Predicting buffer hit ratios with neural networks

Uli Harder *and Tim MacLeod †

September 18, 2003

Abstract

A neural network is used to predict the buffer hit ratio in an Oracle database, given the access pattern and buffer size.

1 Introduction

Neural networks have come and gone into fashion several times over the last decades. Lately they have caused excitement again as computer power of even ordinary desktop machines allow them to be used in places where there was a need for super-computers before. One application for neural networks are multi-dimensional interpolation and extrapolation. Conventional methods tend to be complicated and of limited use when data of more than one dimension has to be fitted. In addition one can choose neural network types that do not require any prior knowledge about the data.

A seemingly unrelated but nevertheless important topic is that of buffer hit ratios in databases. For modern computing systems accessing data in RAM is still much faster than reading it from a hard disk. A database will have a much shorter response time when all of its data is stored in memory. However, hardware and budget limitations make this impossible for most commercial databases. Therefore a database application will have longer response times for clients whose data has to be read from disk. The buffer hit ratio measures the likelihood of finding requested data in buffer and is therefore a key metric to determine the response time. In practice it remains very complicated to predict the buffer hit ratio. For analytical results, sweeping assumptions have to be made. It is very important to get the size of the buffer right as the option of using simply a huge buffer may be very wasteful. Memory is still very expensive and also the size may cause the operating system to page which is invisible to the database. Also, for most access patterns a sufficient buffer hit ratio is achieved with very little memory compared to the size of the database.

In the design phase of databases it is notoriously difficult to predict the buffer hit ratio given the size of the buffer and the amount of accessed data. And even for a database that is already in use predicting the effects of a change in buffer size or access patterns have on the buffer hit ratio is impossible without artificial assumptions.

*Uli.Harder@metron.co.uk, Metron Technology Ltd., Taunton
†Tim.MacLeod@metron.co.uk Metron Technology Ltd., Taunton

This leads the authors to attempt a new approach using neural networks. The idea is to measure the buffer hit ratio of a database when key parameters like the size of the buffer, the amount and frequency of data accesses are changed. This data will then be interpreted using a neural network and predictions it makes will be checked against reality.

Instead of using simulated results, a commercial database (Oracle 8) was used for the experiments. A database was installed on a server and from a client PC access to the data were made. The access pattern was changed and the buffer hit ratio recorded. The figure (1) shows the buffer hit ratio for various access patterns. The access pattern as explained in more detail later is mainly the probability of a particular data set being chosen. The x-axis of the graph is the access pattern. The y-axis (pointing towards the back) is the relative buffer size. Finally the z-axis is the resulting buffer hit ratio. The plot is the surface created by the buffer hit ratio for different values of the access pattern and the relative buffer size. The shading indicates areas of the values in the buffer hit ratio.

The measured data is listed in table (1). The left column is the access pattern the top row the relative buffer size. One should note the dip in the data of the buffer hit ratio for the access pattern 50% and rel. buffer size. It will be interesting to see what the neural network makes of this measurement error.

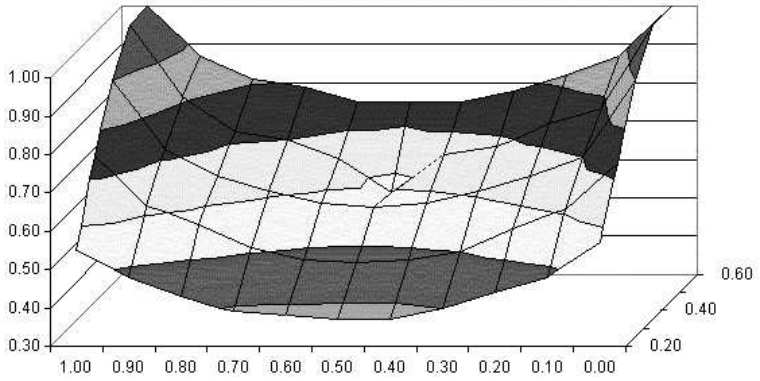

Figure 1: The data collected in the experiment

Access pattern / rel. buffer size	0.20	0.30	0.40	0.50	0.60
1.00	0.55	0.75	0.90	1.00	1.00
0.90	0.48	0.62	0.72	0.81	0.87
0.80	0.43	0.57	0.65	0.72	0.81
0.70	0.39	0.51	0.61	0.70	0.79
0.60	0.38	0.48	0.58	0.65	0.75
0.50	0.37	0.47	0.57	0.56	0.75
0.40	0.37	0.48	0.58	0.66	0.75
0.30	0.40	0.50	0.61	0.68	0.78
0.20	0.44	0.56	0.65	0.74	0.82
0.10	0.48	0.61	0.70	0.78	0.87
0.00	0.57	0.73	0.90	1.00	1.00

