A NEURAL NETWORK APPROACH TO FEATURE RECOGNITION ALONG CARTOGRAPHIC LINES

James E. Mower
Department of Geography and Planning
University at Albany
Albany, New York 12222
Bitnet: jem97@ALBNY1VX

ABSTRACT

The purpose of this paper is to suggest that models of cartographic lines are well suited for representation within neural network data structures and architectures. The use of such representations will likely improve the performance of automated line simplification routines that apply specific simplification techniques to recognized patterns. Buttenfield has discussed several methods for segmenting and describing variation in cartographic lines and has developed a procedure to recognize line features automatically (1986, 1987). Pawlicki (1988) introduces an architecture for the storage and reference of model memories based on neural network, parallel distributed, and connectionist architectures. This paper will discuss the implementational issues and expected performance statistics associated with the application of Buttenfield's line recognition procedure to sequential and connectionist processing environments.

INTRODUCTION

Line generalization involves the systematic removal of unwanted detail from the visual display of a linear feature. In digital computing environments, line generalization is performed to satisfy several constraints. First, the visual complexity of a displayed line must be made appropriate for the scale of presentation. Additionally, the digital representation of the line must support efficient storage and processing operations (Douglas and Peucker, 1973).

APPROACHES TO AUTOMATED LINE FEATURE CLASSIFICATION

Buttenfield (1987) argues for the selection of tolerance values for line generalization procedures based upon the geomorphic structure of the feature represented by the line as well as the scale at which the line is displayed. The selection of appropriate tolerance values is thus dependent upon the ability to recognize specific structures from digital line files.

The paper will examine the performance characteristics of automated line generalization routines in sequential and parallel processing environments. Buttenfield's technique for identifying the structure of cartographic lines will serve as the basis for procedures for both processing environments. The author will compare the expected performance of the two versions and discuss the computational issues associated with their implementation.

Buttenfield's Technique for Identifying the Structure of a Cartographic Line

Buttenfield uses a two-step procedure to classify features isolated in digital line files. Individual features along a line are first isolated and assigned to a category on the criterion of maximum similarity to the greatest number of measures. In the second step, the percentage of features along a line falling within each category is found. For each category, the percentage for a known line is subtracted from that of the sample. This value is squared and summed with a running total, providing a least squares calculation. The line is then assigned to a category by virtue of its proximity to a known line in the cluster space.

LINE RECOGNITION FOR SEQUENTIAL COMPUTING ENVIRONMENTS

A sequential version of Buttenfield's classification technique is summarized in Figure 1.

```
For each unknown line:
    For each feature along the unknown line (FAUL):
        For each feature category:
            For each variable:
                Find z value for FAUL;
        Find the feature category of FAUL by minimizing z scores;
        Note that another feature has been added to the feature
      category;
    Find percentage of total features in each category;
    For all line categories:
        For each feature category:
            Square the difference of the feature percentages for
          the line category and the unknown line and add to the
          running sum;
        If the squared difference is the minimum so far, record it;
    Assign the unknown line to the line category with which it has the
    minimum squared difference.
```

Figure 1. Buttenfield's procedure for the automatic identification of cartographic lines

If it is assumed that the numbers of feature variables, feature categories, and line categories are constant, the expected running time of this procedure will grow with the product of the number of lines to be analyzed and the number of features found along each line. Of course, increasing the values of the constants will also increase the expected running time of the procedure.

LINE RECOGNITION MODELS FOR PARALLEL COMPUTING ENVIRONMENTS

Neural Networks

Feldman and others (1988) note that the time required for people to correctly identify a picture is on the order of one hundred time steps, where a time step is measured by the time required for a neuron to process its input (several milliseconds). They argue that the throughput achieved by the brain is a product of massively parallel processing with a high degree of connectivity between individual neurons. Models of such neural systems are referred to as neural networks.

