Some Experiments with Fixed Non-Linear Mappings and Single Layer Networks.

Richard W. Prager

July 1992

Cambridge University Engineering Department

Technical Report CUED/F-INFENG/TR.106

Abstract

Kanerva's sparse distributed memory model consists of a fixed non-linear mapping, called location matching, followed by a single layer of adaptive dot-product step-threshold links. Various networks of this type are tested on three tasks in order to discover the circumstances in which this type of network provides an efficient solution.

The networks provide more competitive performance when the dimensionality of the input patterns is fairly low. For high dimensional inputs the performance of the single layer network alone is better on the test data, whereas for low dimensional inputs a two layer adaptive network is required and this needs much more training before it can produce a better test performance.

A new 'location pruning' technique is reported which improves the design of the location matching mappings. The resulting network is extensively tested on large pattern classification tasks to demonstrate the benefits of the algorithm.

The experiments show that the main benefit of the location matching networks is their ability, using the location pruning algorithm, to train in roughly $\frac{1}{2}$ to $\frac{1}{10}$ of the training iterations required by single or double layer adaptive networks on the same tasks.

Contents

1 Introduction

Kanerva's sparsely distributed memory model consists of a fixed non-linear mapping followed by a single layer of dot-product step-threshold links [PK88]. I will describe this structure, and various modified versions, in some detail and then show how they perform on three realistic pattern classification tasks. The tasks vary in the dimensionality of their pattern vectors and in the number of possible classes. The object is to shed light on the types of application for which this sort of model is most useful.

2 Types of Networks

The sparse distributed memory model, as originally described by Kanerva, consists of a location matching layer, which requires binary inputs, followed by a single layer of adaptive links. The location matching procedure has been extended to work with real valued input vectors [P&F88, CP&F91] and it is this more general version of the mapping which is described below. A novel enhancement to increase the efficiency of the location matching is also reported. This improvement is called "location pruning."

In this section both the mappings of the Kanerva model are described. First we discuss the adaptive single layer network and its training algorithm; then the fixed location matching preprocessor. Finally the new location pruning algorithm is introduced.

2.1 A Single Layer Network

This work is based on a simple adaptive network consisting of one layer of dot-product links. An input vector $\mathbf{x} = \{x_1, x_2, x_3, \ldots, x_n\}$ determines the values of a set of n input units. These units are fully connected to m outputs $\mathbf{y} = \{y_1, y_2, y_3, \ldots, y_m\}$ through a matrix of weights. The weight between input i and output j is w_{ij}. A bias input is provided as x_0 which is considered to be fixed equal to one. The outputs are calculated from the inputs as follows:

$$\begin{aligned} \sigma_j &= \sum_{i=0}^{n} x_i w_{ij} \\ y_j &= \begin{cases} 1 & \text{if } \sigma_j \geq 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Notice that the output vector of the network is always binary. The input vector may be either binary or real valued.

2.2 Training the Network

In this work the network was trained using a relaxation algorithm. This involves repeatedly presenting training examples to the network and each time altering the link weights just enough to make all the output units respond correctly. This algorithm, and a number of others which perform the same task are clearly presented in Chapter 5 of [D&H73].

To train the network, pairs of associated input and output vectors are taken from the training set at random. The network is run using the input vector and the outputs are compared with the known correct values. Where an output bit is correct no alterations are made to its link weights. For each incorrect y_j we compute the correction

$$\delta_{ij} = \rho \left(\frac{\sigma_j x_i}{\sum_{i=0}^{n} x_i^2} \right)$$

for each of the weights on links associated with this output. The quantity ρ is called the learning rate, and is set to one initially and gradually reduced as the training progresses.

For every w_{ij} leading to an incorrect output we correct the corresponding weight using

$$w_{ij}^{\mathrm{new}} = w_{ij}^{\mathrm{old}} - \delta_{ij}$$

Notice that the algorithm involves adjusting the weights immediately after processing each training example. This enables fast convergence on very large training sets and incremental training if some of the training data is not initially available.

