Vehicle Steering Control Using Modular Neural Networks

Michael Olsen Darter
California State University, Sacramento
michael@olsendarter.com

V. Scott Gordon
California State University, Sacramento
gordonvs@ecs.csus.edu

Abstract

The RoadView^{TM} project at the Advanced Highway Maintenance Construction Technology (AHMCT) research center seeks to improve the safety and efficiency of snow removal by providing information to the driver using an in-vehicle computer. The calculation of future vehicle lateral position is achieved with cooperative modular artificial neural networks, trained using data generated from a known, but somewhat slow, mathematical model. The performance of a single monolithic neural network is compared against a cooperative modular neural network trained with uniform matching criteria. Additionally, an algorithm to calculate a best achievable matching criterion for each network is described and the best achievable matching criterion is combined with a modular network partitioning scheme. The use of cooperative modular artificial neural networks reduces mean error between 46% and 55% compared with the single network.

1. Introduction

1.1 RoadView^{TM} SnowPlow Application

The Advanced Snowplow System (RoadView) is an intelligent vehicle research project at the Advanced Highway Maintenance Construction Technology (AHMCT) research center. The RoadView project seeks to improve the safety and efficiency of snow removal by providing information to the driver. The system consists of an in-vehicle computer connected to an LCD screen used by the driver. The first generation system was installed in a CalTrans 10 wheel snowplow [1]. The in-vehicle computer is connected to various sensors—a digital compass, GPS receiver, steering wheel position encoder, magnetometer, and are integrated to provide driver steering guidance using auditory and visual feedback.

In low visibility conditions, steering guidance is important for lane keeping, improves driver efficiency and enables the snowplow to hug road edges, guardrails, and other obstacles while simultaneously reducing snowplow and obstacle contact.

Figure 1 shows the RoadView display as seen by the driver. The solid rectangle in the lower left of the display represents the current vehicle position within the lane. The two white vertical lines represent lane edges. The bright square near the top of the display represents the future (predicted) position of the vehicle. Current and future lane position is shown in figure 2.

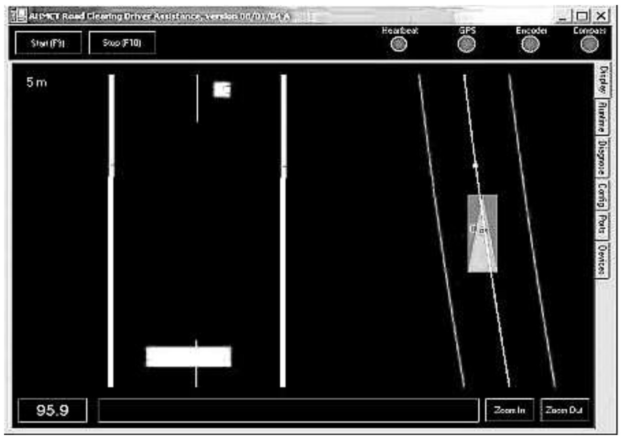

Figure 1 — RoadView Application Display

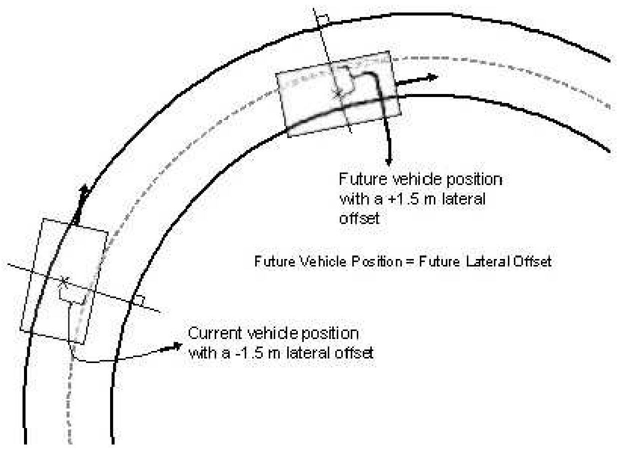
Future vehicle position with a +1.5 m lateral offset
Future Vehicle Position = Future Lateral Offset
Current vehicle position with a -1.5 m lateral offset