Table 1: The buffer hit ratio data collected in the experiment

The neural network was very successful at predicting buffer hit ratios. The authors believe it will turn into a successful tool to tune buffer sizes of databases. It will have to be tested whether the results are generic enough to be used for other databases. A downside of using neural networks, rather than an explicit mathematical model, is that one gets nowhere nearer at understanding what's

going in the databases. However, from the pragmatic point of view there is finally a tool to predict buffer hit ratios confidently.

2 The experimental configuration

The aim was to predict the buffer hit ratio of a database given the access pattern of its clients, its buffer size and the actual amount of data accessed by the clients.

In the experiment (2) there are 2 clients connecting to the database (Oracle 8.0.5). The database buffer uses a LRU algorithm to free space in the buffer if needed. The clients choose 1 out of 10 tables each. Each table holds 100,000 rows with three columns of numbers. 1000 rows are chosen at random for each select statement. The tables are created in such a way that a complete table scan is performed for each select statement and the data is kept in the buffer (by default long table scans are kept at the end of the LRU list in Oracle, in the experiment this has been changed s.t. data from lang table scans is put at the front of the LRU list as well).

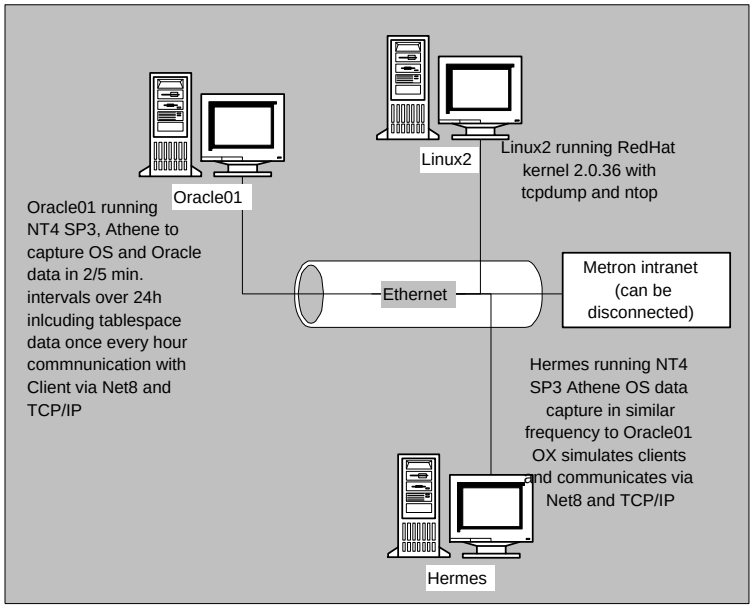

Figure 2: Experimental configuration

The total data accessable to the clients is 24,880 blocks. The buffersizes are 5000, 7500, 10000, 12500 and 15000 blocks. For the neural network computation the size of the buffer is expressed as ratio of the total database size. This makes the computations more generic and is also beneficial for the neural network.

After each session is finished the next session is chosen to be executed by one of the clients with a certain probabilty. Since there are only two clients only the probability for one of them is quoted as access pattern of the database. As a test for the experimental data the access pattern was varied from 0 to 100 % in steps of 10. This allowed a simple check for the accuracy of the data as the resulting graphs were expected to be symmetric. This was indeed the case for the data.

3 The neural network

For the calculation a simple Multi-Layered Perceptron (MLP) feedforward neural network with a single hidden layer was used [2]. The architecture of the MLP can be seen at figure (3). The number of units in the hidden layer was varied from two to twenty. The number of input units remained static at three, corresponding to the a number representing the relative size of the buffer, the access frequency of user one and the access frequency of user two. The single output unit represented the measured buffer hit ratio. Hence, a single training

item comprised of an input vector of three elements and an output vector of a single element.

