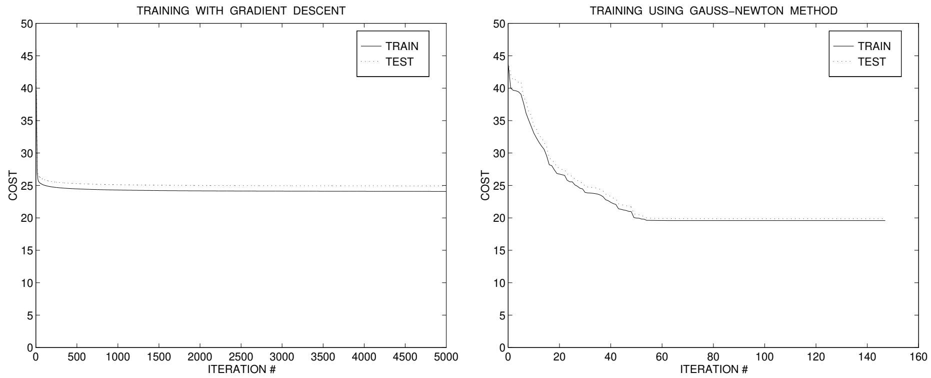

Figure 1: Learning in a Boltzmann chain (see text). Left panel: Training using gradient descent. Right panel: Training using the Gauss-Newton method.

Figure 1 compares the performance between gradient descent and the damped Gauss-Newton method when training a Boltzmann chain [6] from the same initial weights. The problem was identification of a left-to-right HMM having three observation states and three hidden states from fixed-length observation sequences generated by the HMM. In this example we observed a 30% increase in computation time per iteration when using the Gauss-Newton method; the increase is however highly justified by a much more rapid convergence. In ten runs, only once did gradient descent reach the level of error obtained using the Gauss-Newton method within 5000 iterations; the Gauss-Newton method only needed 100–200 iterations.

Regularization is crucial for the application of second-order training methods to especially Boltzmann chains as the Hessian for the entropic cost function is inherently rank-deficient for this model type. This is so since addition of an arbitrary constant to all the weights in these networks will not influence the Boltzmann distribution Eq. 2 and therefore leaves the entropic cost unchanged. Augmentation by a simple quadratic weight decay term, $\lambda/2\,\mathbf{w}^{\mathrm{T}}\mathbf{w}$, solves this problem.

3.2 Pruning: Saliency based methods

Saliency based pruning algorithms developed for feed-forward nets include Optimal Brain Damage (OBD) [1] and Optimal Brain Surgeon (OBS) [3]. Both use second-order expansions of the error to estimate the importance, or saliency, of the parameters if these are reset to zero. The rationale behind both methods is that if we remove the least salient weights according to training error, we gracefully relieve the danger of overfitting, and thereby simplify the network and (often) improve generalization.

Having derived the second derivatives for the entropic cost function allows for the application of OBD and OBS to Boltzmann networks. For simplicity, we here focus on the analog of OBD, for which the saliency for parameter j is then computed as

$$\delta H_j^{\mathrm{OBD}} = \left(\lambda + \frac{1}{2} \frac{\partial^2 H(\mathbf{w})}{\partial w_j^2} \right) w_j^2 \tag{10}$$

when working from a cost function augmented by a quadratic weight decay term. We note that the pruning methods only requires information already provided by the learning algorithms. Both pruning as well as the weight decay will bias the parameters towards the value zero and thus bias the Boltzmann distribution towards a smooth uniform distribution.

4 SPEECHREADING PROBLEM

Speechreading — audio-visual speech recognition, or more colloquially "lipreading" — refers to the use of *both* visual and audio signals for speech recognition [8]. There is ample evidence, from a number of independent research groups, that the incorporation of visual information can improve recognition accuracy, especially in noisy environments.[8]

4.1 Sensory integration

The central novel problem in speechreading is sensory integration — how to best integrate (and learn) information from the acoustic and visual channels. There are three principal methods: early, intermediate and late integration. In early integration, the (preprocessed) audio and video features are effectively concatenated to provide an expanded feature vector, which is then recognized. In late integration, there are, effectively, two separate classifiers: one for the video information, one for the audio information. The probability estimates of the categories, provided by the two channels, are then pooled for overall classification. Intermediate integration is a somewhat vague term which depends upon the fundamental recognition engine in question (HMMs, neural nets, Dynamic Time Warping, ...), but has come to mean any integration scheme "between" early and late integration.

