A Simple Randomized Quantization Algorithm for Neural Network Pattern Classifiers

Jihoon Yang & Vasant Honavar*
Artificial Intelligence Research Group
Department of Computer Science
Iowa State University
Ames, IA 50011. U.S.A.

Abstract

This paper explores some algorithms for automatic quantization of real-valued datasets using thermometer codes for pattern classification applications. Experimental results indicate that a relatively simple randomized thermometer code generation technique can result in quantized datasets that when used to train simple perceptrons, can yield generalization on test data that is substantially better than that obtained with their unquantized counterparts.

1 Introduction

Artificial neural networks offer a particularly attractive framework for the design of pattern classification and inductive knowledge acquisition systems for a number of reasons including their potential for parallelism and fault tolerance.

A single threshold logic unit (TLU), also known as *perceptron*, is a simple neural network that can be trained to classify a set of input patterns into one of two classes. A TLU is an elementary processing unit that computes a function of the weighted sum of its inputs. Assuming that the patterns are drawn from an N-dimensional Euclidean space, the output O^p, of a TLU with weight vector $\mathbf{W}$, in response to a pattern $\mathbf{X}^p$, is a bipolar hardlimiting function of $\mathbf{W} \cdot \mathbf{X}^p$, i.e. $O^p = 1$ if $\mathbf{W} \cdot \mathbf{X}^p > 0$ and 0 otherwise. Such a TLU or threshold neuron implements a $(N-1)$-dimensional hyperplane given by $\mathbf{W} \cdot \mathbf{X} = 0$ which partitions the N-dimensional Euclidean pattern space defined by the coordinates $x_1 \cdots x_N$ into two regions (or two classes). Given a set of *examples* $S = S_+ \cup S_-$ where $S_+ = \{(\mathbf{X}^p, C^p) \mid C^p = 1\}$ and $S_- = \{(\mathbf{X}^p, C^p) \mid C^p = 0\}$ (C^p is the desired output of the pattern classifier for the input pattern $\mathbf{X}^p$), it is the goal of a *perceptron training* algorithm to attempt find a weight vector $\hat{\mathbf{W}}$ such that $\forall \mathbf{X}^p \in S_+, \hat{\mathbf{W}} \cdot \mathbf{X}^p > 0$ and $\forall \mathbf{X}^p \in S_-, \hat{\mathbf{W}} \cdot \mathbf{X}^p \leq 0$. If such a weight vector ($\hat{\mathbf{W}}$) exists for the pattern set S then S is said to be *linearly separable*. Several iterative algorithms are available for finding such a $\hat{\mathbf{W}}$ if one exists [Nilsson, 65; Duda & Hart, 73] or a reasonably good weight vector that correctly classifies a large fraction of the training set if S is not linearly separable (e.g., *pocket algorithm* [Gallant, 93], *thermal perceptron* [Frean, 90], *barycentric correction procedure* [Poulard, 95].) For a detailed comparison of the single TLU training algorithms see [Yang *et al.*, 96].

When S is not linearly separable, a multi-layer network of TLUs is needed to learn a complex decision boundary [Chen *et al.*, 95] that correctly classifies all the training examples. In recent years, several *constructive* learning algorithms that incrementally construct such multi-layer networks have been proposed in the literature [Honavar, 90; Honavar & Uhr, 93; Parekh *et al.*, 95; Gallant, 90; Mézard & Nadal, 89; Frean, 90]. Most such algorithms decompose the the hard task of determining the necessary network topology and weights to two subtasks: Incremental addition of one or more threshold neurons to the network when the existing network topology fails to achieve the desired classification accuracy on the training set; and

*This research was partially supported by the National Science Foundation grant IRI-9409580 to Vasant Honavar. The authors are grateful to Rajesh Parekh for helpful suggestions on this paper.

training the added threshold neuron(s) using some variant of the perceptron training algorithm (e.g., the pocket algorithm).