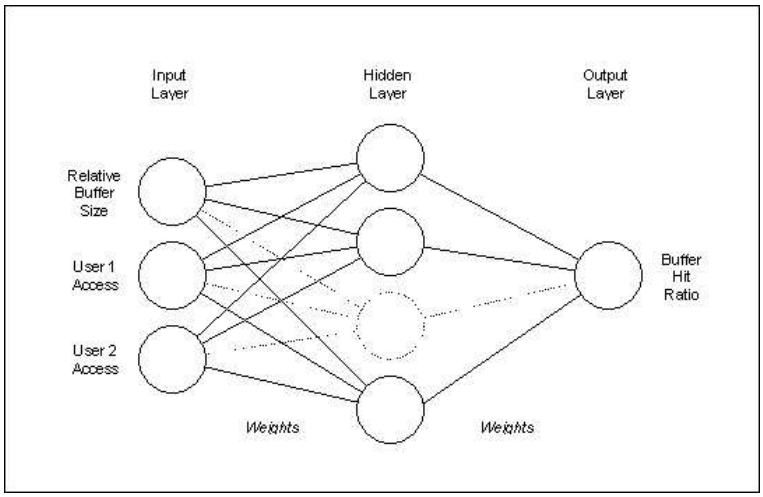

Figure 3: Neural network

The MLP was trained using a simple Backpropagation algorithm [2] where the error between expected and actual outputs was estimated for both the hidden and output layers and the weights adjusted accordingly. The number of epochs, a complete pass through all training items, varied from 1000 to 20000.

The measured data includes an obvious mesurement error for 50% buffer size and access pattern 50% .

Typically the total error of the neural network decreases as in graph (4) with a growing number of hidden layers (here 4000 epochs were used).

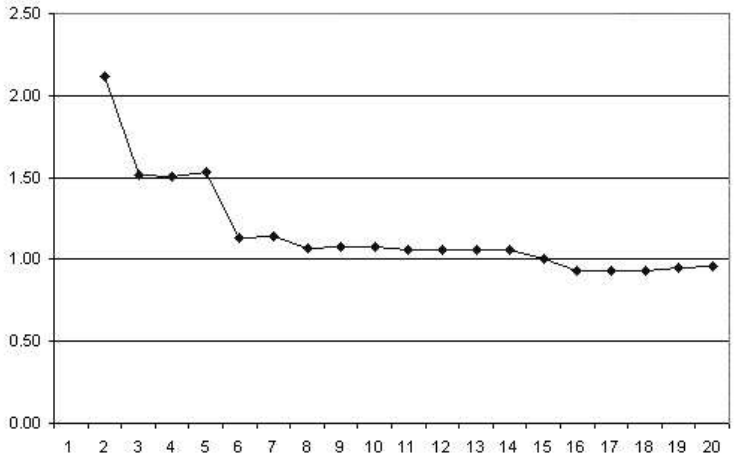

Figure 4: The total error versus the the number of hidden layers over 4000 epochs

Similarly the the number of successes grows with the number epochs. Graph (5) shows the number of successes against the length the net is run in 1000s of epochs (15 hidden layers were used in the network).

4 Evaluation of the data

For a start the network was trained to with the full dataset depicted in graph (1). When the output in grpah (6) is compared with the raw data it becomes obvious that the network is able to reproduce the input data rather well with an error typically smaller that 1%. Of course, it may be the case that the network is overtrained similarly to data being over fitted when polynomial fits are made.

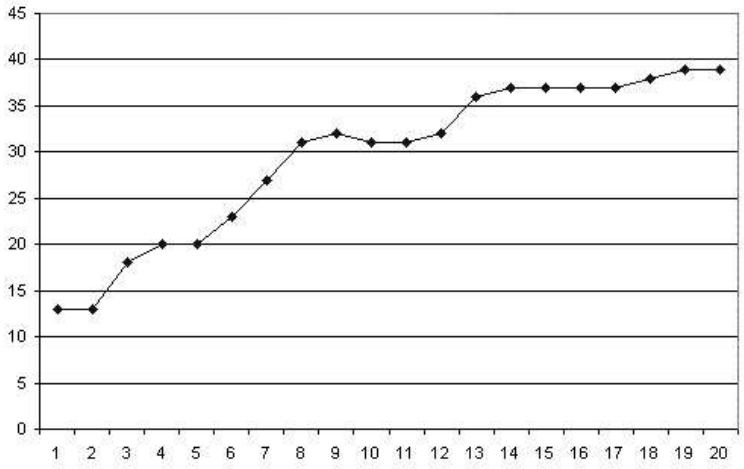

Figure 5: Epochs in 1000s vs. success in a network with 15 hidden layers

But the network actually smoothly interpolates the measurement error in the experimental data for buffer size 50% and access pattern 50%.

