NEURAL NETWORK-BASED SENSOR SIGNAL ACCELERATOR

Michael C. Vogt

Energy Systems Division
Argonne National Laboratory
Argonne, IL 60439

The submitted manuscript has been created by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under Contract No. W-31-109-ENG-38 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

October 2000

Submitted to SPIE Intl. Symposium on Environmental & Industrial Sensing, Boston, MA, Nov. 5-8, 2000, sponsored by SPIE.

*Work supported by the U.S. Department of Energy under Contract W-31-109-Eng-38.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Neural network-based sensor signal accelerator

*Michael C. Vogt

Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL, 60439 USA

ABSTRACT

A strategy has been developed to *computationally accelerate* the response time of a generic electronic sensor. The strategy can be deployed as an algorithm in a control system or as a physical interface (on an embedded microcontroller) between a slower responding external sensor and a higher-speed control system. Optional code implementations are available to adjust algorithm performance when computational capability is limited. In one option, the actual sensor signal can be sampled at the slower rate with adaptive linear neural networks predicting the sensor's future output and interpolating intermediate synthetic output values. In another option, a synchronized collection of predictors sequentially controls the corresponding synthetic output voltage. Error is adaptively corrected in both options. The core strategy has been demonstrated with automotive oxygen sensor data. A prototype interface device is under construction. The response speed increase afforded by this strategy could greatly offset the cost of developing a replacement sensor with a faster physical response time.

Keywords: Neural networks, sensors, high-speed, interface, adaptive algorithm, numerical acceleration

1. INTRODUCTION

A closed [proprietary] emissions control system needed a chemical sensor with a faster response rate than what was commercially available. Development of a new sensor or sensing mechanism was too costly and time consuming, so alternatives were investigated for artificially [numerically] accelerating the sensor signal. The acceleration needed to be introduced as an interface between the sensor and the control system to allow the controller to remain unchanged.

A strategy was developed to use a sequence of neural network adaptive linear predictors [computational algorithms]. Each predictor uses the over-sampled and synchronized recent history of the sensor to predict a "time-step" into the future. Two synchronized predictors allow doubling and self-correcting of the sensor original response time; five predictors allows a quintupling, etc. Over-sampling the sensor [sampling faster than it can physically respond to changes in environment] cannot *directly* provide the values the control system needs at the rate that they are needed, but it should provide each of the synchronized predictors with slightly time-shifted inputs to generate synthetic values at the faster rate the control system *does* need.

A prototype system is in mid-development. The neural network predictor stage has already been demonstrated with sample emissions data. The synchronization stages are being completed at the time this paper was written. This paper represents an opportunity to introduce the approach, invite mid-investigation peer review and comment, and promote discussion regarding additional applications.

2. BACKGROUND

There are many control applications in industry that cannot be properly addressed at this time. While sensors for those applications exist, the existing sensors do not provide a signal quickly enough for real-time or near-real-time feedback. The following is a description of sensor response characteristics and a description of the neural network components that can be used to improve those sensor response characteristics.

2.1 Sensor characteristics

A sensor's *response time* is described as the period between changing physical conditions and a change in the sensor's (electrical) output. The sensor's *sensitivity* also comes into play by describing the amount of change necessary in the physical conditions that is needed to initiate a similar change in the sensor's output signal. Taken together, the response time and sensitivity control the output signal.

* Correspondence: Email: vogt@anl.gov, WWW website for Energy Systems Division, <http://www.es.anl.gov>

2.2 Sensor response time definition

If a given sensor's response time is N (i.e. $N = 50$ ms) and its sensitivity is S (i.e. $S = \pm 10$ ppm for a chemical sensor) then N milliseconds after an S ppm change is experienced the sensor will produce a proportional, measurable change in the generated electrical signal¹. A control system external to the sensor cannot poll the device any faster than it can physically respond because no change (or an inaccurate change) in electrical signal will be detected. If the external control system requires a feedback signal at higher sampling frequencies, then the classical solution is to develop a new sensor with different (improved) physical response characteristics. This can be very costly and often not possible.

2.3 Sensor signal conditioning

In a sensor-controller system, algorithms executed by the controller can "condition" the incoming sensor signal, performing scaling and compensating for off-sets and other numerical adjustments. If new sensors are attached to the system, new conditioning algorithms can be added to the controller's programs. Additional, more advanced algorithms executed by the controller also can perform *prediction* of the sensor's output-signal.

