Evaluating Timeliness and Accuracy Trade-offs of Supervised Machine Learning for Adapting Enterprise DRE Systems in Dynamic Environments[*]

Joe Hoffert[†]
Electrical Engineering & Computer Science, Vanderbilt University, 2015 16th Avenue S.
Nashville, Tennessee 37212, USA

Douglas C. Schmidt
Electrical Engineering & Computer Science, Vanderbilt University, 2015 16th Avenue S.
Nashville, Tennessee 37212, USA
E-mail: d.schmidt@vanderbilt.edu
www.vanderbilt.edu

Aniruddha Gokhale
Electrical Engineering & Computer Science, Vanderbilt University, 2015 16th Avenue S.
Nashville, Tennessee 37212, USA
E-mail: a.gokhale@vanderbilt.edu
www.vanderbilt.edu

Abstract

Several adaptation approaches have been devised to ensure end-to-end quality-of-service (QoS) for enterprise distributed systems in dynamic operating environments. Not all approaches are applicable, however, for the stringent accuracy, timeliness, and development complexity requirements of distributed real-time and embedded (DRE) systems. This paper empirically evaluates constant-time supervised machine learning techniques, such as artificial neural networks (ANNs) and support vector machines (SVMs), and presents a composite metric to support quantitative evaluation of accuracy and timeliness for these adaptation approaches.

Keywords: Adaptation, Machine Learning, Distributed Real-time Embedded Systems, Dynamic Environments.

1. Introduction

Emerging trends and challenges. Enterprise distributed real-time and embedded (DRE) systems manage resources and data that are vital to organizations or projects. Examples include shipboard computing environments, air traffic management systems, and recovery operations in the aftermath of regional or national disasters. These systems often adjust their operations based on their external environment. For example, search and rescue missions as part of disaster recovery operations can adjust the image resolution used to detect and track survivors depending on the resources available (*e.g.*, computing power, network bandwidth) [1].

Many enterprise DRE systems autonomically monitor their environment and modify their modes as the environment changes since manual adjustment is too slow and error prone. For example, a shift in network reliability can prompt quality-of-service (QoS)-enabled middleware, such as the OMG Data Distribution Service (DDS) [2], to change mechanisms (such as the transport used to deliver data) since some transports provide better reliability than others in some environments. Likewise, cloud computing applications, where elastically allocated resources (*e.g.,* CPU speeds and memory) cannot be characterized accurately *a priori,* may need to adjust to available resources (such as compression algorithms optimized for given CPU power and memory) at system startup. The mission(s) of the system could be jeopardized if these adjustments take too long.

[*]This work sponsored by NSF TRUST, AFRL, and JFCOM.

[†]First author email address: jhoffert@dre.vanderbilt.edu

One way to adapt enterprise DRE systems autonomically involves the use of policy-based approaches [3] that externalize and codify logic to determine the behavior of managed systems. Policy-based approaches provide deterministic response times to perform appropriate adjustments given changes in the environment. The complexity of developing and maintaining policy-based approaches for enterprise DRE systems can be unacceptably high and compromise trustworthiness, however, since developers must determine applicable policies for different environmental properties. Moreover, developers must manage the interaction of policies to provide needed adjustments.

Machine learning techniques [4, 5] support algorithms that allow systems to adjust behavior based on empirical data (*e.g.*, inputs from the environment). These techniques can be used to support autonomic adaptation by learning appropriate adjustments to various operating environments. Unlike policy-based approaches, however, machine learning techniques can automatically recognize complex sets of environment properties and make appropriate decisions accordingly.

Conventional machine learning techniques, such as decision trees and reinforcement learning, address autonomic adaptation for non-DRE systems [6]. These techniques are not well-suited for enterprise DRE systems, however, since they do not provide data-independent bounded times when determining adjustments [7]. Reinforcement learning techniques [8] explore the solution space until an appropriate solution is found, regardless of the elapsed time. Decision tree techniques have time complexities dependent upon the specific data and cannot be determined *a priori*. Moreover, decision trees may contain branches that are much longer than others, making the determination of appropriate adaptations unpredictable, which is undesirable in DRE systems.

Supervised machine learning techniques use training data to guide learning [9]. These techniques can provide constant time complexity along with perfect accuracy in determining appropriate adaptations for environments on which they have been trained (*i.e.*, known *a priori*). Techniques that trade off generality for specificity with perfect accuracy (*i.e.*, are specialized for the environments they have seen and on which they have been trained) are called "overfitted" [10], which makes the accuracy equal to policy-based approaches (*i.e.*, 100% accurate). Moreover, several techniques with constant time complexity that are not overfitted provide high accuracy for determining adaptations for environments unknown until run-time.