Reducing the value of ρ as the training progresses is often not necessary for linearly discriminable problems. It does however improve the quality of the solution obtained for training sets which are not linearly discriminable.

2.3 Location Matching

In order to compute a non-linear discriminant function using a single layer network it is necessary to transform the input to a higher dimensional space (see figure 1). We use a location matching mapping to convert the representation to a binary vector which provides a good input for the relaxation training algorithm described above.

The simplest form of location matching involves choosing a large number of random points in the input space $\{\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3, \ldots, \mathbf{G}_n\}$. These are the locations, and each one is associated with a single bit $\{x_1, x_2, x_3, \ldots, x_n\}$ in the output representation from the location matching mapping. The locations are randomised when the network is initially set up and they are then considered fixed. Location matching is a fixed non-linear mapping, the locations are not permitted to move around during network training.

The idea is that each bit x_i in this output is turned on if and only if the input is close, in some sense, to the corresponding location $\mathbf{G}_i$. To implement this we define a metric $\mathcal{M}()$ on the input space and fixed constant radius, R. For an input $\mathbf{F}$ each element of the the $\mathbf{x}$ vector is determined as follows:

$$x_i = \begin{cases} 1 & \text{if } \mathcal{M}(\mathbf{F}, \mathbf{G}_i) < R \\ 0 & \text{otherwise} \end{cases}$$

The radius R is set so as to ensure that some specified fraction of the locations are activated for each input. Depending on the problem and the architecture of the rest of the network, between 1% and 20% location activation is used.

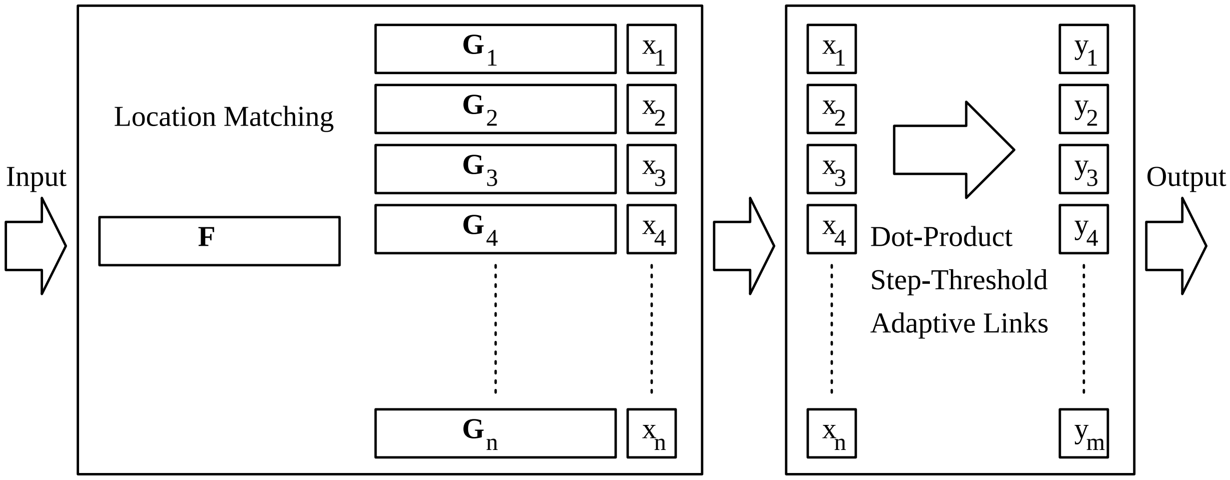

Figure 1: Network structure consisting of a fixed non-linear mapping followed by a single layer of adaptive links.

This location matching procedure transforms the input pattern to a high dimensional binary representation while preserving information on the closeness of input patterns under the metric $\mathcal{M}()$. Similar patterns in the input space will map to binary vectors with a large number of identical bits, whereas different inputs will map to vectors a large Hamming distance apart.

