Neural Network Processing using Database Management Systems

Michael Amos, Dr. Bruce Segee

University of Maine Department of Electrical and Computer Engineering
Instrumentation Research Laboratory

Abstract

Database Management Systems (DBMS) have become an integral part of any data storage, processing, or retrieval system. They are uniquely suited to manipulate large amounts of data while maintaining a level of data integrity and security. Meanwhile, Neural Networks are being used to perform operations on data to discover the unknown underlying input/output relationships. A significant part of Neural Network processing is reducing the input space to a manageable size. These algorithms utilize some of the same techniques that databases use (such as hashing functions) to retrieve or store large amounts of data quickly. This project utilizes a popular DBMS to build a Neural Network application that resides inside the database. This will yield many benefits that weren't previously attainable, e.g., the data and the Neural Network are both located in the same physical location. There is no need to export the data from the database, manipulate it into an appropriate format, and then use a separate Neural Network application to process the data. Another benefit is that there is no need for database users to learn external applications to process the data from their database. They simply issue an appropriate SQL statement and the Neural Network will perform the calculations and give a result. If the data is changed, the user simply re-issues the SQL statement to recalculate the outputs of the system. Utilizing this system, database users will be able to perform high level data processing tasks using the tools with which they are familiar, while appreciating significant performance gains over other implementations.

I. Introduction

Problem Description

Neural Networks have been a staple of the engineering community for decades[6]. They are useful for many engineering applications where an adaptive learning tool proves useful, such as classification and approximation of unknown functions. Unfortunately, most applications that perform Neural Network calculations are cryptic at best and are only understood by someone with years of experience and an engineering degree. If Neural Networks were able to be used by someone with little or no engineering expertise, then more people could use these powerful tools.

Since many Neural Network applications are written for a specific purpose, they expect a certain format for input data. The application is designed for that particular input format and if that format happens to change, then the application will more than likely need major changes as well. Of course, this is quite difficult if the person that wrote the original application is not still part of the group needing these modifications. Even if the person is still available to make

modifications, re-writing code that was written months, if not years, ago is a difficult and time-consuming task.

So, in order to use the given Neural Network package, one needs to extract the data from the appropriate storage place (usually a database system), convert the data to the appropriate format for the application, and perform the Neural Network calculations. One then needs to take the application output, and format it into a database friendly form and insert that data into the database for reporting and sharing throughout the group. It seems like much of the time is spend massaging the data into the appropriate input format of the various applications and little is spent actually analyzing the data.

The Solution

The solution that we are developing moves the data processing from a custom application into the database system. We believe that this will yield many benefits:

- Since the data is not moved from the database storage location, there is no need to export large amounts of data to intermediate files.
- If the data is kept in the database, it can be shared by many users simultaneously and intermediate files will not be out of date when data changes.
- The database server is probably located on a machine that is capable of handling large amounts of traffic. It probably has multiple processors, large memory capacity, and RAID capable disk storage. These factors may allow the same calculations to execute at a faster rate than on a local desktop machine.
- A database user with little or no engineering expertise may be able to utilize some of the power of neural networks.
- A significant amount of time is spend formatting the output of the CMAC program into a format that is easily shared and printed. Containing the information in the database facilitates seamless integration into WWW applications for reporting and displaying output data.

Concerns

The system must address certain concerns in order to be a viable alternative to the current Neural Network process.

Speed – The database Neural Network must perform at a comparable speed to the standard custom application. If it does not, then the usefulness of the system will be greatly reduced.

Ease of Use – The system must prove easy to use for the intermediate database user. The system must recognize standard SQL commands and be able to process the data using these commands.

Performance – If the database Neural Network is not as powerful (cannot work with large data sets with many receptive fields) as the desktop equivalent, then the system will not be a practical alternative to the standard applications already in use. It is also undesirable if the calculations utilize too many database server resources, restricting other users from using the database as before.

II. Implementation

This section focuses on the concerns raised in the previous section and how we are addressing them to produce a reliable system.

Neural Network

Since the Neural Network is the heart of the system, some time was taken to research different techniques and algorithms to chose the appropriate type. Although there are many different implementations, many (such as Radial Basis Function[8]) require complex calculations to determine the distance of a data point from the center of an activation area, or node. These complex calculations are not well suited to database computations.