The neural network procedure for line recognition that will be presented below will be expressed specifically as a connectionist model. A connectionist model consists of a network of units, sites, and links (Feldman and others, 1988; Rumelhart and others, 1987). In such a network, each unit, site, and link can compute a function of its input. Units possess an activity level ranging from arbitrarily negative to arbitrarily positive values. Links establish computational

relationships among units through sites. A unit x has n links associated with it, where each link serves as input to the unit through a site. At each processing step, the activity levels of the units are computed simultaneously as a function of their inputs and previous states. Weighting functions along links control the mutual influence of units upon one another.

Pawlicki (1988) presents a top-down connectionist architecture for rapid model indexing in computer vision systems. In a top-down system, the output of a lower level unit is not allowed to affect a higher level unit. Pawlicki's architecture extracts features (items) from an input image, compiles them into an item list, and selects configurations of items from the item list to match known patterns held in an internal model (objects). The spatial relationships between the pairs of objects are held in a relational array. The contents of the relational array are matched against patterns held in the memory model (state space memory vector) using content-addressable memory operations. Hopfield (1982) and Rumelhart and others (1987) refer to content-addressable memory as the ability to access entire memory items (in Pawlicki's system, an item held in the state space memory vector) from a partial description of their contents. Pawlicki notes that neural networks have been shown to perform content-addressable memory operations in a few time steps. The architecture supports multiple canonical views, allowing the referencing of features over multiple viewing angles.

Line Recognition and General Vision Problems

The problem of recognizing lines in an input map image is more constrained than that of recognizing representations of three-dimensional objects in two-dimensional scenes. First, only linear features are referenced. Second, assuming that input map images are constructed using orthogonal perspective exclusively, the storage of multiple canonical views of lines in memory may be abandoned in favor of the storage of scale dependent representations of lines.

A CONNECTIONIST PROCEDURE FOR LINE RECOGNITION

A connectionist procedure for line recognition will now be developed using the syntax and terminology of Pawlicki's architecture. It is assumed that line features have been extracted from an input map image using Buttenfield's raster algorithm (1984) and that the extracted features have been processed to obtain values for their length, width, bifurcations, and other indexing variables.

The connectionist implementation of Buttenfield's line identification procedure is relatively straightforward. The type of input to a function will vary with the level of processing. At the feature category level, unit sites will calculate z-scores between feature descriptor variables for features (input) and feature categories held in the model memory. The activation level of a feature will be a function of the similarity of its measures to those held in a feature category. Slots in the feature category list will claim features with which they have high activations. To prevent features with low activation levels from being claimed by a feature category, the minimum level of activation required to associate a feature with a feature category will most likely be determined by a non-linear threshold function. At the line category level, units will perform least squares calculations between the percentages of features claimed by each feature category (input) and the feature configurations for known lines held in the state space memory vector. The unit having the lowest least squares statistic will be selected as the representative line.

Programming Issues

The connectionist procedure for line recognition outlined above may be programmed on a Connection Machine or on one of several connectionist simulators. The author is currently developing an implementation on the Rochester Connectionist Simulator (RCS). RCS simulates a massively parallel computer architecture on a sequential processing machine. RCS is written in the C programming language for the Unix operating system. The current version of the simulator will support a network of approximately 250,000 links on a Sun 3 workstation.

At each level in a network, units, sites, and links are associated with processors capable of computing functions on their inputs. Users may select functions from a system library or define their own. The output of a function is referred to as the level of a unit's activation.

If correlations are used to identify lines, the operation of the network will be more complex. At the feature category level, the unit sites will calculate correlations between feature descriptor variables for features (input) and feature categories held in the model memory. At the line category level, the unit sites will calculate correlations between selected features from the feature list (input) and feature configurations for known lines held in the state space memory vector. The value of the correlation will become the activation of the unit at the lower level. The unit at the lower level having the highest activation will claim the slot for the higher level in winner-take-all competition. At the line category level, low correlations will cause the system to search for new configurations of features. Low correlations at the selected feature level will cause the system to search for new features to fill the slots in the selected item list.