In [CP&F91] we showed that if the a hypercube metric is chosen for $\mathcal{M}()$ then each location is usually only functionally dependent on a small fraction of the elements of the input vector. When the input vector is of high dimensionality a considerable increase in efficiency can be obtained by using this metric. The hypercube metric is defined:

$$\mathcal{M}(\mathbf{F}, \mathbf{G}) = \text{max over all } i \text{ of } |f_i - g_i|$$

Instead of having to work out a sum of squares across the complete input one just has to check a small number, usually between 3 and 10, of the input values against fixed thresholds. All the other coordinates of the input vector are then taken to be within the required radius of the location irrespective of their values. In order to configure this arrangement correctly the number of checks performed by each location should be randomised from a Poisson distribution. Given the likelihood of a check being satisfied (usually this averages to 0.5) the poisson distribution can be scaled to give any desired average rate of activation among the locations.

2.4 Location Pruning

Choosing the locations completely randomly is inefficient. Often the result is a model in which a significant proportion (say 20%) of them play no useful role in the pattern recognition task. We need a technique for choosing more locations in regions of the input space which are most critical to the task being performed, while retaining a reasonable coverage over the rest of the space in order to ensure robust recognition performance.

One idea might be to base the location centres on the some examples from the training data. This is not a good strategy because it is the edges of the activation domains of the locations rather than their centres which need to be close to the input patterns. A good configuration would place the edges of the activation regions of many locations along pattern class boundaries in the input space.

In this work an approach has been devised which involves starting with a large number of locations and pruning them down until one is left with a useful subset. The object is to keep a group of locations which are all activated by a reasonable proportion of the training examples, and which are as different from each other as possible. One might, for example, start with 20,000 locations and prune them down to 500. These 500 may then be used to to form a location matching mapping which is more efficient to train and use than the original 20,000 and much more effective than 500 completely random locations.

The pruning is performed in two stages. In the first pass all the locations which are not active for more than a specified percentage of the input patterns are discarded. Typically 20,000 locations can be reduced to roughly 1,500 by keeping only those which respond to 20% or more of the training examples. The second pruning stage is more complicated and involves using simulated annealing to choose 500 of the remaining 1,500 locations such that the sub-sets of input patterns that they respond to are as different as possible.

The optimisation required in this second pruning stage is performed using a Boltzmann machine [AH&S85]. A single node in the Boltzmann machine is used to represent each of the 1,500 remaining locations. The Boltzmann machine is run through a single annealing schedule. If, at the end of the annealing, the node is on then the location is retained. If the node is off then the location is discarded. The Boltzmann machine is fully interconnected using only negative weights on the links. These weights are set by gathering statistics on the number of training examples which a given pair of locations both respond to. Thus if a pair of locations are on together 50 times when processing the training data, the weight between the corresponding nodes in the Boltzmann machine would be -50.

Every node in the Boltzmann machine is given the same positive bias which is controlled during the annealing schedule to ensure that roughly the correct number of nodes are left on when the machine reaches equilibrium. If, at a particular stage in the annealing, there are n out of a possible p nodes on then

$$\frac{n}{p} = \frac{1}{1 + \mathrm{e}^{\frac{-\varepsilon}{T}}}$$

where ε is the average activation energy of the nodes in the machine including the global bias. Rearranging we obtain:

$$\frac{\varepsilon}{T} = -\ln\left(\frac{p}{n} - 1\right)$$

so if we want w of the p nodes to be on we need to change the global bias by

$$\left(\frac{\varepsilon}{T}\right)_{\text{desired}} - \left(\frac{\varepsilon}{T}\right)_{\text{current}} = \ln\left[\frac{\frac{1}{n} - \frac{1}{p}}{\frac{1}{w} - \frac{1}{p}}\right]$$

The annealing schedule used for the Boltzmann machine is a geometric progression starting at a temperature equal to the maximum magnitude activation energy of any node and with a common ratio of 1.01 [PH&F86].