Some techniques provide lower response times with lower accuracy, whereas others provide higher accuracy and response times. It is hard to manage (1) overfitted and non-overfitted techniques, (2) the accuracy and response times for these constant time techniques, and (3) the trade-offs between them. Developers must empirically evaluate the techniques, combine the results of accuracy and response time manually, and determine the most appropriate technique.

Solution approach→ Integrated supervised machine learning techniques and composite metrics to guide trade-offs of accuracy and response times. This paper describes our *timely-integrated machine learning* (TIML) approach to integrating the following techniques: (1) overfitted supervised machine learning to respond perfectly to environments known *a priori*, (2) non-overfitted techniques to respond to environments unknown until runtime with high accuracy and constant response times, and (3) a composite metric to evaluate the accuracy and response time of different techniques quantitatively. TIML supports low-latency, constant-time complexity for determining adaptations to operating environments, 100% accuracy for environments known *a priori*, and high accuracy for environments unknown until runtime. We evaluate techniques for managing both response times and accuracy.

Our prior work [11, 12, 13] presented an architecture for autonomic adaptation and evaluated machine learning techniques without pinpointing the fastest response times. The work presented in this paper adds new experimental data and analysis, including fastest response times, as well as providing a new composite metric to evaluate accuracy and response time. We (1) overfit an artificial neural network (ANN) [14] (which is a technique modeled on interactions of neurons in human brains) to retain as much information about specific environment configurations and adjustments as possible (*e.g.*, greatly increasing the number of connections between input environment characteristics and output adjustments used in an ANN), (2) integrate non-overfitted ANNs and support vector machines (SVMs) [15] (which generate the boundaries between different groupings to maximize the differences between groupings and aid in classification) to provide low response times and high accuracy for environments unknown until runtime, and (3) evaluate the machine learning techniques using the *AccuLate* metric that quantitatively combines accuracy and latency. Our *ADAptive Middleware And Network Transports* (ADAMANT) framework integrates TIML with the DDS QoS-enabled middleware to ensure accurate, timely, and predictable adaptation to dynamic environments.

2. Motivating Example - Search and Rescue (SAR) Operations for Disaster Recovery

To motivate the need for integrating machine learning techniques, this section describes the challenges associated with search and rescue (SAR) operations. SAR operations are part of disaster recovery enterprise DRE systems which manage relief efforts in the aftermath of a disaster, such as a hurricane or earthquake. SAR operations help locate and extract survivors in a large metropolitan area after a regional catastrophe. SAR operations use unmanned aerial vehicles (UAVs), existing operational monitoring infrastructure (*e.g.*, building or traffic light mounted cameras intended for security or traffic monitoring), and (temporary) datacenters to receive, process, and transmit event stream data from sensors and monitors to emergency vehicles that can be dispatched to areas where survivors are identified.

Fig. 1 shows an example SAR scenario where infrared scans along with GPS coordinates are provided by UAVs and video feeds are provided by existing infrastructure cameras. These infrared scans and video feeds are then sent to a datacenter, where they are processed by fusion applications to detect survivors. Once a survivor is detected the application can develop a three dimensional view and highly accurate position information so that rescue operations can commence.

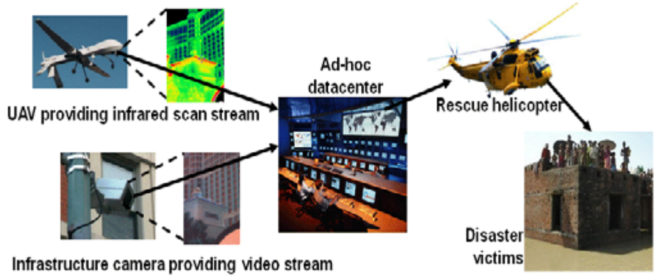
UAV providing infrared scan stream
Ad-hoc datacenter
Rescue helicopter
Infrastructure camera providing video stream
Disaster victims

Fig. 1: Search and Rescue Motivating Example

3. Key Challenges of Enterprise DRE Systems

This section summarizes key challenges that arise when developing autonomic enterprise DRE systems, such as the datacenter in the SAR motivating example in Sec. 2.