A primary advantage of such constructive algorithms is their potential ability to build near-minimal networks of TLUs from a given training set [Honavar, 90; Honavar & Uhr, 93; Parekh *et al.*, 95]. Clearly, if we have a data set that is linearly separable, the task of generating the minimal network reduces to training a single threshold neuron. Thus, every reasonable effort should be made to choose an efficient representation of input patterns that makes the data set separable. In the case of discrete (e.g., boolean) input patterns, this entails automatic or manual selection of critical higher order combinations of inputs. In general, the choice of such a representation is a difficult problem for which no computationally efficient yet general solutions are currently available. However, even rather simple and straightforward *quantization* techniques can turn an otherwise non-separable data set (e.g., the 4-quadrants problem (the real-valued analog of the exclusive OR problem) in which the two quadrants along a diagonal are assigned to one class and the remaining two quadrants to the other class) separable [Hrycej, 92].

The primary focus of this paper is on discretized representation of real-valued input patterns. This is motivated by the following considerations (among others): discrete neural networks can generally be realized much more simply in digital hardware (using current VLSI design technology) than their continuous counterparts; a number of provably convergent learning algorithms are available for constructing multi-layer networks of binary perceptrons.

2 Representation of Real Values by Collection of Discrete Inputs

A variety of techniques for approximating real numbers by collections of discrete inputs are available in the literature [Gallant, 93]. These include:

- *binary representation*: A binary vector is interpreted as a (possibly fractional) binary number. For example, a value x in the interval $[1, 15]$ in subintervals of 0.5 is represented by a binary or bipolar vector of length 5 (so that 32 possible values can be encoded). Binary representations are highly sensitive to error and it is extremely difficult to increase or decrease precision by adding or removing some of the binary input cells.

- *uniform point approximation*: A real value x in the interval $[a, b]$ is represented using a collection of bipolar inputs $u_1, u_2, \cdots, u_k$ where $u_j = 1$ iff $a + \frac{(j-1)(b-a)}{k} \leq x < a + \frac{j(b-a)}{k}$ and $u_j = -1$ otherwise. While point approximations are better than binary representation, they are susceptible to error if several input cells (including one that has a value of $+1$) fail. Also, adding new input cells to increase precision changes the *interpretation* of existing cells.

- *uniform cutoff approximation* (thermometer code): A real value x in the interval $[a, b]$ is represented using a collection of bipolar inputs $u_1, u_2, \cdots, u_k$ where $u_j = 1$ iff $x \geq a + \frac{j}{k+1}(b - a)$ and $u_j = -1$ otherwise. The advantage of cutoff approximations is that they are very robust to error or missing values because even removal of several cells leaves us with usable representations. Similarly, it is rather straightforward to increase precision by adding new input cells.

In light of the advantages of thermometer code, we will restrict the discussion that follows to algorithms for generating suitable thermometer codes for a given set of multi-dimensional real-valued (or multi-valued) patterns. For reasons of economy, it is advantageous to allow *non-uniform* cutoff approximations so that different real input dimensions (and possibly different parts of the domain for each input dimension) can be encoded at the necessary levels of precision. One approach to this problem is to use algorithms for generation of non-uniform thermometer codes start with low precision representations and increase the precision as necessary. Such algorithms need to be able to select both the dimension as well as the particular sub-interval to expand at each step until a suitable specified stopping criterion is met. In this paper we focus on the former.

Several different stopping criteria for thermometer code generation suggest themselves. Two such criteria that seem to be quite natural ones to consider are:

- *faithfulness:* A training set is said to be *faithful* if no two training examples belonging to different classes have the same input representation. (Clearly, if a training set is not faithful, it is possible to have contradictory examples).

- *separability:* (see above). If patterns differ in terms of at least some attribute values, (which is often the case for real-valued pattern sets), they have a fairly high likelihood of becoming separable as a result of quantization. This is due to the fact that in such a case the probability that there exists a positive linear combination of the quantized patterns that is equal to 0 (i.e., the sufficient and necessary condition for linear nonseparability [Siu *et al.*, 95]. is met) is rather small.