Figure 2 — Future and Current Vehicle Lateral Position

Future vehicle position is calculated using a system which performs a complex set of nonlinear computations [4] - referred to as the *oracle*. The oracle has seven inputs: vehicle speed, road curvature radius, heading, lateral offset, steering angle, yaw rate, and a road shape indicator. The driver's task is to keep the future position indicator in the center of the lane. Turning the steering wheel moves the indicator to the left or the right. As long as the driver keeps the indicator near the center of the lane within the display, the vehicle will remain in the center of the lane.

1.2 Problem Description

In the present RoadView system, the screen is updated 10 times a second. The oracle is computationally intensive relative to RoadView's calculation cycle time budget. Ideally, a low power embedded processor would be sufficient to run the RoadView application. The goal of this project is to replicate the effective computation of the oracle with faster speed, by using a neural network.

Feedforward multilayer perceptron neural networks are function approximators and can be trained to represent nearly any function [7]. The proposed solution uses multiple cooperative modular neural networks to calculate future vehicle lateral position. Each of these modular networks is responsible for a decomposed portion of the input domain. The input domain contains five inputs (see table 1) and is disjointedly partitioned. Each modular neural network is independently trained using a single partition. Together, all of the independently trained modular neural networks provide a solution for the full input domain.

1.3 Modular Neural Networks

In the 1990s researchers used multiple network solutions to solve a variety of problems. It is useful to distinguish between networks arranged in *ensemble* and *modular* combinations. Each individual network in an *ensemble* works to solve the entire problem. Different networks in the ensemble might use different solution strategies, methods, or different input data (e.g. data filtered differently). Results from multiple ensemble networks are combined using averaging and other statistical methods. Benefits of using multiple ensemble networks are better generalization and increased reliability through redundancy [10]. By contrast, *modularity* uses decomposition for problem solving [6][8]. Each network is responsible for solving a portion of the decomposed problem. That is, each network is a specialist at solving a particular portion of the problem.

Module results may be combined cooperatively, competitively, supervisory, or sequentially [10]. A cooperative combination uses unqualified results from each module. Advantages of using a modular approach include superior solutions and the ability to solve otherwise intractable problems [2]. Another useful distinction in multi-network solutions is between decomposition and replication. Modular systems use decomposition and ensembles do not. Both ensemble and modular systems may use replication. This project uses a cooperative modular arrangement. Each modular network contributes equally to the final solution. All modules use the same network structure (replication).

Lideng [8] used multiple neural networks to model the viscosity of an atmospheric column with 6 inputs and 1 output. Fu [3] used a divide and conquer scheme to generate a self-growing collection of modular networks, by recursively dividing into 2 partitions until the entire domain was well learned. Haecker [5] asserts that neural networks are appropriate for control applications.

2. Neural Network Solution

Many different models of neurons and networks are available [9]. We use feedforward networks with a single hidden layer, and backpropogation training.

2.1 Single Neural Network Solution

We first used a monolithic feedforward network with 1 hidden layer and backpropagation training. There were 5 inputs and 1 output. The oracle requires seven inputs. The first five of these inputs were used as neural network inputs (see table 1). The 6th input (current vehicle yaw rate) and 7th input (road shape indicator) were not used. A straight road was approximated with a very large roadway radius.

The range for each neural network input is shown in table 1 below.

Input	Input Description	Unit	Range	Oracle's Range
1	Vehicle heading	Degrees	-90 to 90	-90 to 90
2	Vehicle lateral position	Meters	-4 to 4	-4 to 4
3	Steering wheel angle relative	Degrees	-270 to 270	-2,160, +2,160
4	Radius of road curvature	Meters	(-1000,-50) (+50,1000)	-10000, 10000
5	Vehicle speed	Miles/hr	0 to 36	>= 0

Table 1 — Future Lateral Offset Calculation Inputs

To generate network training cases each input was evenly divided into 4 data points. For example, vehicle speed ranges from 0 to 36 mph, so 4 training values for speed were generated: 0, 12, 24, and 36. The total number of training cases was therefore 1024 (4^5), which covered the input domain. A single training case was considered solved if the output calculated by the oracle and neural network were within a specified tolerance of each other, referred to here as the *criterion value*.