This two stage pruning strategy provides an efficient way of getting a fairly small set of useful locations while still retaining a reasonable degree of coverage of the input space.

Task	Inputs	Binary Outputs	Training Examples	Testing Examples
Vowels	4	10	1000	494
Words	329	146	6620	2620
Wheat	33	1	23006	2000

Table 1: Characteristics of the three tasks.

3 Pattern Classification Tasks

In this section the three tasks used in the experiments are described. The length of the input and output vectors and the sizes of the training sets are summarised in table 1.

3.1 The Peterson & Barney Vowel Classification Problem

The version of the Peterson Barney [P&B52] data used in these experiments consists of two repetitions of 10 vowels by 76 people. There were 33 men, 28 women and 15 children. Some of the data is missing and 1494 of the 1520 possible vowels are present in the database. We received the data from Steven J. Nowlan (Toronto) who originally obtained it from Ann Syrdal (AT&T Bell Labs, Holmedell) in October 1989.

Each data item consists of four real numbers specifying the four formant frequencies together with an indication of the vowel being spoken and whether the speaker was a man, woman or child. The age and gender information was not used in these experiments. The database was split randomly into a 1000 example training set and a test set of 494 examples. The task thus consisted of taking four numbers at the input of the classifier and determining which vowel corresponded to that particular set of formant frequencies.

3.2 The Lex-Net Word Classification Problem

This word recognition problem involves generating a mapping from the output of a connectionist phoneme recogniser [R&F91] to a transcription in terms of complete words. The phoneme recogniser produces an output vector every 16ms specifying the probabilities of each phoneme in that segment of speech. These probability vectors are processed using the time-compression network developed by the LEARN project [RFR&P91, RFP&R92]. Sequences of similar probability vectors are removed and replaced by a single representative vector derived from their average. The number of frames compressed in this way is also recorded for future use if needed.

Each of the composite phoneme probability frames is labelled with the word in which it occurred. In the initial experiment 6620 such frames, derived from the DARPA resource management task, form the training set and a further 2620 make up the test set. There are forty nine possible phonemes in each frame and 146 possible output words. In order to be able to recognise the output words from the phoneme frames the classifier has to make use of context. In these experiments three frames of context were provided on each side of the labelled frame. This resulted in seven-frame input vectors with a total of $7 \times 49 = 329$ values.

The pattern classes are not linearly discriminable but a linear classifier can perform quite well as we shall see.

3.3 The Wheat Problem

This problem involves detecting one particular strain of wheat from among 22 others. The database contains 23006 grains of wheat of which roughly half are mercia and the others are different varieties. Each grain is described by a vector of 33 real numbers and a single bit indicates whether it is a mercia grain or not. The feature vectors are derived using a new image based system developed at the National Institute for Agricultural Botany in Cambridge [K&D86, PDK90]. A further set of 2000 grain measurements forms a test set for the task.

To solve this problem a system must have 33 real valued inputs and a single bit output to indicate mercia or non-mercia. As there are 23 different types of wheat, only one of which we wish to detect, the background pattern class is highly multi-modal. This provides a particular challenge for pattern recognition systems.

4 Experiments

Three networks were used to solve each of the three pattern classification tasks. A single layer of links alone, the links with a conventional location matching preprocessor, and the links with a location matching preprocessor configured using location pruning. A number of extra control experiments were performed using the vowel data alone because the smaller training set in this task made it quicker to train.

Preliminary experiments were performed to adjust the parameters of the models and appropriately configure them for the tasks. An average location activation of 20% was chosen as this appeared to give the best performance on the test data with the relatively small numbers of locations being used. The weight update factor ρ was varied according to the formula $\rho = \frac{10}{9+\mathrm{cycle}}$, where 'cycle' was the number of training cycles completed.