It is to be noted that although it appears to be desirable to transform a given training set into a *separable* training set using a suitable quantization (since it allows one to use mathematically well-characterized and efficient training algorithms to attain zero classification error on the training set), it may not always yield classifiers that generalize well on a test set that is disjoint from the training set. The study reported in this paper was motivated by the need for efficient quantization techniques that yield relatively compact codes that enhance separability without sacrificing generalization.

3 Two Algorithms for Thermometer Code Generation

The basic procedure of thermometer code generation algorithms investigated is as follows:

1. Generate an initial representation by assigning 1 bit for each real-valued attribute of the input patterns.

2. (a) If the termination condition (e.g., faithfulness or separability) is satisfied, return the thermometer-coded training set. Otherwise,

   (b) Select an attribute to be expanded by one dimension and update the thermometer codes for each of the patterns in the training set.

   (c) Go to Step 2 (a).

Note that several thermometer code generation algorithms can be derived from this general strategy, depending on: choice of the termination criterion; choice of the strategy for selecting an attribute in step 2 (b); and choice of how the chosen attribute is quantized in step 2 (b). Also note that the results of quantization depend critically on (the typically a-priori unknown) relationships between attributes. Furthermore, anything like a systematic search over the space of possible thermometer encodings is computationally intractable in all but the most trivial of training sets. Against this background, we implemented the following relatively simple thermometer code generation algorithms and compared their performance on several non-separable training sets.

The first algorithm RQuantizer1 selects an attribute in step 2 (b) at random according to a discrete uniform probability distribution (i.e., each attribute has the same probability ($1/N$ – where N is the number of attributes)) of being selected at a given invocation of step 2 (b).

The second algorithm EQuantizer1 uses the magnitude of *entropy* [Shannon, 48] of the training set for each possible choice of attribute to guide the selection at step 2 (b). *Entropy* of a set S provides a measure of the amount of information (in bits) needed to identify class membership of a randomly selected pattern from the set. Any change in the representation of instances in S can change the entropy associated with S. Thus it makes sense to choose an attribute which when assigned an extra code bit, yields the lowest entropy (over all of the N possible choices of the attribute) for the thermometer-coded training set. The entropy of a set of patterns S made of instances belonging to one of M classes $C_1, C_2, \cdots, C_M$ is given by: $-\sum_1^M (\frac{|C_j|}{|S|}) \log_2(\frac{|C_j|}{|S|})$ where C_j represents the number of instances in S that belong to class C_j. Note that EQuantizer1 is a *greedy* strategy that does not search over all possible sequences of attribute selections for thermometer code update.

Once an attribute is selected in Step 2 (b), both RQuantizer1 as well as EQuantizer1 update the thermometer codes for patterns in the training set by requantizing the selected attribute using uniform thermometer coding. (A variety of other approaches are possible but we leave them for future investigation).

4 Experimental Results

4.1 Description of Datasets

Both artificial and real world datasets are used in our simulation (real world datasets are from the machine learning database repository of University of California at Irvine[1]).

4.1.1 Artificial Datasets

- Two Circles (2CL): This data is for classifying two concentric circles. Points in smaller circle belong to one class and the rest belong to the other class. 200 data points for training and 100 data points for testing are selected at random from a circle of radius 3 in two dimensional space. Points whose distance from the origin is less than 1.5 are assigned to one class as against those whose distance is greater than 1.5.

- Two Spirals (2SP): Two spirals are generated between [-7,7] for both axes in two dimensional space. 192 points are generated, of which 144 patterns are selected for training and the remaining 48 patterns are used for testing.

- Four Quadrants (4Q): This data is the real-valued analog of the exclusive OR problem. The two quadrants along a diagonal are assigned to one class and the remaining two quadrants to the other class. 140 data points for training and 70 data points for testing are randomly generated between [-3,3] in two dimensional space.

4.1.2 Real-World Datasets

- Thyroid Gland (Thyroid): This dataset is used to predict whether a patient is thyroid to the class euthyroidism, hypothyroidism or hyperthyroidism. There are 5 real-valued attributes and 3 classifications. 143 patterns are chosen for training and 72 patterns are chosen for testing from 215 total patterns.