COMPARING THE SEQUENTIAL AND PARALLEL PROCEDURES

The expected performance of a connectionist system will depend on the number of processors and their degree of connectivity. Connectivity can be stated in terms of the fan-in and fan-out values of units. Fan-in is the number of units that directly affect a given unit. Fan-out is the number of units directly affected by a given unit (Rumelhart and others, 1987). Feldman and others (1988) note that fan-in and fan-out is typically large in animal brains (over 1,000) but low in conventional computer chips (6 or less). RCS will allow the construction of a network having a maximum of approximately 2,000 units, each having a fan-out of 100.

RCS allows the user to control the degree of fan-in and fan-out for the units. If z-score functions are assigned to sites, a large number of z-score calculations can be performed during a single time step. If 100 features compete for 10 positions on the selected feature list at some point during processing, 1,000 z-scores can be computed simultaneously by processors at unit sites. In such a case, the units at the selected feature level would have a fan-out of 100 and units at the feature level would have a fan-in of 10.

A Sample Problem

Table 1 summarizes the expected performance of the sequential and connectionist implementations of Buttenfield's line identification procedure for a sample problem. For the problem, it is assumed that 10 features have been extracted from a given line and that the state space memory vector holds descriptions of 4 lines (feature configurations) and 9 feature categories.

Using the sequential procedure in Figure 1, and assuming that a z-score computation can be performed in one time step, 450 time steps will be required to calculate z-scores for 5 line descriptor variables. 10 additional steps are required to find the maximum similarity of each feature to each feature category. 36 time steps will be required to perform all of the least squares calculations assuming each requires one time step. 1 step will be required to assign the line to a line category.

Using the connectionist procedure, 5 time steps are required to perform all the z-score calculations, one for each feature descriptor variable, assuming that each unit at the feature level is linked to all the feature categories. 1 step is required for slots at the feature category level to claim features at the feature level. 1 step is required for units at the line category level to compute the necessary least squares calculations. 1 step is required to assign the line to a line category.

Function	Connectionist Steps	Sequential Steps
Z-score	5	450
Maximum similarity	1	10
Least squares	1	36
Line identification	1	1
TOTAL STEPS	8	497

Table 1. Time steps required by the sequential and connectionist implementations of Buttenfield's cartographic line identification procedure

DISCUSSION

Connectionist implementations of automated line recognition procedures have two attractive properties. First, they are expected to terminate in far fewer time steps than are sequential implementations. Second, line recognition appears to integrate well with connectionist models of human vision systems.

Connectionist architectures may be suitable environments for developing other cartographic expert system procedures. The author is currently exploring the use of RCS to model cartographic name placement. Although the development of connectionist procedures may precede the availability of compatible hardware, the expected performance of connectionist architectures for cartographic problems justifies its exploration.

REFERENCES

Buttenfield, B.P. 1984, Line Structure in Graphic and Geographic Space: unpublished PhD Dissertation, University of Washington.

Buttenfield, B.P. 1986, Digital Definitions of Scale-Dependent Line Structure: Proceedings: Auto Carto London, pp. 497-506, London: Auto Carto London Ltd.

Buttenfield, B.P. 1987, Automating the Identification of Cartographic Lines: The American Cartographer, Vol. 14, No. 1, pp. 7-20.

Douglas, D.H. and T.K. Peucker 1973, Algorithms for the Reduction of the Number of Points required to Represent a Digitized Line or its Caricature: The Canadian Cartographer, Vol. 10, pp. 112-122.

Feldman, J.A., M.A. Fanty, N.H. Goddard, and K.J. Lynne 1988, Computing with Structured Connectionist Networks: Communications of the ACM, Vol. 31, No. 2, pp. 170-187.

Hopfield, J.J. 1982, Neural Networks and Physical Systems with Emergent Collective Computational Abilities: Proceedings of the National Academy of Sciences, USA, Vol. 79, pp. 2554-2558.

Pawlicki, T. 1988, A Neural Network Architecture for Rapid Model Indexing in Computer Vision Systems: Proceedings of SPIE--The International Society for Optical Engineering, pp. 352-359, Bellingham, WA: SPIE--The International Society for Optical Engineering.

Rumelhart, D.E., J.L. McClelland, and the PDP Research Group 1987, Parallel Distributed Processing, Volume 1: Foundations, Cambridge, MA: The MIT Press.