A training cycle did not necessarily involve training on every pattern in the training data. Training examples were picked at random from the training set and used to train the network. The weight updates were performed immediately after each training presentation. When a number of examples equal to the number in the training set had been presented, a cycle was said to be complete.

The networks were trained until the performance on the training data appeared not to be improving any further. All three tasks were attempted using a single layer network alone, then a fixed mapping of 500 random locations was added as a preprocessor. An alternative fixed

Network Type	Vowels	Words	Wheat
Single Layer Network Alone	59.3	67.5	86.9
100 Random Locations	53.8		
Approx. 100 Pruned Locations	72.9		
500 Random Locations	82.4	43.2	77.4
Approx. 500 Pruned Locations	83.8	59.3	82.8
Two layer Backprop Network	85.2		

Table 2: Network Performance Percentages on Test Data.

mapping of pruned locations was also tried. In all cases the algorithm was set to prune to a target of 500. The wheat and word tasks started with 20,000 locations and pruned to 453 and 437 respectively. The vowel task was started with 10,000 and pruned to 467.

As the vowel task involved a rather smaller training set a number of extra control experiments were performed. Location matching layers with 100 random and 96 pruned locations were tried. In this case the pruning started from 10,000 locations with a target of 100. Results for a conventional two layer backpropagation network applied to exactly the same problem were kindly made available by the author of [JPMG92]. In this network there were 15 hidden units and the learning rate η was 0.9, with a momentum term α of 0.1

In section 2.3 above, the use of checks on individual elements of the input vector is described. Each check normally has three parameters; which element of the input vector it refers to, the threshold and a binary quantity specifying which side of the threshold the input must be on for the check to be satisfied. A location is considered to be on when all its checks are satisfied. In the Lex-Net word classification experiment a slightly more complicated type of check was used. In this case two sets of four input elements are randomly chosen and the check is satisfied if the largest of the eight input values occurs in one set of four rather than the other.

5 Results

The results are presented using two error measures. For both training and test performance the percentage of output vectors completely correct is calculated and these figures are detailed in the subsections below. For the test data I have also measured the proportion of the examples for which the highest responding output unit is the correct answer. These percentages are summarised for all three tasks in table 2.

5.1 Peterson & Barney Vowel Results

The results for the Peterson and Barney vowel task are shown in table 3. The single layer network on its own performs poorly. This is probably because of the low dimensional input vector; just the four formant values.

Network Type	Training Cycles	Complete correct output vectors		Test: Highest Output is the Correct Answer
		Train	Test	
Single Layer Network Alone	4371	33.4	36.8	59.3
100 Random Locations	2811	48.8	39.7	53.8
Approx. 100 Pruned Locations	2780	83.3	55.5	72.9
500 Random Locations	1303	100.0	67.0	82.4
Approx. 500 Pruned Locations	136	100.0	74.9	83.8
Two layer Backprop Network	1000	91.5	82.0	85.2

Table 3: Network Performance Percentages: Vowel Classification.

Network Type	Training Cycles	Complete correct output vectors		Test: Highest Output is the Correct Answer
		Train	Test	
Single Layer Network Alone	395	99.6	52.3	67.5
500 Random Locations	3089	97.9	26.5	43.2
Approx. 500 Pruned Locations	168	100.0	39.7	59.3

Table 4: Network Performance Percentages: Word Classification.

The experiment with the 100 random and pruned locations shows how the pruning algorithm results in much more effective location positions. This makes a difference to the performance when the network is working with a minimal number of locations. The random location network achieves only 53.8% on the test set where the pruned net reaches 72.9%. With 500 locations, even the random network does quite well and the benefits of the pruning algorithm are less noticable.

The backpropagation network performs best of all, but notice that the location matching network with 500 pruned locations does almost as well and requires significantly fewer training iterations.