Any speech recognition architecture must be insensitive to the variable rate of natural speech, and for this reason Hidden Markov Models have enjoyed great popularity in the speech community. However, both early and late integration using HMMs seem to be suboptimal for the automatic speechreading problem. In early integration, the acoustic and visual features are concatenated and input to a single HMM. The problem is that, for the vast majority of words, the acoustic signal requires more hidden states than the visual, and thus any *single* HMM represents a poor compromise. In late integration, one has two separate HMMs, with the appropriate, different number of hidden states, and pool their probability estimates for the final classification. However, this precludes the learning of low-level bi-modal features, for instance based on both the instantaneous visual shape of the mouth and the sound.

In short, it appears that an HMM-based architecture supporting some form of intermediate integration is needed — one where the number of hidden states can differ between the two channels, while also allowing cross-modal features to be learned. Recent work has shown how two linked Boltzmann chains [6] (which have been dubbed "Boltzmann zippers" because of their architecture) have this ability. It is for this reason we explored their use on a real-world speechreading problem.

4.2 Experiment

We constructed an isolated-word speechreading system using Boltzmann zippers for classifying the three nonsense utterances `/asklee/`, `/asklaa/` and `/askluu/` (Fig. 2). These were chosen because they illustrate well a vexing problem in speech recognition: co-articulation. The sound of the `/s/` phoneme differs dramatically in

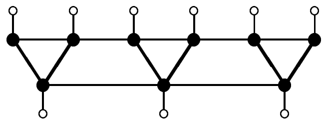

Figure 2: Topology of a Boltzmann zipper. The white circles represent groups of visible units/states; at the top for the "fast" (in speechreading, the acoustic channel), at the bottom the "slow" (visual) channel. Dark circles represent groups of hidden units/states. This architecture is formally equivalent to two HMMs, fully cross-connected by trainable weights.

these utterances, due to the influence of lip rounding associated with the different vowels coming "later". We hypothesized that, for the linked-HMM models for these utterances, interactions between the "later" visual states would have significant interactions with the "early" acoustic ones. Indeed, an analysis of the trained zipper weights confirmed this hypothesis.

Repetitions of each word were recorded (including video information) from a single talker in a low-noise environment. Each of the three sets of utterances was divided into two sets: 15 utterances for training and five for testing. The audio part of each utterance consisted of a sequence of 48 feature vectors, the video information for each utterance consisted of a sequence of 24 feature vectors. Thus, the video information was obtained at half the speed of the audio information. Each video feature vector was a collection of visual parameters representing height and width of the mouth opening, thickness of the upper lip etc. The feature vectors were converted into discrete observations using the LBG algorithm.

A Boltzmann zipper having five 'fast' hidden unit states and three 'slow' hidden unit states was trained for each of the utterances using second-order methods combined with OBD pruning. In the left panel of Fig. 3 is shown the evolution of the entropic cost for the model trained on the `/asklee/` utterances as parameters were pruned away. Initially both the training and test error decreased as parameters were removed. This was caused by the Gauss-Newton algorithm being able to locate lower-lying local minima during retraining of the reduced model. As the zipper was increasingly restricted by pruning the training error began to increase again, as is common in pruning experiments. Around 60 parameters left in the model the test error suddenly dropped to the lowest level obtained. This is explained by the