2.2 Modular Neural Network Solution

In a modular network, each network is solely responsible for a particular portion of the input space. The complete problem is solved by using all of the modular networks' outputs, as shown in figure 3.

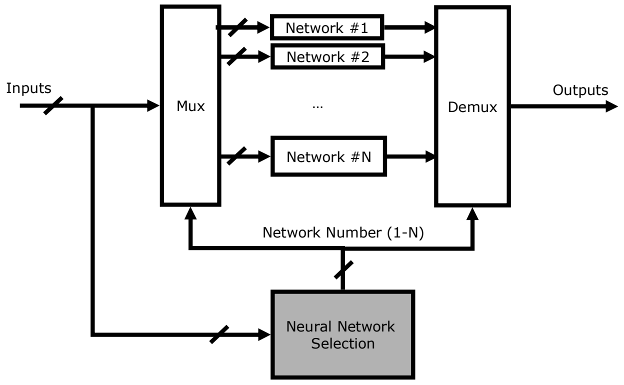

Figure 3 — Modular Neural Network Schema

The first step in setting up the scheme in figure 6 is to partition the problem input domain. Input variables are partitioned into equal sized intervals. For example, input #1 (vehicle speed) ranges between 0 and 36. It was divided into 4 uniform intervals: [0,9], [9,18], [18,27], and [27,36]. Each network input was similarly divided. After the input domain is partitioned, the task of associating an arbitrary set of inputs with the appropriate modular network is performed, shown by the gray box in figure 3.

2.3 Modular Neural Networks with Best Achievable Criterion

The modular approach uses multiple independently trained neural networks. Ideally, network training maximizes accuracy, which implies a need to minimize each network's matching criterion. If a network's matching criterion is too small, the backpropagation training algorithm will not converge. If it is too large, the resulting network loses accuracy. To determine the best achievable network criterion, the algorithm shown in figures 4 and 5 is used.

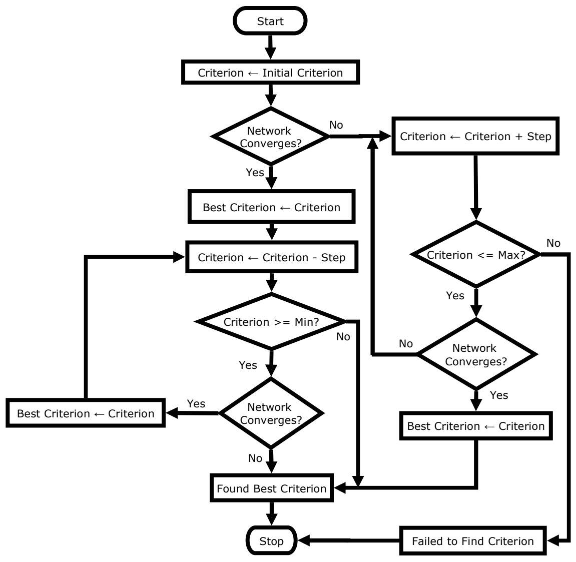

Figure 4 — Best Achievable Criterion Algorithm

The modular partitioning method shown in Figure 5 was used to solve the 5 input future vehicle lateral offset problem. In steps 1-4, each input is separately partitioned and tested. In steps 5-6, the best partitions for each input are combined. In steps 7 and 8 a tournament is used to select the final best combination.

Step	Action
1	Select the 1st input out of N inputs.
2	Partition selected input into 1..P partitions, keeping remaining inputs unpartitioned.
3	For each of the partition sets in step 2, calculate the mean error for each.
4	Perform steps 2 and 3 for all remaining inputs. Go to step 5 when done.
5	For each input, note the 2 (B=2) scenarios with the lowest mean error. Note the corresponding number of partitions for each input.
6	Create a scenario for each combination (cross product) of the partitions for each input in step 5. There should be B●N scenarios.
7	Test the scenarios created in step 6.
8	Choose the scenario from step 7 with the lowest mean error.