3.1. *Challenge 1: Timely Adaptation in Dynamic Environments*

Due to the dynamic environment inherent in enterprise DRE systems, application operations (such as image compression to reduce network traffic or disseminating data with timeliness *and* reliability properties) must adjust in a bounded—ideally constant time—manner as the environment changes. Operations that cannot adjust quickly and in a bounded amount of time will fail to perform adequately when resources change. For example, if resources are lost or withdrawn—or demand for information increases—operations must be configured to accommodate these changes with appropriate responsiveness to maintain a minimum level of service. If resources increase or demand decreases, operations should adjust as quickly as possible to provide higher fidelity or more expansive coverage. Manual modification is often too slow and error prone to maintain QoS.

3.2. *Challenge 2: Accurate Adaptation to Dynamic Environments*

Application operations in enterprise DRE systems must accurately adjust to changes in the environment. As changes in enterprise DRE systems occur (*e.g.*, increases in networking capability or requests for data from new senders and receivers), the system should take advantage of additional resources or provide access to additional data producers and consumers while maintaining or increasing QoS. For a given environment configuration, a most appropriate response exists and the enterprise DRE system must accurately implement adjustments to fully leverage existing resources.

3.3. *Challenge 3: Flexibility in Trading Off Accuracy and Timeliness*

Application operations in enterprise DRE systems must be able to trade-off adaptation accuracy with timeliness. The situation may demand that finding a less accurate adaptation in time is better than finding an ideal adaptation too late [16]. For example, selecting an adaptation that responds more quickly but has a lower probability of accuracy may be needed when response time is crucial (*e.g.*, failure of critical infrastructure is imminent or groups of injured survivors must be detected quickly). If perfect detection of survivors is performed too late, it may not be possible to rescue the survivors.

4. Solution Approach - Integrating Machine Learning Techniques and Composite Metrics

Timely-integrated machine learning (TIML) integrates multiple machine learning techniques to provide both (1) perfect accuracy and low response latency in determining appropriate adjustments, such as adjustments to transport protocols to support QoS in dynamic environments, for environments known *a priori* and (2) high accuracy for environments unknown until runtime. The AccuLate composite metric provides quantitative guidance for balancing accuracy and response time latency. This approach enables enterprise DRE systems to adjust to their environments autonomically and evaluate accuracy and response latency quantitatively. Moreover, we leverage techniques that provide the constant time complexity assurance needed for enterprise DRE systems.

TIML overfits ANNs to retain a high degree of information about specific environment configurations and adjustments, e.g., increasing the number of hidden nodes used in an ANN. Hidden nodes are the computational components that provide connections between the relevant properties of the operating environment (*e.g.*, CPU speed, network reliability) with the adjustments needed for those environments. As the ANN learns, it strengthens or weakens the connections between inputs, hidden nodes, and outputs to provide appropriate adjustments. Increasing the number of hidden nodes increases the level of detail that the ANN maintains. Moreover, TIML utilizes SVMs configured with different *kernels* (*i.e.*, approaches to generating additional features from the environment configurations [17]) to increase accuracy over ANNs for environments unknown until runtime. Our approach resolves the challenges presented in Sec. 3 as described below.

- *Machine learning techniques that use a static number of equations for learning* address Challenge 1 in Sec. 3.1 by providing predictable time complexities for determining appropriate adjustments. In particular, we apply overfitted ANNs for environments known *a priori* and multiple machine learning techniques for environments unknown until runtime to support enterprise DRE systems by incorporating the appropriate QoS-enabled middleware and transport protocol adjustments based on accuracy and timeliness concerns. When machine learning techniques are used in an enterprise DRE system, the time needed to make an appropriate adjustment is bounded by a constant number of equations.
- *Integrating machine learning techniques* address Challenge 2 in Sec. 3.2 by increasing the accuracy for environments known *a priori* and increasing the accuracy for environments unknown until runtime. Our approach increases the accuracy of determining appropriate adjustments by using an overfitted ANN for environments known *a priori* and integrated machine learning techniques that provide increased accuracy as compared to overfitted ANNs. Specifically, overfitting ANNs provides accuracy equal to policy-based approaches for environments known *a priori*, while non-overfitted techniques increase accuracy for environments unknown until runtime.
- *Incorporating multiple machine learning techniques and evaluating accuracy and timeliness simultaneously* addresses Challenge 3 in Sec. 3.3 by supporting multiple techniques with different levels of accuracy and response times for environments unknown until runtime and providing a composite metric to evaluate the trade-offs of these different techniques quantitatively. TIML supports ANNs and SVMs with various configurations. Integrating these ANNs and SVMs provides flexibility to support timeliness and accuracy for systems that need to balance the two concerns. Moreover, the AccuLate composite metric described in Sec. 5.3 allows quantitative evaluation of these techniques.

