Weight Quantization for Multi-layer Perceptrons Using Soft Weight Sharing

Fatih Köksal[1], Ethem Alpaydın[1], and Günhan Dündar[2]

[1] Department of Computer Engineering
[2] Department of Electrical and Electronics Engineering
Boğaziçi University, Istanbul Turkey

Abstract. We propose a novel approach for quantizing the weights of a multi-layer perceptron (MLP) for efficient VLSI implementation. Our approach uses *soft weight sharing*, previously proposed for improved generalization and considers the weights not as constant numbers but as random variables drawn from a Gaussian mixture distribution; which includes as its special cases k-means clustering and uniform quantization. This approach couples the training of weights for reduced error with their quantization. Simulations on synthetic and real regression and classification data sets compare various quantization schemes and demonstrate the advantage of the coupled training of distribution parameters.

1 Introduction

Since VLSI circuits must be produced in large amounts for economy of scale, it is necessary to keep the storage capacity as low as possible to come up with cheaper products. In an artificial neural network, the parameters are the connection weights and if they can be stored using fewer bits, storage need will be reduced and we gain from memory.

In this work, we try to find a good quantization scheme to achieve a reasonable compression ratio for parameters of the network, without significantly degrading accuracy. Our proposed method finds a method to partition the weights of a neural network into a number of *clusters* so that only one value is used for one cluster of weights. Thus, the actual memory which stores the real-numbered values will be small and weights will be pointers to this memory. Then, the weights in a cluster will point to the same location in the real memory. For example, given an MLP with 10,000 weights that can be grouped into 32 clusters, for each weight only five bits are used.

An analogy is the color map. To get 16 million colors, one requires 24 bits for each pixel. Graphics adapters have color maps, e.g., of size 256, where each entry is 24 bits and is one of 16 million colors. Then using eight bits for each pixel, we can index one entry in the color map. So the storage requirement for an image is one third. Although this means that an image can contain only 256 of the 16 million possible colors, if quantization [1] is done well, there will not be a degradation of quality. Our aim is to do a similar quantization of weights of a large MLP.

G. Dorffner, H. Bischof, and K. Hornik (Eds.): ICANN 2001, LNCS 2130, pp. 211–216, 2001.

The benefit is obvious for digital storage in reducing the memory size. For analog storage of weights, capacitors are perhaps the most popular method, and in such circuits, the area of the capacitors generally dominates the total area. Thus, it is very important to reduce the number of weights.

There is a great amount of research on the quantization and efficient hardware implementation of neural networks (specifically MLP). The aim is to get an MLP with weights which are represented by less number of bits, which consumes less memory at the expense of loss of precision; example studies are given in [2,3,4,5,6]. In our approach which is novel, we do not reduce the precision; the operations are still with full precision. We decrease storage by clustering of the weights.

The organization of the paper is as follows: In Section 2, we discuss the possible methods for quantization and divide them into two groups: after-training methods and during-training methods. Section 3 contains experimental design and the results of these methods, and in Section 4, we conclude and discuss future work.

2 Soft Weight Sharing

We can generally classify the applicable methods for weight quantization into two. The simplest method would be training the neural network normally and then directly applying quantization to the weights of the trained network. We call these *after-training* methods. However, the application of quantization without considering the effect of quantization leads to large error and therefore combining quantization and training is a better alternative which we call *during-training* methods [5].

The methodology we use is *soft weight sharing* [7] where it is assumed that the weights of an MLP are not constants but are random variables drawn from a mixture of Gaussians

$$p(w) = \sum_{j=1}^{M} \alpha_j \phi_j(w) \tag{1}$$

where w is a weight, α_j are the mixing coefficients (prior probabilities), and the component densities $\phi_j(w)$ are Gaussians, i.e., of the form $\phi_j(w) \sim \mathcal{N}(\mu_j, \sigma_j^2)$. The main reason for choosing this type of distribution for weights is its generality and analytical simplicity. There are three types of parameters in the mixture distribution, namely, prior probabilities, α_j, means, μ_j, and variances, σ_j. Assuming that the weights are independent, the likelihood of the sample of weights is given by

$$L = \prod_{i=1}^{W} p(w_i) \tag{2}$$

In *after-training*, once the training of the MLP is complete, Expectation-Maximization (EM) method can be used to determine the parameters [8]. It is known that when all priors and variances are equal, the well-known quantization method *k-means clustering* is equivalent to EM. One can even view *uniform quantization*

Table 1. The test set error values (average±standard deviation) on the regression dataset for several quantization levels. The unquantized MLP has an error of 4.10±0.06.