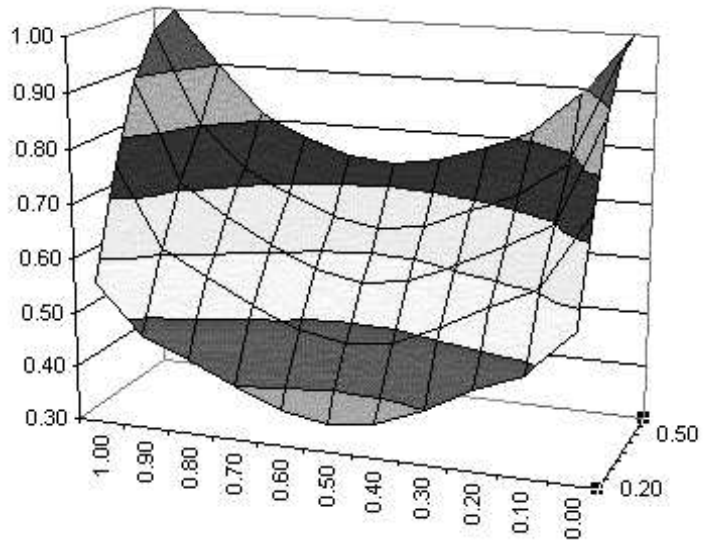

Figure 6: The neural network trained with all data from table (1). Note that the dip of the original data has been smoothed out.

Further experiments were made by removing parts of the data set. First the whole column corresponding to 40% buffersize in table (1) was removed. The result is presented in graph (7). As can be seen the neural net is able to interpolate the missing data extremely well.

Equally good results were achieved when the columns corresponding to 30% and 50% buffer size in table (1) were removed from the training data. Graph (8) shows the result. Again the missing data is remarkably well interpolated.

However, trying to recover missing data for buffersizes is not a problem usually encountered in the real world. This corresponds to asking: what is the access pattern for a given buffersize? Still, the answer shows that the neural net is very good at intrepolating data sensibly.

A realistic and sensible question is however: what buffer hit ratio is achieved for a particular access pattern? The network was trained with data only for the access pattern with 10, 30, 50, 70 and 90 %. Graph (9) shows how the missing data was interpolated and extrapolated.

Again the results look very good. Thought the extrapolation at the fringes has a rather large error of about 15%.

A DBA could use this to assess how a change in the access pattern of a

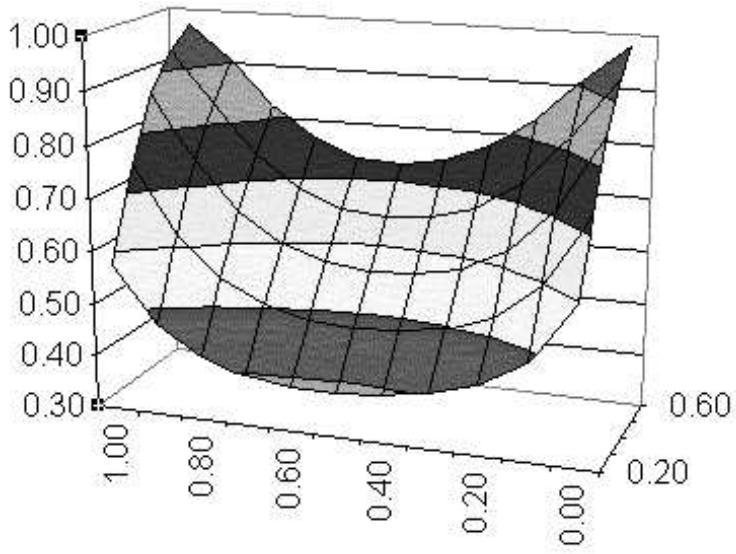

Figure 7: The column corresponding to 40% buffersize in table (1) has been removed from the training data