Figure 5 — Modular Partitioning Method

2.4 Scenario Determination Using the Partitioning Method

The input domain was partitioned using the method discussed in section 2.3. Each of the 5 inputs (N=5) was partitioned into 1 to 8 (P=3) partitions. The number of partitions for the best two scenarios (B=2) for each input were retained. This resulted in 32 (2^5) scenarios. These 32 scenarios were tested and the one with the lowest mean error was selected as the best solution.

3. Results

3.1 Experimental Method

Each input was divided into 17 test points, resulting in 1,419,857 (17^5) test cases. An error value is calculated for each test case as the absolute difference between the value calculated by the oracle and the neural network output. A number of metrics were then used during testing to characterize the accuracy of solutions. These are discussed below:

- Mean error
- Standard deviation of error
- % of cases with error 61cm or less. Measures the proportion of test cases that have high accuracy.
- Percent of test cases with error of 3 meters or more. Measures the proportion of cases with poor accuracy.
- Mean criterion value. a value ranging from 0 to 1 that is both a neural network training parameter and an indicator of network uniformity and accuracy.
- Maximum error: Used to evaluate the severity of the worst case error in a solution.

The project progressed in 4 phases, discussed below.

Phase	Description	Criterion Method	Mean Error (meters)	Improve-ment from Phase 1 (%)
1	Single NN	Best	1.942	N.A.
2	Multiple NN	Static, Uniform	1.684	13.2
3	Multiple NN	Best Achievable	0.997	48.7
4	MultipleNN + Partition Method	Best Achievable	0.875	54.9

Table 2 — Project Phase Results

3.2 Phase 1 Results — Single Neural Network

Table 3 shows parameters and results for phase 1. Determination of the best (smallest) criterion was by trial and error. An arbitrary high (poor) criterion was selected, which resulted in fast network convergence. The criterion was decreased until the network no longer converged within the 750k iteration limit. The smallest (best) criterion found was .20.

Training cases were developed using the ranges specified in table 1. Each input range was used to generate 4 data points per input. The total number of training cases was 1024 (4^5). Mean error was 1.942 meters (6.37 ft). Lane width is 12 ft, so the phase 1 mean error of 1.942 m is 53% the width of a lane.

Phase 1 Parameter	Value
Number of neural networks	1
Mean Error, meters, 17^5 test cases	1.942
StdDev of Mean Error, meters	1.772
Criterion Determination Method	Best
Criterion value	.20
Number of test cases	17^5
Number of training cases	4^5
Training iterations limit	750k
Test cases with error 61 cm or less, %	18.99
Test cases with error of 3 m or larger, %	17.7
Maximum error, meters	18.60
Training time, mins	2
Solution Calculation Time (milliseconds)	.29

Table 3 — Phase 1 Parameters and Results

Figure 6 shows the effect the backpropagation training iteration limit has on mean error. As the iteration limit increases beyond 3 million, the best achievable criterion decreases. However, mean error increases. For example, for a limit of 50 million iterations, mean error is 6.57 m and the criterion is .13. A higher mean error and lower criterion indicates a network is overtrained—it learned training cases better (lower criterion), but failed to generalize between training points, resulting in a higher mean error.

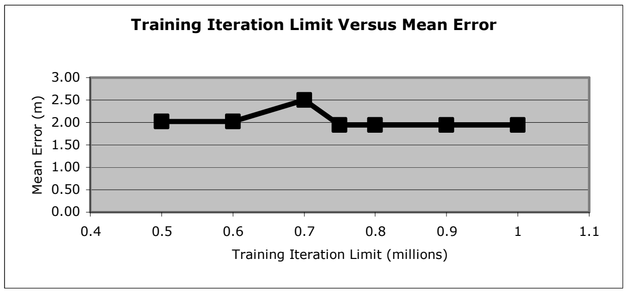

Figure 6 — Effect of Iteration Limit on Mean Error

3.3 Phase 2 Results — Modular Neural Network

Phase 1 network and neuron structures were also used in phase 2. Table 4 shows parameters and results.