	1 Bit	2 Bits	3 Bits
Uniform	245.33±137.60	104.87±60.33	40.64±23.01
K-means	173.34±87.59	57.60±20.97	16.85±13.71
SWS	18.85±2.27	9.99±4.91	4.61±1.05

in this framework where additional to all priors and variances being equal, means are equally spaced and fixed.

In *during-training* methods, to couple the training of weights with their quantization, we take the negative log likelihood converting it to an error function to be minimized

$$\Omega = -\sum_i \ln \left(\sum_{j=1}^{M} \alpha_j \phi_j(w_i) \right) \tag{3}$$

This then is added as a penalty term to the usual error function (mean square error in regression and cross-entropy in classification)

$$\overline{E} = E + \upsilon \Omega \tag{4}$$

to get the augmented error function $\overline{E}$ which is then minimized, e.g., using gradient-descent, to learn both the weights w_i, and also the parameters of their distribution, i.e., μ_j, σ_j, and α_j. We do not give the update equations here due to lack of space but they can be found in [7].

3 Experimental Results

We have used one synthetic regression dataset for visualization and two real datasets for speech phoneme and handwritten digit recognition. All datasets are divided as training and test sets. For all problems, we first train MLPs and find the optimum number of hidden units and the number of parameters, then they are quantized with different number of bits. We have run each model ten times, starting gradient-descent from different random initial weights and report the average and standard deviation of error on the test set.

The MLP of regression problem has one input, one output and four hidden units thus using 13 weights. Thus without any quantization, we need four bits. We try quantizing with two, four, and eight clusters (Gaussians) corresponding to one, two, and three bits. An example graph of regression function as quantized by soft weight sharing is given in Figure 1. In this figure, the effect of quantization on the regression function is easily observed.

The means and the variances of error with the three methods are given in Table 1. k-means is an after-training method and works better than uniform quantization. Still better though is the result with soft weight sharing (SWS), which is a *during-training* method and clearly demonstrates the advantage of coupled training.