- Lenses: This dataset is used to fit contact lenses. There are 4 real-valued attributes and 3 classifications. 16 patterns are chosen for training and 8 patterns are chosen for testing from 24 total patterns.

- Balance Scale Weight and Distance (Balance): This dataset is generated to model psychological experimental results. There are 4 real-valued attributes and 3 classifications. 400 patterns are chosen for training and 225 patterns are chosen for testing from 625 total patterns.

- Glass Identification (Glass): The use of this data is motivated by criminological investigation. There are 9 real-valued attributes and 7 classifications. 161 patterns are chosen for training and 53 patterns are chosen for testing from 214 total patterns.

- BUPA Liver Disorders (Liver): This dataset is for checking the sensitive factors to liver disorders. There are 6 real-valued attributes and 2 classifications. 200 patterns are chosen for training and 145 patterns are chosen for testing from 345 total patterns.

- Pima Indians Diabetes (Pima): This dataset shows whether the patient shows signs of diabetes. There are 8 real-valued attributes and 2 classifications. 576 patterns are chosen for training and 192 patterns are chosen for testing from 768 total patterns.

4.2 Experimental Results

The performance of RQuantizer1 and EQuantizer1 were compared on both artificial and real-world datasets through a series of experiments. The results presented reflect averages over 10 runs (with random initial weights) using *Pocket algorithm with ratchet modification* [Gallant, 93] for training the single layer perceptron.

[1] The authors would like to thank Dr. Mike Pazzani of University of California at Irvine for maintaining the on-line repository of machine learning datasets.

A run was terminated upon attaining 100% accuracy on the training data or when the *pocket* weights did not undergo update for a stretch of 100,000 epochs. Training and test accuracies were computed and recorded both at the end of a run as well as when *faithfulness* criterion was met.

Table 1 summarizes the results of simulations. Note *'s for several datasets in EQuantizer1. They are due to the indefinite expansion for certain attributes based on entropies, which does not generate any thermometer codes at all. For Pima dataset, separable representation experiment was not included because of its inordinate running time with the high number of expansion for getting 100% training accuracy. Thus, after sufficient time is passed, the simulation is stopped in both cases.

	Real			RQuantizer1			EQuantizer1			
Dataset	*D*	*L*	*G*	*D*	*L*	*G*	*D*	*L*	*G*	*S.C.*
2CL	2	77	74	54.6	100	95.4	168	100	89.7	F,S
2SP	2	56	49	34.1	74.7	63.3	47	75.9	69.6	F
				119	100	61.9	135	100	65	S
4Q	2	73.1	72.3	2	77.9	87.1	2	77.9	87.1	F
				125.2	100	48	155	100	34.9	S
Thyroid	5	100	69.4	55.5	100	92.2		*		F,S
Lenses	4	100	71.3	8	100	81.3	5	100	77.5	F,S
Balance	4	91.2	78.3	23	100	87	16	100	86.7	F,S
Glass	9	86.5	50.9	146.2	98.8	55.1		*		F
				159.1	100	57		*		S
Liver	6	74.1	70.8	97.8	85.6	67		*		F
				222.3	100	67.6		*		S
Pima	8	69.3	63.7	80.7	80.6	74.2		*		F

Table 1: Dimension (D), Training (L) and Test (G) Accuracy For Quantized Datasets. $S.C$ stands for the stopping criterion, F and S indicate *faithfulness* and *separability* respectively. A '*' indicates failure to achieve faithfulness as a result of potentially indefinite expansion of some attributes when EQuantizer1 is used or failure to attain *separability* within the allowed simulation time.

Firstly, we can see that both training and testing accuracy are generally substantially better for thermometer encoded datasets (as compared to real-valued datasets). The only exception was the test accuracy on and Liver dataset and on four quadrants dataset (with separable Quantization). One explanation for this might be lack of sufficient number of training samples for the number of weights used in the perceptron with thermometer coded input. This needs further theoretical and empirical investigation.

Secondly, the test accuracy or generalization obtained using the faithful representation is comparable to (and occasionally better than that of) separable representation. Given the fact that faithfulness is achieved with generally far fewer input dimensions (i.e., coarser quantization) than separability, and that the generalization obtained is generally higher than that for the unquantized real-valued datasets, faithfulness appears to be preferable to separability as a stopping criterion for quantization.