Phase 2 Parameter	Value
Number of neural networks	120
Mean Error, meters, 17^5 test cases	1.684
StdDev of Mean Error, meters	1.869
Criterion Determination Method	Static
Criterion value, for all networks	.14
Number of test cases	17^5
Number of training cases	$120 \bullet 4^5$
Training iterations limit	750k
Test cases with error 61 cm or less, %	29.90
Test cases with error of 3 m or larger, %	14.86
Maximum error, meters	18.88
Total Training time, mins	8
Solution Calculation Time (milliseconds)	.38

Table 4 — Phase 2 Parameters and Results

The 5 inputs were divided into partitions: 5 (heading), 2 (current lateral offset), 2 (steering angle), 3 (road radius), and 2 (speed). This resulted in 120 neural networks (5●2●2●3●2). This partitioning scheme was arrived at through trial and error. All networks in this phase used a uniform criterion value of .14. This is the smallest criterion that allowed convergence for all 120 networks within the maximum iteration limit.

Mean error was reduced 13% from phase 1 (1.68 versus 1.94). Standard deviation of mean error increased 6% to 1.87 meters. The percent of test cases with mean error of 61 cm or less increased from 19% to 30%. Mean error test cases of 3 meters or more decreased from 15% to 18%. In summary, the phase 2 solution is clearly superior to the phase 1 solution.

3.4 Phase 3 Results — Modular Network Solution with Best Criterion

In this phase each neural network was trained iteratively for a best achievable criterion using the algorithm in figure 5. Best achievable criteria values ranged from .02 to .14. The mean was .051, substantially less than .14 used for all networks in phase 2 and .20 used for the phase 1. In this phase, fifteen input partitioning scenarios were tested. They produced between 2 and 405 neural networks. The 120 neural network scenario (5●2●2●3●2) used here (and in phase 2) produced the smallest mean error. Table 5 shows phase 3 parameters and results.

Mean error was reduced 49% from phase 2 (1.94 to .997 m). Standard deviation of mean error decreased 7% to 1.729 meters. The percent of test cases with mean error of 61 cm or less increased from 30% to 66%. Mean error test cases of 3 meters or more decreased from 18% to 10%. The phase 3 solution is clearly superior to the phase 1 and phase 2 solutions. Effectively partitioning the input domain was the principle difficulty encountered in this phase. However, for the 15 scenarios tested, it was not difficult to find a scenario that produced a better result than the single network solution.

Phase 3 Parameter	Value
Number of neural networks	120
Mean Error, meters, 17^5 test cases	.997
StdDev of Mean Error, meters	1.729
Criterion Determination Method	Best
Mean criterion value	.051
Number of test cases	17^5
Number of training cases	$120 \bullet 4^5$
Training iterations limit	750k
Test cases with error 61 cm or less, %	65.98
Test cases with error of 3 m or larger, %	9.51
Maximum error, meters	18.54
Total Training time, mins	317
Solution Calculation Time (milliseconds)	.38

Table 5 — Phase 3 Parameters and Results

3.5 Phase 4 Results — Modular Solution With Partitioning Method

In the previous phase mean error was reduced substantially by minimizing each network's criterion value. In this phase, the partitioning method shown in Figure 5 was used to generate and test 32 input domain partitions. Results for the best of these 32 scenarios are shown in table 6.