5. Experimental Results

The section presents the results of experiments we conducted using ANNs and SVMs to determine timeliness, accuracy, and the balance between them to show the SAR datacenter leveraging ADAMANT in selecting an appropriate transport protocol configuration for a given operating environment. The experimental input data used to train the machine learning techniques include ADAMANT with multiple properties of the operating environment varied (*e.g.*, CPU speed, network bandwidth, DDS implementation, percent data loss in the network), along with multiple properties of the application being varied (*e.g.*, number of receivers, sending rate of the data), as would be expected with SAR operations.

We collected 394 inputs from previous experiments [18] where an input consists of data values that determine a particular operating environment (*e.g.*, CPU speed, network bandwidth, number of data receivers, sending rate). We also provided the expected output to the ANNs and SVMs, *i.e.*, the transport protocol that provided the best QoS with respect to data reliability, average latency, and jitter (*i.e.*, standard deviation of the latency of network packets). An example of one of the 394 inputs is the following: 3 data receivers, 1% network loss, 25Hz data sending rate, 3GHz CPU, 1Gb network, using the OpenSplice DDS implementation, and specifying reliability and average latency as the QoS properties of interest. Based on our experiments, the corresponding output would be the NAK-based multicast protocol with a 1 ms retransmission timeout.

5.1. *Evaluating the Accuracy of ANNs and SVMs*

Our addressed the SAR accuracy requirement by first training the ANNs and SVMs on the 394 inputs mentioned above. We used the *Fast Artificial Neural Network* (FANN) library [19] as our ANN implementation due to its configurability, documentation, and open-source availability. FANN offers extensive configurability for the neural network including the number of hidden nodes connecting inputs with outputs. For SVMs, we used the libSVM library [20] due to its configurability, documentation, and open-source availability.

To determine the most accurate ANN and SVM we ran training experiments with the ANNs using different numbers of hidden nodes and SVMs with different kernels. Only one SVM, however, provided 100% accuracy for the environment configurations on which they had been trained (*i.e.*, known *a priori*). For a given number of hidden nodes we trained the ANN 10 different times. The weights of the ANN determine how strong connections are between nodes. The weights are randomly initialized and these initial values have an effect on how well and how quickly the ANN learns.

Fig. 2 shows the accuracies for the ANN configured with 3, 4, 6, and 12 hidden nodes over 10 training runs. Fig. 2 also shows the effect of random initial weights on the accuracy of the ANN since the accuracy can vary across training runs. Accuracy was determined by querying the ANN with the data on which it was trained.

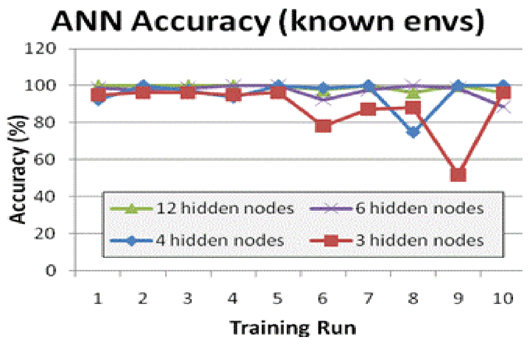

Fig. 2: ANN Accuracy for Known Environments

100% accurate classification was generated at least once with all hidden node configurations except when using 3 hidden nodes. The ANN with 12 hidden nodes provided the best accuracy across all the training runs—100% accuracy all but 3 times out of 10 which would make it more likely to provide 100% accuracy for any single training run. However, we need only a single 100% accurate classification and therefore choose the ANN with 4 hidden nodes since it has the lowest response time as shown in Sec. 5.2. No ANNs with hidden nodes fewer than 4 and only one SVM configuration provided 100% accurate classifications. We do not include the accuracy data for SVMs since the SVM response times are an order of magnitude greater than ANNs as shown in Sec. 5.2. The training data values had to be scaled from -1 to 1 to achieve 100% accuracy for the configurations in Fig. 2.

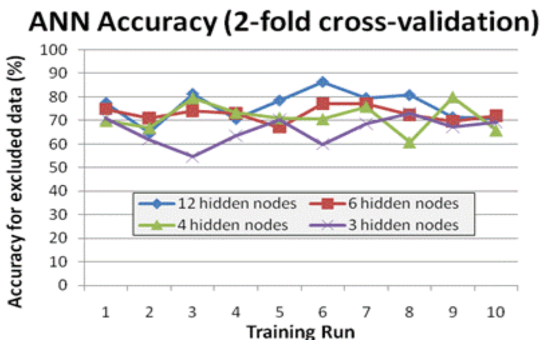

Fig. 3: ANN Accuracy for Unknown Environments

Fig. 3 and Fig. 4 present accuracy results for ANNs and SVMs respectively for operating environments unknown until runtime. The ANNs are randomly assigned initial weights for the connections between nodes which accounts for variations in accuracy across training runs. We average the accuracy results across all runs below. The accuracy for SVMs is dependent upon scaling of the input and output training data. The different scaling scenarios are presented in Fig. 4.