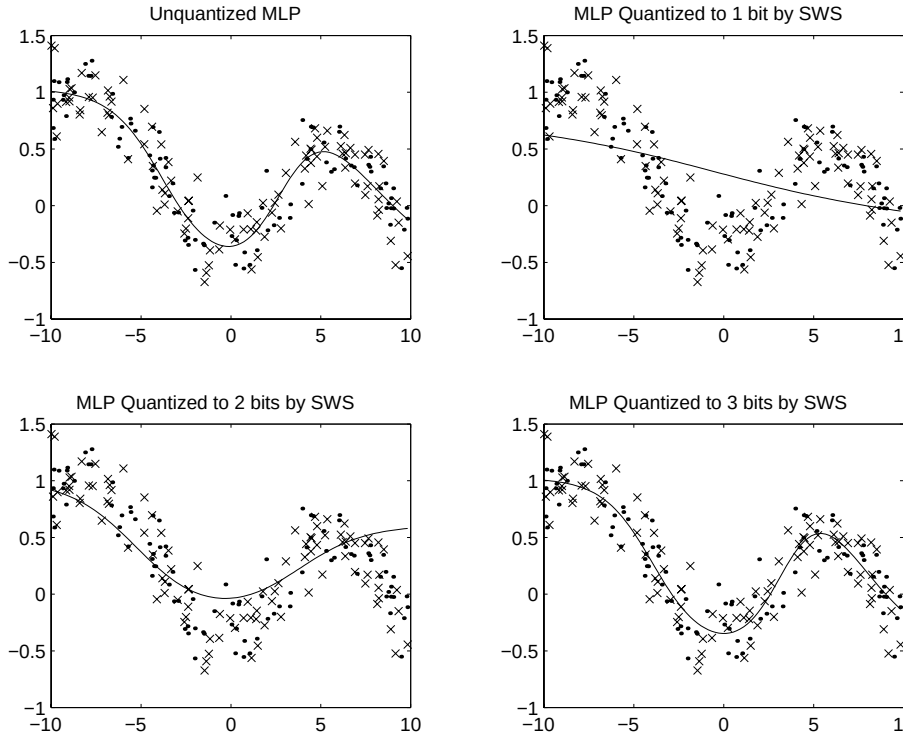

Fig. 1. The graphs of regression function; unquantized and quantized by soft weight sharing (SWS) with one, two, and three bits.

The speech phoneme is represented as a 112 dimensional input which contains 200 samples from six distinct classes (six speech phonemes) and is equally divided into two as training and test sets. The MLP has ten hidden units with 1196 parameters and thus unquantized, we need 10 bits. We have used one, two, three, four and five bits for our performance evaluation. Table 2 reports the simulation results. In Figure 2, we draw the scatter of weights and the fitted probability distribution using soft weight sharing.

Note that the probability distribution of weights is clearly a Gaussian distribution centered near zero. Although, there are some papers (e.g. [3]) which claim and assume that the weights of an MLP are distributed uniformly, we face a zero-mean normal distribution. This must not be surprising because there are more than 1000 parameters and they are initialized randomly to be close to zero and a large majority are not much updated later on. Note that if we were using a pruning strategy, the connections belonging to a cluster which is centered very close to zero would be the ones to be pruned. The significant ones are those which are distant from zero, since they have large error gradient and are updated much larger than the other parameters.

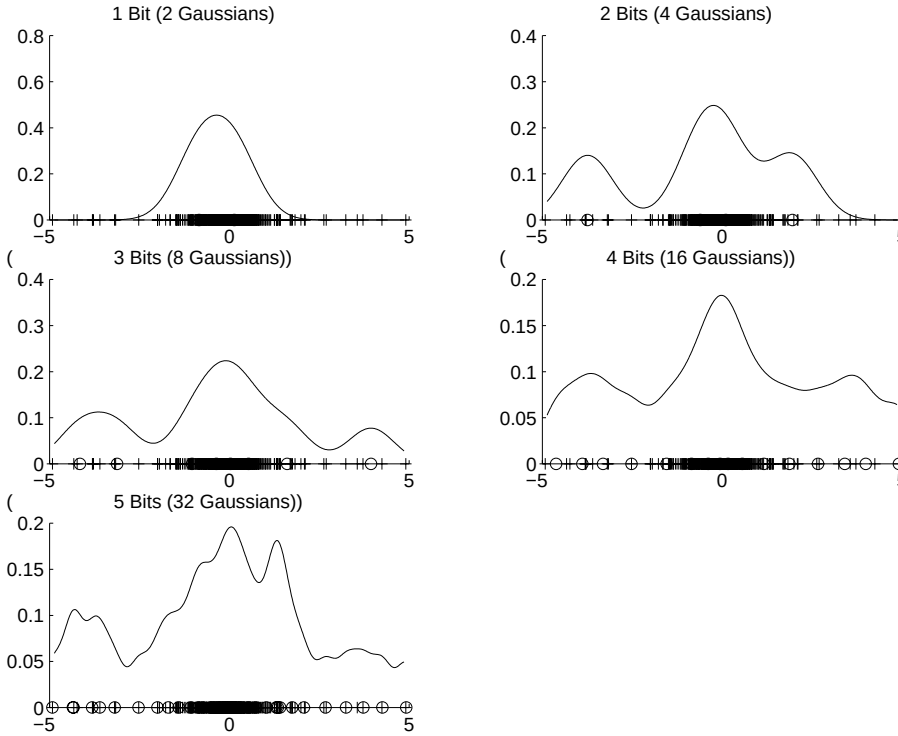

Fig. 2. The distribution of weights of the MLP used for speech phoneme recognition and the fitted Gaussian mixture probability distribution by soft weight sharing. Vertical bars are the weights; circles are the centers of the Gaussians.