The Cerebellar Model Arithmetic Computer (CMAC)[6] Neural Network does not utilize these complex algorithms. Most of the computation is performed with integer variables and addition. This is matched well with current database technology. The CMAC network, as many other Neural Networks also do, tries to map the input space into a conceptual memory, and further into physical memory. (Figure 1 shows the logical mapping from input to output.) For example, if the input range of eight integer variables is from 0 to +100 the size of the input space is 100^8, which is an unmanageable size to contend with…much larger than memory sizes in today's computers. If the amount of memory that is actually being used is small in comparison to the total size, we can use a technique called hashing to store that information in a smaller physical memory size. The CMAC utilizes a hashing algorithm which is very similar to the hashing algorithms that database systems use to rapidly locate the information stored in a table. Since this hashing algorithm is a significant part of the CMAC code, and the DBMS utilizes similar

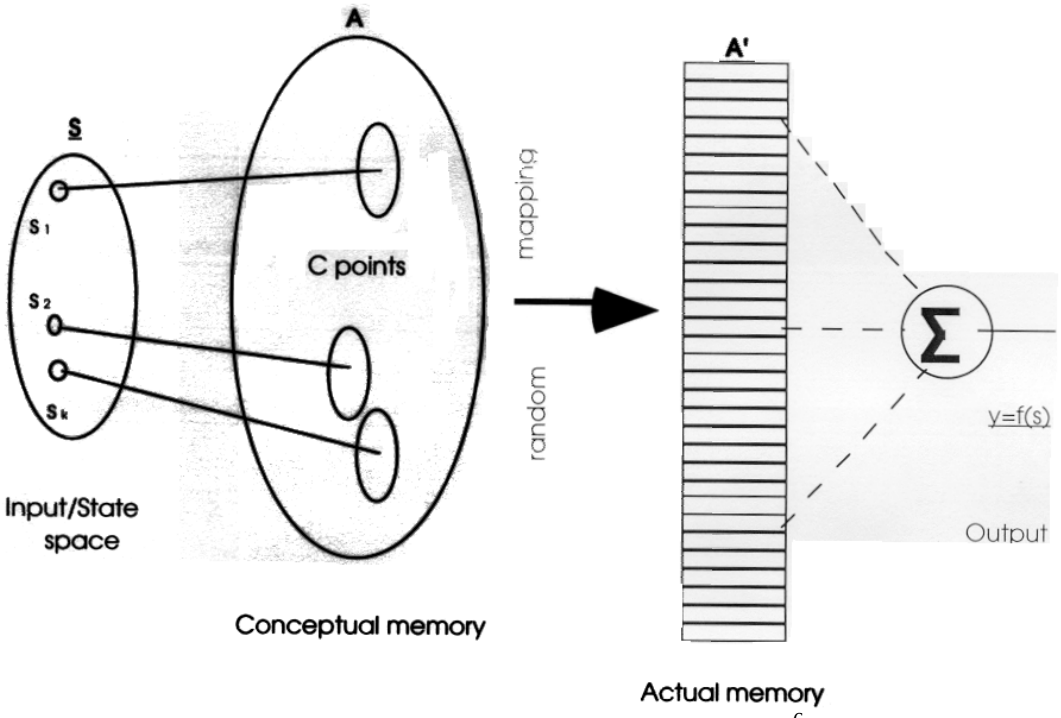

Figure 1. Input space mapping to physical memory[6]

technology, the Neural Network code can be simplified greatly.

Just because the Neural Network seems well suited for use in a database system, does not mean that the Neural Network will perform adequately. The Instrumentation Research Lab at the University of Maine has done extensive testing with the CMAC network and produced good results with test data. This previous research will provide not only valuable background for current research, but also supply a test platform to perform performance comparisons.

Database

The database system consists of several components, the first of which is the database server. The server is a computer that is specifically designed to handle the demands that running the database software will put forth. It is designed with extra memory, large, fast drive storage, powerful processors, and fast network connections. Each of these items helps to make the database server a machine (or machines) that can handle several user requests simultaneously. Having the database server helps to put the important data in one location. If the database server is too slow, then only one machine needs to be upgraded, instead of a whole office of desktops.

Having one central database server also helps to maintain data consistency. If there are multiple copies of the same data on many machines, then chances are that one of them is not correct. The database server keeps all of the data in one central location, making backups and access by all users a simple task.

The second part of the database system is the database management system. This consists of the software that controls the database, as well as a set of applications that allow the database administrator to grant and refuse access to certain users, make backups of the data, and change the schema, or structure, of the data in the system. This schema is vitally important to make the database operate quickly and efficiently.