To evaluate accuracy with unknown environments we use 2-fold cross-validation, where 394 environment configurations are split into two mutually exclusive training and testing data sets [21]. ANNs and SVMs are trained using training data and evaluated using testing data. The highest average accuracy for ANNs across the 10 training runs is produced with 12 hidden nodes (*i.e.*, 76.09% average accuracy). The second highest average accuracy is produced with 6 hidden nodes (*i.e.*, 72.84% average accuracy). SVMs produce higher accuracies, however, (*i.e.*, 86.29% accuracy for SVMs using either the RBF or polynomial kernel).

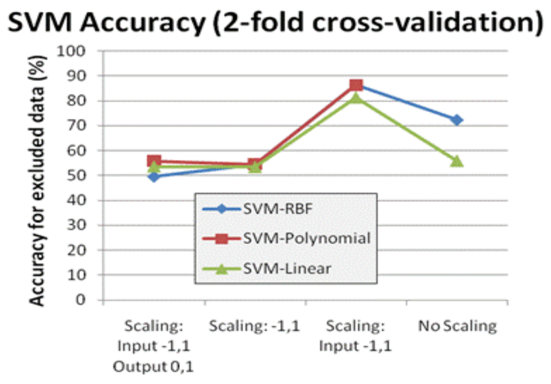

Fig. 4: SVM Accuracy for Unknown Environments

5.2. *Evaluating the Timeliness of ANNs and SVMs*

As described in Challenge 2 in Sec. 3.2, the datacenter for SAR operations requires timely configuration adjustments. This section provides timing information for ANNs and SVMs when queried for an optimal transport protocol. We used a 3 GHz CPU with 2GB of RAM running Fedora Core 6 with real-time extensions. Timeliness was determined by querying the ANNs and SVMs with all 394 inputs on which they were trained. A high resolution timestamp was taken before and after each call to the ANNs and SVMs.

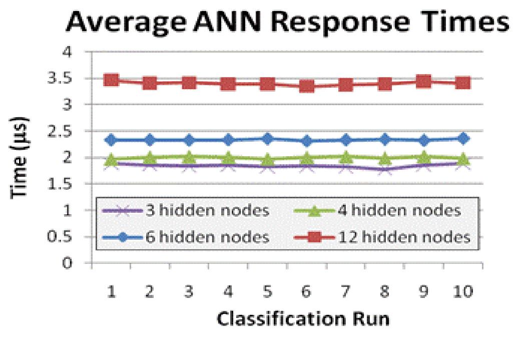

Fig. 5: ANN Average Response Times (µseconds)

Fig. 5 shows the average response times for 10 separate experiments where for each experiment we query the ANN for each of the 394 inputs. The figures show that ANNs provide timely and consistent responses. As expected, the response times using more hidden nodes are slower than response times with fewer hidden nodes. The increase in latency is less than linear, however (*e.g.*, response times using 12 hidden nodes are less than twice that using 6 hidden nodes).

Fig. 6 shows the response times for SVMs configured with different kernels and data scaling approaches. The SVM with the linear kernel tends to have the lowest response time with the polynomial and RBF kernels being the next most responsive respectively. Scaling the environment configuration input data and the transport protocol output response has an effect on the response times as well since this scaling affects the specific kernels that are created for the data.

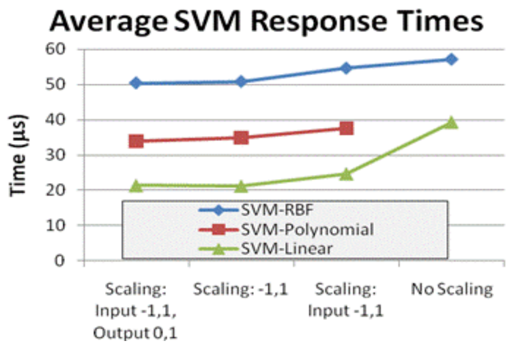

Fig. 6: SVM Average Response Times (µseconds)