The handwritten digit dataset contains 1,200 samples each of which is a 16×16 bitmap. The MLP has ten hidden units and we have 2,680 parameters for quantization. The distribution of weight values after the training phase of MLP is again a normal distribution like with the speech data set. Table 3 contains the quantization results.

We see in both classification datasets that when the number of bits is large, even uniform quantization is good; the advantage of soft weight sharing, i.e., coupled training, becomes apparent with small number of clusters.

4 Conclusions

We propose to use soft weight sharing, previously proposed for improved generalization, for quantizing the weights of an MLP for efficient VLSI implementation and compare it with the previously proposed methods of uniform quantization and k-means. Our results indicate that soft weight sharing, because it couples the training of the MLP with quantization, leads to more accurate networks at the same level of quantization. Once quantization is done, the results also indi-

Table 2. On the speech phoneme recognition problem, average±standard deviation of number of misclassifications out of 600 on the test set are given. For comparison, with the unquantized MLP using 10 bits, the misclassification error is 40.70±3.79.

	1 Bit	2 Bits	3 Bits	4 Bits	5 Bits
Uniform	409.29±80.36	376.50±77.40	219.89±55.35	104.19±33.69	55.50±8.96
K-means	365.10±82.93	248.00±69.31	123.40±48.74	51.59±8.66	45.50±5.93
SWS	387.50±74.10	209.39±37.55	89.00±45.40	52.00±9.85	44.79±5.92

Table 3. On the handwritten digit recognition problem, average±standard deviation of number of misclassifications out of 600 of the MLP on the test set are given. For comparison, with the unquantized MLP, the misclassification error is 23.50±3.58.

	1 Bit	2 Bits	3 Bits	4 Bits	5 Bits
Uniform	526.70±21.75	512.70±42.37	441.70±70.21	95.19±37.08	29.29±5.67
K-means	448.39±114.14	263.29±75.21	58.79±27.43	27.70±5.38	24.50±2.80
SWS	397.20±139.41	244.19±53.66	64.19±37.24	30.70±7.28	27.90±6.47

cate the *saliency* of weights which can be used further to prune the unnecessary connections; we leave this as future work.

Acknowledgment

This work is supported by Grant 00A0101D from Boğaziçi University Research Funds.

References

1. Gersho, A. and R. Gray, *Vector Quantization and Signal Compression* Norwell, MA:Kluwer, 1992.
2. Choi, J. Y. and C. H. Choi, "Sensitivity Analysis of Multilayer Perceptron with Differentiable Activation Functions," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 101–107, 1992.
3. Xie, Y. and M. A. Jabri, "Analysis of the Effects of Quantization in Multi-Layer Neural Networks Using a Statistical Model," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 334–338, 1992.
4. Skaue, S., T. Kohda, H. Yamamato, S. Maruno, and Y. Shimeki, "Reduction of Required Precision Bits for Back Propagation Applied to Pattern Recognition," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 270–275, 1993.
5. Dündar, G. and K. Rose, "The Effects of Quantization on Multi Layer Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 6, pp. 1446–1451, 1995.
6. Anguita, D., S. Ridella and S. Rovetta, "Worst Case Analysis of Weight Inaccuracy Effects in Multilayer Perceptrons," *IEEE Transactions on Neural Networks*, Vol. 10, pp. 415–418, 1999.
7. Nowlan, S. J. and G. E. Hinton, "Simplifying Neural Networks by Soft Weight Sharing," *Neural Computation*, Vol. 4, pp. 473–493, 1992.
8. Alpaydın, E. "Soft Vector Quantization and the EM Algorithm," *Neural Networks*, Vol. 11, pp. 467–477, 1998.