2.4 Signal-prediction in open and closed control systems

In a control system, such as an internal combustion engine, *predicted* values could be used to decide what adjustments need to be made to control actuators (throttles, fuel injector duty cycles, spark advance values) to minimize gaseous emissions. In an "open" control system, prediction algorithms would simply be added to the code representing the composite system controller, but in a "closed" system, the control algorithms are treated as black box and are not available for interaction of any kind. This is common for proprietary control systems, where the only data available to the engineer are output specifications a sensor must meet to match the input specifications of the closed control system. In this closed system, an interface device needs to be produced that can accept the signals from the sensor, predict their near-future values, and output a "synthetic" sensor signal for the control system to measure.

2.5 Advanced signal processing

The Argonne National Laboratory Microsensors Program has been developing an approach to systemize the introduction of "smart" behaviors and capabilities to generic sensor signal processing. The approach involves focusing on capturing and processing a sensor's response in *vector* or *profile* form as opposed to single point form. Time-based and parameter-based "features" in each sampled profile contain significantly more information regarding rate-of-change and inter-variable relationships than is available in any single sample point. These features represent a powerful mechanism to identify events and trends in a signal, and to associate those events and trends with an appropriate control action. This vector or profile processing can be embedded into an open control system or can be implemented externally to a closed system. A suite of such signal processing algorithm modules has been collected into a special-purpose computer aided software engineering (CASE) tool called SAGE: the Sensor Algorithm Generation Environment². SAGE allows a comprehensive processing algorithm to be pieced together for a wide range of sensor and instrument data types. Standard signal conditioning is augmented by statistical functions and advanced artificial intelligence based pattern recognition and analysis [using several varieties of neural networks]. The final algorithm can be exported in ANSI C/C++ for use with embedded controllers and higher performance computers³. An algorithm coded in C/C++ is well-staged for direct translation into Java for virtual machine execution.

SAGE has demonstrated that the neural algorithms can be coded onto a simple, inexpensive microcontroller for deployment. With this miniaturization of code established, other types of networks also can be shrunk and coded on a support microcontroller, specifically, *forward-predicting adaptive linear networks*.

2.6 Neural networks

While a complete description of neural network operation is beyond the scope of this paper, the brief introduction that follows is appropriate to illustrate the level and complexity of computation required for signal prediction.

One way to perform predictions is with an ADaptive LINEar Element (ADALINE) computational neural network (CNN). A neural network is an algorithm that mimics the information processing capability of the biological brain⁴. An Artificial Neural Network (ANN) accomplishes this by simulating each biological neuron with an integrated circuit as a collection of gates and transistors while a Computational Neural Network (CNN) accomplishes this through execution of a series of computer instructions.

Figure 1. Simple 5-input adaptive linear neuron

Figure 1 above illustrates a computational neuron. A neural *network* is a collection of one or more of these neurons. Each neuron is broken into two stages, the SUMMATION stage, designated with the Sigma symbol, Σ , and the ACTIVATION stage represented by δ symbol and the box with the diagonal line in it⁵. In the SUMMATION stage, the inputs p_1, p_2, p_3, p_4, p_5 (or however many are used) are each multiplied by a respective scaling weight value w_1, w_2, w_3, w_4, w_5 and summed up to produce a weighted product 'n'. [Included in this weighted sum is a constant input of '1' times a bias weight b . This improves training algorithms and allows adjustment parameters for control of the output.] This is used as the input to the ACTIVATION stage which is just an equation for a line with linear slope i.e. $a = (C1 * n) + C2$ where $C1$ and $C2$ are constants representing $C1$ = slope and $C2$ = y-intercept of a line. For the prototype $C1 = 1, C2 = 0$. This produces an output $a = n$.

2.7 ADALINE

An ADALINE is a simple neural network model with limited capabilities, but one of those capabilities is the ability to predict a changing signal's value a short time interval into the future based upon limited near-past values and a simple error-correction mechanism⁶. This type of prediction can be used to make decisions in advance of actual events. While this cannot predict stock market changes days in advance, it can predict potentially dangerous conditions in a furnace and trigger a power reduction or power shut down prior to an accident.