5.3. *Evaluating the Trade-offs of Accuracy and Timeliness for ANNs and SVMs*

Deciding which machine learning technique to use for environments known *a priori* is straightforward. The ANN configured with 4 hidden nodes provides a configuration with 100% accuracy (shown in Fig. 2) and the lowest latency with a 100% accurate configuration (shown in Fig. 5). It is more challenging, however, to decide which technique to use for environments unknown until runtime. ANNs generally provide a lower response time, while SVMs provide higher accuracy.

$$AccuLate_t = \left(\frac{total\_samples_t - correct\_samples_t}{total\_samples_t} \times 100 + 1 \right) \times latency_t$$

where t is the machine learning technique being evaluated and latency is measured in microseconds

Fig. 7: AccuLate Formula

We created the AccuLate composite metric to provide quantitative evaluation of machine learning techniques when considering both accuracy and response latency. As shown in Fig. 7, the AccuLate metric multiplies the inaccuracy percentage of a technique by its average latency. The number of total samples minus the number of correct classifications yields the number of inaccurate classifications. This result is divided by the number of total samples to produce inaccuracy as a fraction. We multiply the inaccuracy fraction by 100 to get the inaccuracy percentage. This multiplication by 100 gives the inaccuracy equal weight with the latency when multiplying the two values (*i.e.*, the inaccuracy values range from 0 to 100 while the latency values for our current timing experiments range from single digits to double digits of microseconds).

We then add one to the inaccuracy percentage to account for perfect accuracy where the inaccuracy value would otherwise be zero and making the entire AccuLate value zero. Adding one to the inaccuracy percentage allows AccuLate to produce a useful quantitative value for comparing machine learning techniques even when the techniques are 100% accurate. The utility of this adjustment is shown for some machine learning techniques (*e.g.*, overfitted ANNs) when they are queried against the data on which they have been trained (*i.e.*, known environments).

We use inaccuracy rather than accuracy as a factor in the AccuLate formula so that a technique that has both desirable qualities of high accuracy/low inaccuracy and low latency will produce a lower AccuLate value than a technique that has either (1) the same high accuracy and higher latency or (2) lower accuracy and the same latency. The inaccuracy percentage is based on 2-fold cross-validation as outlined in Sec. 5.1 which gives guidance as to how a machine learning technique will perform given an operating environment configuration on which it has never been trained. The AccuLate formula can be easily modified to use units of measurement other than microseconds for latency. We use microseconds since the techniques we evaluated all responded within 10s of microseconds.

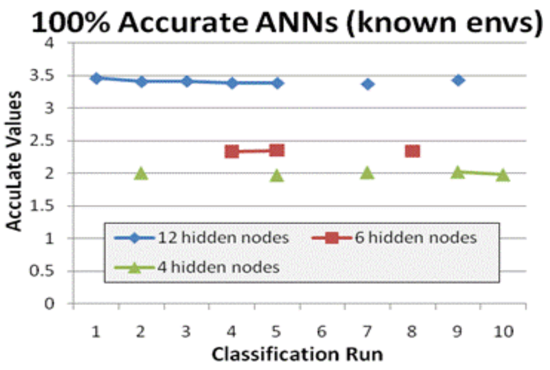

Fig. 8: AccuLate Values for 100% Accuracy Known Envs.

Fig. 8 shows the AccuLate values for the various machine learning techniques we evaluated for operating environments known *a priori* when 100% accuracy was achieved The ANN configured with 4 hidden nodes produces the best AccuLate value since it provides 100% accuracy (*i.e.*, for 5 different classification runs) and the lowest overall latency. These values are equal to the values shown in Fig. 5 when 100% accuracy is achieved which highlights AccuLate's utility in comparing techniques when accuracy is equal.

Fig. 9 shows the AccuLate values for ANNs when operating environments were unknown until runtime. This figure shows that when accuracy and latency are given roughly equal weight (*i.e.*, same order of magnitude for values), the ANN with more hidden nodes provides a better balance of both accuracy and low latency for deciding an appropriate transport protocol for a given operating environment. The ANN with 12 hidden nodes consistently provides the best (*i.e.*, lowest) AccuLate values while the ANN with 3 hidden nodes provides the worst (*i.e.*, highest) values.

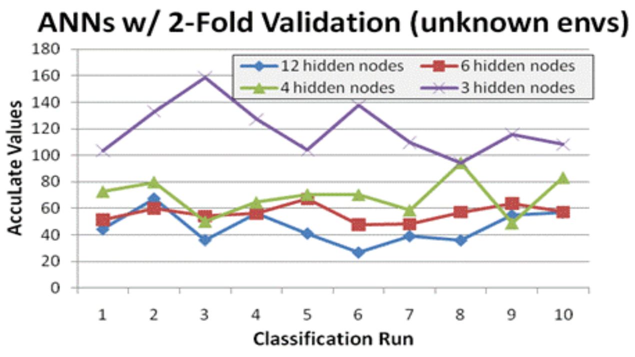

Fig. 9: ANN AccuLate Values for Unknown Envs.