Phase 4 Parameter	Value
Number of neural networks	3528
Mean Error, meters, 17^5 test cases	.875
StdDev of Mean Error, meters	1.702
Criterion Determination Method	Best
Mean criterion value	.034
Number of test cases	17^5
Number of training cases	$3528 \bullet 4^5$
Training iterations limit	750k
Test cases with error 61 cm or less, %	74.68
Test cases with error of 3 m or larger, %	8.80
Maximum error, meters	18.85
Total Training time, mins	4671
Solution Calculation Time (milliseconds)	3.68

Table 6 — Phase 4 Parameters and Results

Table 6 shows the lowest achieved mean error of .88 meters. This is a 38% decrease in mean error from phase 3, and a 55% decrease in error from the single network solution. The percent of test cases with mean error of 61 cm or less increased 13% to 75%. Mean error test cases of 3 m or more decreased from 9.51% to 8.80%. The partitioning method also found another solution with nearly the same quality as the phase 3 solution, but with 53% fewer networks (56 versus 120). In summary, the partitioning method produced a number of superior solutions compared with the phase 3 trial and error partitioning approach.

How fast are the single and multiple network solutions? A single network calculates a solution in about 1/5 of a millisecond on a 3 GHz Pentium, which is sufficiently fast. Table 7 shows calculation times. Project goals related to calculation speed were met.

Phase	Number of Networks	Single Solution Calculation Time (milliseconds)
1	1	0.18
2, 3	120	0.25
4	3528	3.68

Table 7 — Solution Calculation Times

4. Conclusions

The goal of this project was the calculation of future vehicle lateral position with sufficient speed and accuracy. The solution used multiple cooperative modular neural networks to calculate future vehicle lateral position. Backpropagation training was used. An oracle was used to generate training cases. An algorithm to calculate each modular network's best achievable matching criterion was developed. An input domain partitioning method was developed that produced modular solutions that reduced mean error between 46% and 55% compared with a single network solution. All modular solutions were substantially superior to the monolithic solution. Increasing the number of networks in a modular solution generally improved the solution quality. Modular solutions were much less sensitive to greater than optimal increases in the number of training cases compared with the monolithic solution. Modular solutions also produced superior results for nearly all variations in the number of training cases. Neural network calculation speeds were very fast and uniform compared with the non-linear mechanistic vehicle location prediction algorithm. In summary, the use of cooperative modular artificial neural networks to calculate future vehicle position provided a substantially more accurate approximation compared with results from a single monolithic neural network.

5. References

[1] AHMCT RoadView Project Page (2005), http://www.ahmct.ucdavis.edu/roadview/r_mn.htm

[2] R. Anand, K. Mehrotra, C.K. Mohan, and S.Ranka, "Efficient Classification for Multiclass Problems Using Modular Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 6, No. 1, January 1995, pp. 117.

[3] H.C. Fu, Y.P. Lee, C.C. Chiang, and H.T. Pao, "Divide-and-Conquer Learning and Modular Perceptron Networks", *IEEE Transactions on Neural Networks*, Vol. 12, No. 2, pg. 250. 2001.

[4] M. Gabibulayev, B. Ravani, and T.A. Lasky, "Stochastic Modeling for Lateral Control in Snowplow Operations" in *9th World Congress on Intelligent Transport Systems*, Chicago, IL, October 2002.

[5] J. Haecker and S. Rudolph, "On Neural Network Topology Design for Nonlinear Control", *Proceedings SPIE Aerosense 2001 Conference On Applications and Science of Computational Intelligence IV*, Orlando Florida, April 16th 2001.

[6] B. Happel and M.J. Murre, "The Design and Evolution of Modular Neural Architectures", *Neural Networks*, Vol. 7, No. 6/7, pp 985-1004. 1994.

[7] K. Hornik, M. Stinchcomb, and H. White, "Multilayer Feedforward Networks are Universal Function Approximators, " *IEEE Transactions on Neural Networks*, Vol 2, No. 5, pp. 359-366. 1989.

[8] P. Lideng, M. Junying, and Z. Yuning, "The Application of Multiple Neural Networks in Software Instrument", Beijing University of Chemical Technology, http://www.paper.edu.cn/scholar/download.jsp?file=panlideng-5

[9] D.M. Skapura, *Building Neural Networks,* ACM Press, New York, 1996.

[10] A.J. Sharkey, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, Springer-Verlag New York, 1999.