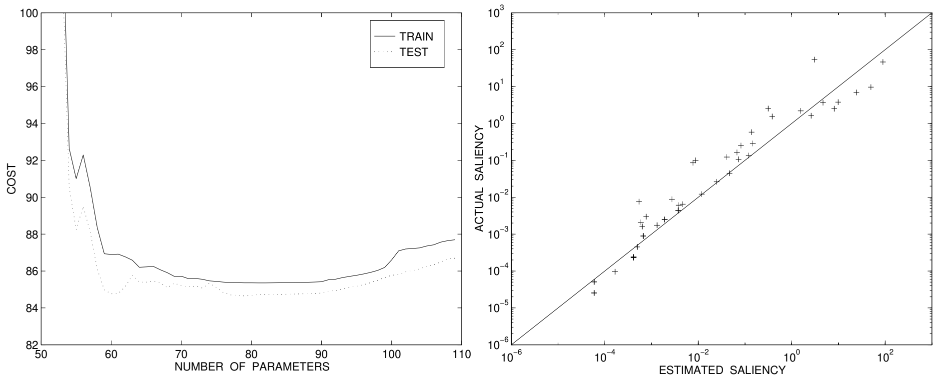

Figure 3: Left: Evolution of the relative entropy cost function for the model trained on the `/asklee/` data as the parameters of a Boltzmann zipper are pruned to the zero. Right: Estimated versus actual saliencies for the fully connected Boltzmann zipper.

zipper being sufficiently gracefully restricted and thereby reducing the possibilities of overfitting the training sequences.

In the right panel of Fig. 3 we illustrate the quality of the saliency estimates. For the fully connected zipper, we plot the estimated saliency versus the actual saliency computed by setting the weight to zero and calculating the resulting change in training error. We note that especially for low saliency weights the estimates are approximately equal to the actual saliencies. Note that rank ordering of parameters according to estimated saliency is consistent with rank ordering according to actual saliency; and thus the correct weight is pruned at each step.

Similar results were obtained for the Boltzmann zippers optimized for the `/asklaa/` and `/askluu/` utterances. The three optimal zippers were then combined into a speechreading system, which was able to correctly classify the 15 test utterances.

5 CONCLUSION AND FUTURE WORK

We have derived the second derivatives of the entropic cost function for Boltzmann networks and have applied these for training using a damped Gauss-Newton method as well as well as pruning by Optimal Brain Damage. We demonstrated the methods on Boltzmann zippers (linked HMMs) on a limited, isolated-word speechreading task. The improved speed and ease of implementation should remove some of the criticisms of these methods, and lead to greater use of this powerful architecture. Our results suggest that second-order methods, and Boltzmann zippers themselves, can be employed on larger, more complex pattern recognition problems as well.

Acknowledgments

This work was completed partly during a visit to the Ricoh California Research Center, supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT). The authors would like to thank Lars Kai Hansen and Greg Wolff for support, and Lawrence Saul for valuable discussions and for making available his Boltzmann zipper code.

References

[1] Y. L. Cun, J. S. Denker and S. A. Solla, "Optimal Brain Damage," in D. Touretzky, ed., **Advances in Neural Information Processing Systems**, Morgan Kaufmann, San Mateo, CA, 1990, vol. 2, pp. 598–605.

[2] P. E. Gill, W. Murray and M. H. Wright, **Practical Optimization**, London: Academic Press, 1981.

[3] B. Hassibi and D. G. Stork, "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon," in S. J. Hanson, J. D. Cowan and C. L. Giles, eds., **Advances in Neural Information Processing Systems**, Morgan Kaufmann, San Mateo, CA, 1993, vol. 5, pp. 164–171.

[4] S. Haykin, **Neural Networks. A Comprehensive Foundation**, New York: Macmillan College Publishing, 1994.

[5] D. J. C. MacKay, "Equivalence of Linear Boltzmann Chains and Hidden Markov Models," **Neural Computation**, vol. 8, no. 1, pp. 178–181, 1996.

[6] L. K. Saul and M. I. Jordan, "Boltzmann Chains and Hidden Markov Models," in G. Tesauro, D. Touretzky and T. Leen, eds., **Advances in Neural Information Processing Systems**, The MIT Press, 1995, vol. 7, pp. 435–442.

[7] G. A. F. Seber and C. J. Wild, **Nonlinear Regression**, New York: John Wiley & Sons, 1989.

[8] D. G. Stork and M. E. Hennecke, eds., **Speechreading by Humans and Machines**, New York: Springer-Verlag, 1996.