Fig. 10 shows the AccuLate values for SVMs when operating environments were unknown until runtime. This figure shows that generally the SVM with the linear kernel produces the best (*i.e.*, lowest) AccuLate value. The figure also highlights that scaling the data (*i.e.*, the input operating environment and the output transport protocol) has an effect on the AccuLate values due to the corresponding change in accuracy.

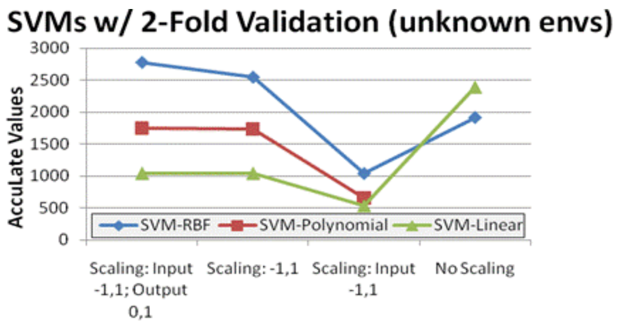

Fig. 10: SVM AccuLate Values for Unknown Envs.

In particular, when no scaling of the data is done the SVM with the linear kernel produces the worst (*i.e.*, highest) AccuLate values. Scaling the operating environment data to be between -1 and 1 produces the best AccuLate values for all the SVMs including the worst performing SVM using this scaling (*i.e.*, with the RBF kernel) which produces results better than any other kernel using a different scaling approach.

AccuLate values can also be useful in evaluating machine learning techniques within a latency threshold. These values can aid real-time systems with deadlines where several techniques may exist that fulfill the timing requirements (*i.e.*, are within the deadline). Based on response times shown in Fig. 5 and Fig. 6, if the average response time must be below 2.5 microseconds then the relevant techniques are ANNs configured with 3, 4, and 6 hidden nodes. For unknown environments,

Fig. 9 shows that the ANN with 6 hidden nodes provides the best (*i.e.*, lowest) AccuLate values relative to the required deadline with an average AccuLate value of 56.22 (as compared to values of 69.26 and 119.22 for ANNs with 4 and 3 hidden nodes respectively).

6. Concluding Remarks

The results of the experiments presented in this paper show how integrating ANNs and SVMs help address the timeliness, accuracy, and trade-offs between them for adaptive enterprise DRE systems. Below we describe some lessons learned from our work on TIML:

- **ANNs provide perfect accuracy and low latency for guidance in operating environments known *a priori*.** Our experiments showed that ANNs accurately determined which protocol supported the desired QoS for operating environments known *a priori*. Several different configurations of ANNs were able to provide perfect accuracy. We chose the ANN with the least number of hidden nodes that still provided 100% accuracy since this ANN also provided the lowest response latency.
- **SVMs provide higher accuracy than ANNs for operating environments unknown until runtime at a cost of higher response latency.** Our experiments showed that SVMs increased accuracy in determining which protocol supported the desired QoS for operating environments unknown until runtime. SVMs produced a 13% increase in accuracy over ANNs when comparing the most accurate SVM with the most accurate ANN (*i.e.*, 86.29/76.09 – 1 = 0.13). ANNs produced a 91% decrease in response time over SVMs, however, when comparing the most responsive SVM to the most responsive ANN (*i.e.*, 1 - 1.84/21.23 = 0.91).
- **Integrating ANNs and SVMs can leverage the strength of both approaches with the AccuLate providing quantitative comparisons.** When ANNs and SVMs are integrated together in constant-time, DRE systems in dynamic environments can leverage the low response time and accuracy of ANNs for operating environments known *a priori* and the accuracy of SVMs for environments unknown until runtime. When the timeliness constraints of the system preclude certain SVMs, the AccuLate metric can be used to determine which technique provides the best mix of accuracy and response latency.
- **Scaling the environment configuration and transport protocol data affects accuracy.** We were not able to produce an ANN with 100% accuracy for environments known *a priori* if the data was not scaled. Moreover, SVMs sometimes produced their most accurate results for environments known *a priori* when the data was not scaled while for environments unknown until runtime scaling the data produced the best accuracy.