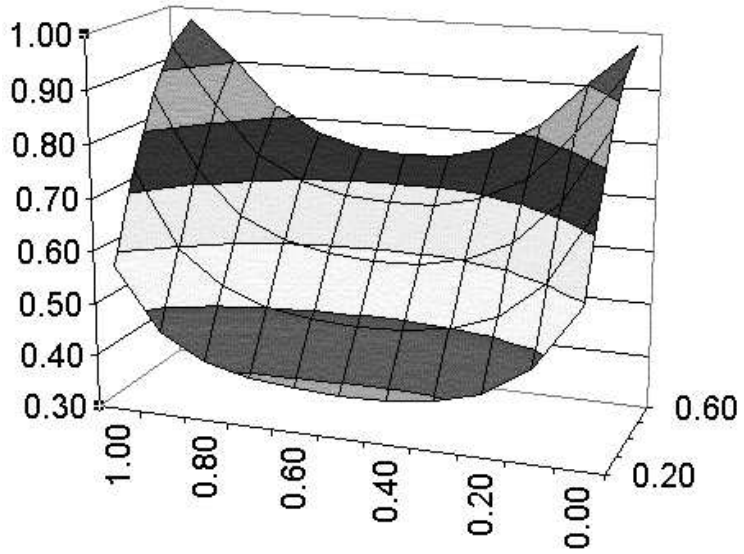

Figure 8: Columns corresponding to 30% and 50% buffer size in table (1) were removed from the training data

database would affect the buffer hit ratio. If there was a target buffer hit ratio that needs to be met, the graph could also used to make recommendations for the size of the buffer for the new access pattern.

Since all values that were used in the actual calcualtions were relative the result should be fairly generic.

5 Conclusion

It has been shown the a suitable neural network can be used to predict the buffer hit ratio of a relational database. It was possible to predict misssing data, that was measured previously, with the neural network in interpolation and extrapolation situation with at least 15 % accuracy. In most the error was well below this error margin. It has also been shown that the neural net was not overtrained as it smoothed some of the measurement errors very nicely.

The data used made extensive use of the knowledge about the access pattern of the clients using the database. However in real life the situation is slightly different. Only using very intrusive methods like tracing allow to determine

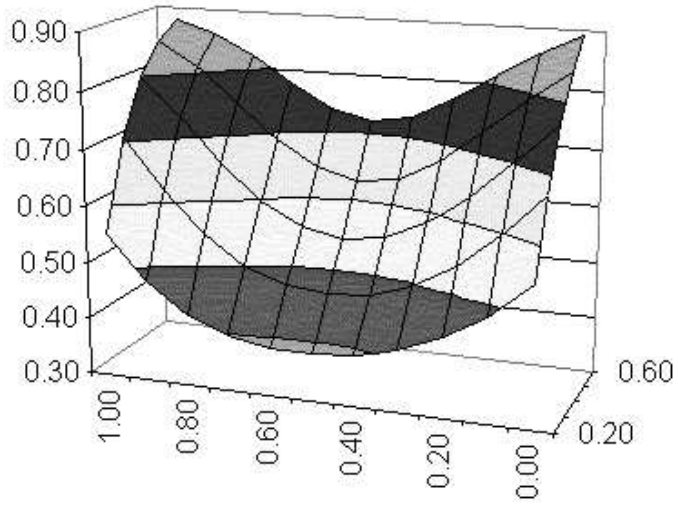

Figure 9: Network trained with data only for the access pattern with 10, 30, 50, 70 and 90 %

access pattern with reasonable accuracy. It is planned to try the same method with data that provides less detailed information.

Data that is easily obtained is the number of concurrent transactions. Also, they are easily classifiable as high/medium/low in the CPU usage, disk reads writes and long reads. One would expect that these variables, together with the buffersize, to influence the buffer hit ratio. The authors hope to investigate this matter further. another idea involves the use of neural networks to classify databases and session types, which would be immediately useful for the reailistic approach just described.

References

[1] Oracle Server 8 Concepts, Oracle documentation available at `http://technet.oracle.com/`

[2] see ch.4 in *Neural Networks for Pattern Recognition*, C.M. Bishop, OUP 1994, ISBN 0-19-853864-2; ch.6 in *Introduction to the theory of Neural Networks*, J. Hertz, A. Krogh and R.G. Palmer, Addison-Wesley 1991, ISBN 0-201-51560-1