Finally it is noteworthy that RQuantizer1, has performance comparable to the much more computationally expensive EQuantizer1 (in terms of the number of dimensions and generalization). Thus RQuantizer1 offers a promising approach to Quantization of real-valued pattern sets for pattern classification using discrete perceptrons.

5 Conclusions

Appropriate quantization can often transform a difficult pattern classification problem into an easier one. Although several quantization techniques (e.g., LVQ) have been developed in the literature [Haykin, 94] , most of them seem to be motivated by applications such as image compression (where the fidelity of reconstruction is the primary performance measure of interest) as opposed to pattern classification. This paper has compared the performance of some rather simple and intuitively appealing algorithms for quantizing patterns using uniform thermometer codes (for each real-valued or multi-valued attribute). It is worth noting that in several cases, a non-separable real-valued dataset became separable when faithful thermometer codes were generated. Even in cases where faithfulness did not result in separability, generalization on test data was generally substantially better than that for unquantized data. Generalization properties of faithful thermometer codes deserve further theoretical and empirical investigation.

The randomized thermometer code generation algorithms explored in this study can potentially improved further along several directions. In particular, use of non-uniform thermometer codes might yield more efficient representations (i.e., with fewer dimensions) with superior generalization than their uniform counterparts. Also of interest is the judicious use of quantization with appropriate constructive neural network learning algorithms for pattern classification as well as comparison of variants of the proposed quantization algorithms with other techniques developed in the context of signal or image compression.

References

Chen, C-H., Parekh, R.G., Yang, J., Balakrishnan, K., & Honavar, V. (1995). Analysis of Decision Boundaries Generated by Constructive Neural Network Learning Algorithms. *Pages 628–635 of: World Congress on Neural Networks*, vol. 1.

Duda, R. O., & Hart, P. E. (1973). *Pattern Classification and Scene Analysis.* New York: John Wiley and Sons.

Frean, M. (1990). *Small Nets and Short Paths: Optimizing Neural Computation.* Ph.D. thesis, Center for Cognitive Science, University of Edinburgh, UK.

Gallant, S. I. (1990). Perceptron Based Learning Algorithms. *IEEE Transactions on Neural Networks*, **1**(2), 179–191.

Gallant, S. I. (1993). *Neural Network Learning and Expert Systems.* Cambridge, Massachusetts: The MIT Press.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* New York: Macmillan.

Honavar, V. (1990). *Generative Learning Structures and Processes for Generalized Connectionist Networks.* Ph.D. thesis, University of Wisconsin, Madison, U.S.A.

Honavar, V., & Uhr, L. (1993). Generative Learning Structures and Processes for Connectionist Networks. *Information Sciences*, **70**, 75–108.

Hrycej, T. (1992). *Modular Learning in Neural Networks.* New York: John Wiley and Sons.

Mézard, M., & Nadal, J. (1989). Learning in Feel-forward Networks: The Tiling Algorithm. *J. Phys. A: Math. and Gen.*, **22**, 2191–2203.

Nilsson, N. (1965). *Learning Machines.* New York: McGraw-Hill Book Co.

Parekh, R. G., Yang, J., & Honavar, V. (1995). *Multi-category Constructive Neural Network Learning Algorithms for Pattern Classification.* Tech. rept. ISU-CS-TR 95-15a. Department of Computer Science, Iowa State University, Ames, Iowa.

Poulard, H. (1995). Barycentric Correction Procedure: A Fast Method of Learning Threshold Units. *Pages 710–713 of: World Congress on Neural Networks*, vol. 1.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 379–423.

Siu, K-Y., Roychowdhury, V., & Kailath, T. (1995). *Discrete Neural Computation - A Theoretical Foundation.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Yang, J., Parekh, R. G., & Honavar, V. (1996). *Empirical Comparison of Graceful Variants of the Perceptron Learning Algorithm on Non-Separable Data Sets.* In preparation.