2.8 Sensors aided by neural networks

A very simple, limited-element adaptive neural predictor can monitor a sensor's time-based signal (basically any single-parameter signal) over a short history, and very accurately (small fraction of % error) predict its signal a short time into the future. While this short-range prediction is very accurate, it cannot look far enough into the future to be of significance to larger systems. It *can*, however, *significantly improve the usefulness of sensors by effectively accelerating their signal response* allowing much higher speed control system sampling based upon their improved predicted (and corrected) output.

3. APPROACH

3.1 Computational [numerical] acceleration strategy

Accelerating the signal through numerical processing represents a robust alternative strategy for improving the sensor's response output *without changing the sensor itself*. The goal in numerical acceleration is to produce an updated sensor output that changes faster than the sensor's own physically-limited output, and at a speed to match the control system's. This signal must not only be faster, but must also be an accurate prediction of the sensor's own lagging real output.

3.2 Computational acceleration theory of operation

The "acceleration" of the signal output starts by performing an accurate forward prediction of the sensor signal. The prediction is performed using an ADALINE CNN that uses a limited number of successive sensor past output values. If the current measured sensor output is represented by n , then $n-1$ would be one time interval earlier and similarly with $n-2, n-3$, and $n-4$ measured at uniform intervals limited by the sensors normally best response time. These values would be used to predict the value of $n+1$, one "time interval" in the future. When the actual sensor-limited sampling interval has passed and the predicted $n+1$ becomes the now sampled n , then the actual new sampled value n is used with a correction algorithm to adjust the next prediction and the cycle continues. This new value $n+1$ is still predicted as N [milliseconds] in the future, and while this represents useful information in terms of overall control strategies, it still does not provide faster response, only

future response. The *acceleration* involves taking that predicted value and calculating (interpolated) *synthetic intermediate values* and outputting them to the control system in a shorter time period than that in which the sensor responds.

Figure 2. Interpolated intermediate value

3.3 Option 1: Interpolation between points

Figure 2 above illustrates how p1, p2, ... (all prior p's) are used to predict p0. Each of the p's is sampled at a rate limited by the source (such as a sensor). P0 is exactly the same time step ahead of p1 and p1 is ahead of p2. In order to 'predict' a value sooner than p0... a prediction is made for p0 and a separate value between p1 and p0 is interpolated. This is suitable for high-speed operations where computational capability is limited but where the small error introduced by the interpolation is acceptable. The illustration shows a single value interpolated in between p1 and p0, but many values could be calculated using simple linear interpolation. The more values that are interpolated translate into faster signal acceleration.

3.4 Option 2: Synchronized CNN predictors

Additionally, if the microcontroller is fast enough, then using synchronized ADALINE CNNs can make more accurate intermediate point calculations. Using the same sample case, if five ADALINE CNNs were initiated 10 ms apart, then each one would gather a different set of n, n-1, n-2, n-3, n-4 samples and each would predict, respectively, n+0.2, n+0.4, n+0.6, n+0.8, and n+1. In this case, no interpolation would be required and the effective response function approximation would be even more accurate than the single prediction + linear interpolation strategy.

Figure 3. Synchronized predictors

Figure 4. Multiple (two) linear networks

In the above diagram, Figure 3, samples p1-p5 are used to predict p0 and q1-q5 are used to predict q0. The two neural networks (Figure 4) operate completely independently and are executed sequentially or in parallel as hardware allows. New input readings are sampled from the incoming signal. The new readings are gathered at a rate limited by the sampling

analog-to-digital converter (A/D) or by the signal source device, whichever is slower. In this example, the P network and the Q network predict alternating values that are used to control an additional programmable signal. This synthetic signal is what an external 'closed' control system would see, at a rate that is updated twice as often as the actual source signal.

Figure 5. Adaptive correction of weights

The error between the current sampled signal value (p_1) and the previous iteration's predicted value (a) is calculated and used along with the signal values (shown in Figure 5) to adaptively correct the weight values w_1 - w_5 . This error correction is completed each iteration and has been demonstrated to achieve small fraction of a percent error within a few iterations and then to maintain that level of error tracking the changing input. Dramatic changes in input can increase the error marginally, but only temporarily until the network adaptively corrects.