5.2 Word and Wheat Classification Results

The results for the word classification task are shown in table 4 and those for the wheat classification are in table 5. As in the vowel experiments above the networks with pruned locations perform better than the random ones; they train a lot faster for the word task and produce better results on the test data.

It is not surprising that the single layer network does so well and beats the best location matching network on the word recognition task. This task has a very large input vector of 329 elements, in such a redundant representation it is understandable that a good linear discriminant can be found. It is quite remarkable that the single layer network does so well on the wheat task with only 33 input elements, again it provides the best solution of the three architectures tested.

Network Type	Training Cycles	Complete correct output vectors	
		Train	Test
Single Layer Network Alone	42818	83.3	86.9
500 Random Locations	3714	72.8	77.4
Approx. 500 Pruned Locations	3476	78.4	82.8

Table 5: Network Performance Percentages: Wheat Classification.

6 Conclusions

Experiments in [CP&F91] had led me to believe that location matching networks were best suited to classification tasks in high dimensional feature spaces with multi-modal pattern classes. The failure of the this sort of network to perform well on the wheat task is an interesting discovery and suggests that this is not at all the case.

Networks using location matching mappings appear to be useful to perform classification tasks where the input pattern is intrinsically fairly low dimensional and where the required discriminant function is complex. Where this sort of network is appropriate the location pruning algorithm results in faster training than the other networks tested and produces good performance on the test data. The training is robust and the weights may be updated after each example presentation.

7 References

AH&S85 Ackley, D.H., Hinton, G.E. and Sejnowski T.J. (1985) "A learning algorithm for Boltzmann machines" Cognitive Science, 9, pps 147–169.

CP&F91 Clarke, T.J.W., Prager, R.W. & Fallside F. (1991) "The modified Kanerva model: theory and results for real time word recognition." Proceedings of the IEE section F, Vol. 138, pps 25-31.

D&H73 Duda, R.O. and Hart, P.E. (1973) "Pattern Classification and Scene Analysis." John Wiley & Sons. New York.

JPMG92 Gosling, J.P.M. (1992) "Using the GANNET evolutionary neural network training system with the Peterson and Barney vowel database." Technical Report of the GANNET IED project (GR/F 34848). Cambridge University Engineering Department, March.

PK88 Kanerva, P. (1988) "Sparse Distributed Memory" MIT Press, Cambridge, Massachusetts, USA.

K&D86 Keefe, P.D. and Draper, S.R. (1986) "The measurement of new characters for cultivar identification in wheat using machine vision." Seed Sci. and Technol. Vol. 14 pps 715–724.

PDK90 Keefe, P.D. (1990) "Observations concerning shape variation in wheat grains." Seed Sci. and Technol. Vol. 18, pps 629–640.

P&B52 Peterson, G.E. and Barney, H.L. (1952) "Control Methods Used in a Study of the Vowels." JASA 24, pps175–184.

PH&F86 Prager, R.W., Harrison, T.D. and Fallside, F. (1986) "Boltzmann machines for speech recognition" Computer Speech and Language, 1, pps 3–27, Academic Press.

P&F88 Prager, R.W. and Fallside, F. (1988) "The modified Kanerva model for automatic speech recognition, Computer Speech and Language, 3, pps 61-81, Academic Press.

R&F91 Robinson, A.J. and Fallside, F. (1991) "A recurrent error propagation speech recognition system." Computer Speech and Language, Vol. 5, pps 257–274.

RFR&P91 Russell, N.H., Fallside F., Robinson, A.J. & Prager R.W. (1991) "Lexical access using a recurrent error propagation network." Proc. EuroSpeech pps 1023–1026, September 1991.

RFP&R92 Russell, N.H., Fallside, F., Prager, R.W. & Robinson, A.J. (1992) "Lexical access using a recurrent error propagation network with non-linear time compression." Int. Conf. on Spoken Lang Processing, Banf, October 12-16, 1992.