A database does not store data in an ordered format. The data are stored in sets called tables, which are all related by some underlying purpose. Since the data is not stored in any particular order, one would think that looking up a particular data value could take a significant amount of execution time. This could be the case, but with proper database construction, indices can greatly reduce this time.[1]

Neural Network Operation

Having a collection of tables of related information does not constitute a Neural Network. The "glue" that holds these tables together and allows the user to perform operations on the data is stored procedures. These stored procedures are SQL statements that are compiled and optimized ahead of time for rapid execution. The stored procedures will carry out the training function of the network. Keeping the code that manipulates the data stored in the database provides additional cost savings. If there is a program error that is discovered, the code only needs to be fixed in one location…the database server. There is no longer a need to recompile a custom application and proceed through the distribution of that application to the client-base.

Utilizing stored procedures also enables simplified incorporation with a web-based front end, as was shown in previous work[4] from the Instrumentation Research Lab. This could yield the biggest cost savings of the whole project. Instead of using an office full of desktop computers each running a client application to communicate with the server, each computer need only have a web browser. Through the web browser, one can manipulate data and produce reports with the need for custom software. This eliminates distribution of new versions of client applications.

Entire System

Through making decisions that would enable the entire system to address the concerns that were addressed earlier we have built a system that is robust and easy to use. Since the database can be accessed through the WWW front end, it eliminates the need to distribute custom applications and train clients to use new and unfamiliar programs. The Web front end supports many simultaneous users from a wide geographic diversity. Since the database server is typically a powerful machine with extra memory and fast storage, and it utilizes techniques like indices and precompiled stored procedures, the system performs at an acceptable speed. Finally, we have chosen the Neural Network that seems to have simplified calculations. These simplified calculations, that also use simple data types (integers), help to further conserve resources on the server. All of these factors together help to address the initial concerns and produce a satisfactory solution to the original problem.

The Neural Network can be used for many different purposes. Most of the typical uses for neural networks involve discovering unknown underlying functions in data sets. This system will be used in corporation with other laboratories at the University of Maine to evaluate thin film gas sensors. These sensors are used in a multidimensional array to determine specific gas concentration in a gas mixture. Since the information from the sensors is noisy and unstable, neural networks are needed to help separate the information from the noise. This CMAC system will further help these gas sensor development efforts by facilitating easier sharing of processed information via the WWW and other internet technologies.

III. Conclusion

While still in the early in development, all indications are that this will prove to be a useful system for engineers and database users alike. We have taken some of the cryptic, custom applications away from the Neural Network package and replaced them with reusable database stored procedures which can be utilized by many users simultaneously. While reducing some of the complexity of the system, we have also provided a foundation for easy data sharing through the WWW front end.

This project is part of Graduate Research at the Instrumentation Research Lab at the University of Maine where students and Faculty work cooperatively to promote use of classroom ideas.

Bibliography
1. Silberschatz, A., Korth, H.F., Sudarshan, S. Database System Concepts Third Edition, McGraw-Hill, 1999.
2. Miller, W.T., Glanz, F.H., and Kraft, L.G., "CMAC: An Associative Neural Network Alternative to Backpropagation", *Proceedings of the IEEE*, Vol. 78, pp. 1561-1567, October, 1990.
3. Miller, W.T., and Glanz, F.H., "Cerebellar Model Arithmetic Computer," Fuzzy Logic and Neural Network Handbook, McGraw-Hill, Chapter 26, 1996.
4. Amos, M., and Segee, B., "Enterprise-wide Data Gathering and Reporting System," ASEE Conference Proceedings, 2000.
5. Soukup, R., Inside Microsoft SQL Server 6.5, Microsoft Press, Redmond Washington, 1997.
6. Bajaria, P., "Calibration of Solid State Gas Sensors Using Cerebellar Model Arithmetic Computer Aritificial Neural Networks," M.S. Thesis, University of Maine, 1996.
7. Segee, B., "Characterizing and Improving the Fault Tolerance of Artificial Neural Networks," PhD Dissertation, University of New Hampshire, 1992.
8. Lippmann, R.P., "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, April 1987, pp.4-22

MICHAEL AMOS
Michael D. Amos is currently a graduate student at the University of Maine pursuing a Master's degree in Computer Engineering. Mr. Amos received a Bachelor's degree in Computer Engineering with an additional major in Electrical Engineering from the University of Maine in 1999, and an Associates of Applied Science degree in Electromechanical Technology from Central Maine Technical College in 1991.

BRUCE SEGEE
Bruce E. Segee is an Associate Professor of Electrical and Computer Engineering at the University of Maine. His research interests include Instrumentation, Automation, and Intelligent Systems. He is the Director of the Instrumentation Research Laboratory and a Member of the Intelligent Systems Group at the University of Maine. His work focuses on real-world deployable systems for use in manufacturing environments. Dr. Segee received his PhD from the Department of Electrical and Computer Engineering.