This same adaptively correcting algorithm is categorized as an adaptive filter. The prototype implemented this adaptive filter as a neural network, but the collection of instructions is effectively just a group of algebraic calculations and readily can be implemented without the framework of a neural network and can be implemented using discrete circuitry as dedicated hardware instead of software instructions.

The diagram/flowchart shown in Figure 6 below illustrates the major components of the earlier example of two predictors. The first stage is the actual synchronizer where the incoming signal is A/D sampled. The P network is normally executed first calculating p_0 , which is copied to the synthesized output controller module. The q_0 output is copied alternately to the output controller module. The overall effect is that the incoming signal, normally available with an updated reading at only interval m seconds, is now updated and available at interval $m/2$ seconds, twice as fast. Five synchronized networks executed sequentially would produce a 5X acceleration or a rate of $m/5$ seconds, limited by the computational and data acquisition capability of the system.

3.5 A demonstration model structure

The original goal of the work was to speed up the response time of an NO_x or O₂ gas sensor so that an external control system could make system-wide adjustments more frequently. A Bosch automotive oxygen sensor was sampled over the duration of a road course and provided the raw data used. The approach development model was separated in to two parts. The first part modeled the ADALINE CNN using MathWorks® MATLAB™ Neural Network Toolbox and produced data describing the behavior of the network and accuracy of the predictions and interpolations. The second part involved coding the same type of ADALINE CNN on a very inexpensive Microchip™ PIC microcontroller to benchmark the execution speed and estimate the complexity possible with inexpensive systems. A PIC 16C74 running at 20 MHz (200 ns instruction cycle) with 4K bytes of program space, 368 bytes of RAM, A/D input and PWM output was used. This is considered a typical industrial microcontroller unit (MCU) and represents a lower limit to the acceleration possible using this strategy.

The more complex synchronized predictor option is still being developed at the time of this paper. More merit was seen in coding the synchronized predictors along with several other signal processing algorithms as part of this same 'interface' device, allowing time-weighted averaging, proportional scaling, integral scaling, rate of change scaling (derivative scaling), as well as error correction.

Figure 6. Flow chart of major components

4. RESULTS

The numerical acceleration has been partially demonstrated in two stages. One stage established that the sampling rate could support acceptable prediction accuracy and adaptive correction. The second stage established that the algorithm could be coded onto a popular microcontroller and that it could be executed fast enough to be useful. The two stages of the prototype test were not combined in time for this paper.

4.1 MATLAB ADALINE CNN demonstrates prediction accuracy

The example sensor's response time was estimated at 50 ms, and the target control system needed an updated value in 10 ms. A 5x acceleration was required. The sensor's most recent values were stored, $[n, n-1, n-2, n-3, n-4]$, at 50 ms boundaries and a single-neuron ADALINE CNN was used to predict value $n+1$ at 50 ms in the future of n .

The single-neuron ADALINE constructed using the MATLAB Neural Network Toolbox [executing on a 200 MHz Pentium II computer running MS Windows 98 and having 128 MB RAM] required less than 2 seconds to process the 2485-point sample data set. It followed the sample data with an overall error of $> 0.2\%$ with slight increases only following significant changes in the sensor signal (such as observed during hard acceleration).

Figure 7. Actual performance curves

The chart above shows the real raw data in [solid] green and the predicted values with a [dashed] blue line. For the demonstration the predictions were made using only 1 past point, 2 past points, 3 past points, 4, 5, 6, 7, 8, 9, and 10 past points. Very good results and behavior were achieved with this data using 2, 3, 4, and 5 points. Additional points did not improve predictive capability and only required more calculations.

4.2 Microcontroller ADALINE timing and performance calculations

The prototype network used to calculate processing requirements followed the same structure as the MATLAB demonstration network and used a version of the Widrow-Hoff Least Mean Square error minimization method. The adaptive linear network consisted of a single neuron with 5 inputs, arranged in a vector $P = [p_1, p_2, p_3, p_4, p_5]$, 1 bias, $b = [b]$, and 1 output, $a = [a]$. Processing was broken into a summation stage and an activation stage. The feed-forward processing required (1) A/D conversion, (5) multiplications, and (5) additions. The learning stage required calculating the error and