Additional information and code for the technologies and tests are available in open-source form at www.dre.vanderbilt.edu/~jhoffert/ADAMANT.

References

1. N. Shankaran *et al.*, Hierarchical control of multiple resources in distributed real-time and embedded systems, *Real-Time Systems* **39**(1-3) (2007) 237-282.
2. G. Pardo-Castellote, OMG Data-distribution Service: Architectural Overview. In *Proc. 23rd Intl. Conf. on Distributed Computing Systems* (IEEE Computer Society, Los Alamitos, 2003), pp. 200-206.
3. A. Choudhary, Policy based management in the global information grid. *International Journal of Internet Protocol Technology,* **3**(1) (2008) 72–80.
4. P. Domingos, Machine learning, in *Handbook of Data Mining and Knowledge Discovery* (Oxford University Press, New York, 2002), pp. 660-670.
5. M. Xue and C. Zhu, A Study and Application on Machine Learning of Artificial Intelligence. In *Proc. Intl. Joint Conf. on* Artificial Intelligence (IEEE Computer Society, Los Alamitos, 2003), pp. 272-274.
6. A. Hess *et al.*, Principles, Autonomic Adaptation and Analysis of SIP Headers Using Decision Trees, in *Systems and Applications of IP Telecommunications, Services and Security for Next Generation Networks*, (Springer, Berlin/Heidelberg, 2008), pp. 69–89.
7. J. Hoffert *et al.*, Adapting and Evaluating Distributed Real-time and Embedded Systems in Dynamic Environments, In *Proc. 1st Int. Workshop on Data Dissemination for Large scale Complex Critical Infrastructures* (ACM, New York, 2010), pp. 23–28.
8. X. Bu *et al.*, A Reinforcement Learning Approach to Online Web Systems Auto-configuration. In *Proc. 29th IEEE Intl. Conf. on Distributed Computing Systems*, (IEEE Computer Society, Los Alamitos, 2009), pp. 2–11.
9. S. B. Kotsiantis, Supervised Machine Learning: A Review of Classification Techniques. In *Proc. Conf. on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, (IOS Press, Amsterdam), pp. 3-24.

10. T. Dietterich, Overfitting and Undercomputing in Machine Learning, *ACM Computing Surveys*, **27**(3) (1995) 326–327.
11. J. Hoffert *et al.*, Autonomic Adaptation of Publish/Subscribe Middleware in Dynamic Environments, (In submission to) *International Journal of Adaptive, Resilient and Autonomic Systems*.
12. J. Hoffert *et al.*, Integrating Machine Learning Techniques to Adapt Protocols for QoS-enabled Distributed Real-time and Embedded Publish/Subscribe Middleware, *Network Protocols and Algorithms* **2**(3) (2010) 37-69.
13. J. Hoffert and D. Schmidt, Evaluating Supervised Machine Learning for Adapting Enterprise DRE Systems. In *Proc. Intl. Symp. on Intelligence Information Processing and Trusted Computing*, (IEEE Computer Society, Los Alamitos, 2010), pp. 5-8.
14. D. Patterson, *Artificial Neural Networks: Theory and Applications* (Prentice Hall PTR, Upper Saddle River, NJ, 1998).
15. D. Meyer, D. Leisch, and K. Hornik, The Support Vector Machine Under Test, *Neurocomputing* **55**(1-2) (2003) 169-186.
16. J. Hoffert *et al.*, A Taxonomy of Discovery Services and Gap Analysis for Ultra-Large Scale Systems, In *Proc. 45th ACM Southeast Regional Conference* (ACM, New York, 2007), pp. 355-361.
17. S. Theodoridis and K. Koutroumbas, *Pattern Recognition* (Academic Press, Orlando, FL, 2006).
18. J. Hoffert *et al.*, Evaluating Transport Protocols for Real-Time Event Stream Processing Middleware and Applications. In *Proc. OTM Conferences*, (Springer, Berlin/Heidelberg, 2009), pp. 614-633.
19. S. Nissen, Implementation of a Fast Artificial Neural Network Library (FANN), *Technical Report: Department of Computer Science University of Copenhagen*, October 31, 2003.
20. C-H. Lin, J-C. Liu, and C-H. Ho, Anomaly Detection Using LibSVM Training Tools, In *Proc. 2nd Intl. Conf. on Information Security and Assurance* (IEEE Computer Society, Los Alamitos, 2008), pp.166-171.
21. Y. Liu, Create Stable Neural Networks by Cross-Validation. In *Proc. Intl. Joint Conf. on Neural Networks*, (IEEE, Los Alamitos, 2006), pp. 3925-3928.