# DOT APPROX: Making a Case for Dynamic Online Training for Function Approximation Techniques

Aurangzeb Rudolf Eigenmann
Purdue University
{orangzeb, eigenman}@purdue.edu

## Abstract

Function approximation techniques generally need training to collect information about the function under consideration. The quality of approximation may be negatively affected if the information gathered during training is not sufficient or is not representative of the function behavior during actual production runs. This paper discusses different possible training scenarios for approximation schemes. It presents a case study and advocates for dynamic online training (DOT) for function approximation. It also presents two DOT scenarios developed for a function approximation scheme and describes their effect on two applications.

## 1. Introduction

Software approximation techniques that work at coarse granularity and treat the entire application or a module, function or block as a unit for approximation, normally require training. This training is generally aimed at collecting actual input-output history of the module under consideration, to help approximate it for inputs during production. The techniques may analyze this information and infer other properties such as granularity, pattern, variations, and behavior of inputs and outputs. The training history, together with any inferred information, helps the schemes perform approximation. In this paper, we will be focusing on such schemes for *function approximation*, among which are approximate memoization and history-based piecewise approximation. However, the discussion is applicable to all general schemes that work at coarse granularity treating the underlying module as a black-box. The approximate memoization technique described in [3] stores the training input-output history of the function in a table which is accessed during production for memoized values. For unrecorded inputs, either the neighboring output is returned or interpolation is performed. The history-based piecewise scheme [1] divides the overall input range of the function in different regions based on the training history and performs low-order polynomial approximation in each region.

There are different options for training scenarios and they can affect performance and accuracy of approximation. Choosing a right training scenario is as important as choosing the right approximation scheme. We describe different training scenarios and advocate for dynamic online training (DOT), considering its features that make it the most desirable training scenario among others. We also present a case study and give results of using the DOT scenarios we developed for the history-based piecewise approximation scheme for functions.

## 2. Training Scenarios

There are different training scenarios possible as depicted in Figure 1. Training can be done offline - via a profile run, or online - during the actual production run of the application. In the latter case, it can be static - done at the beginning of the application execution, or dynamic. In case of dynamic online training (DOT), there can be different starting options for frequency, timing, and length of each training phase and these parameters may be tuned during the application lifetime. The choice of a training scenario and its parameters (length, frequency, and timing) affect, both, the approximation results as well as the performance of the approximation scheme. So it is important to choose a scenario which facilitates the underlying approximation scheme in achieving better results.

## 3. Making a Case for DOT

The quality of approximation depends on the quantity and quality of information gathered from training. By quality of information, we mean, how representative the training is of the actual application behavior during production. DOT turns out to be better than the other training scenarios in terms of collecting information which is of the right quantity and quality to help produce better approximation results. But it may have an associated overhead due to monitoring that may affect performance. While the general goal of approximation is to trade-off accuracy for performance, DOT seeks to improve the accuracy of approximation for some possible loss in performance to make the approximation results acceptable.

### 3.1 Quality of Training Information

In general, online training is better than offline training, because for some applications, the inputs in the actual production run of an application may be significantly different than the ones in the profile or an earlier run of the application. Likewise, information obtained statically at the beginning of an application execution may not be a true representative of the behavior throughout the lifetime of the application, as the behavior can significantly change over the course of time. DOT is a superior training scenario in the sense that it can capture the true application behavior throughout the lifetime of an application.

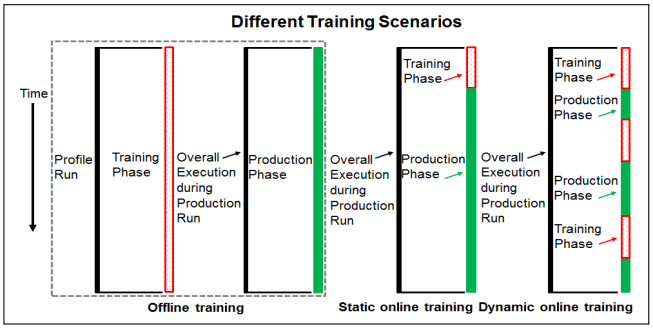


**Figure 1.** Training scenarios for function approximation schemes

### 3.2 Quantity of Training Information

Offline and static online training may require multiple trial runs with different training lengths to come up with a training history that adequately captures the behavior of an application. This process needs to be repeated for each application. Offline training is more expensive in this regard, as it requires extra profile runs to collect information. Sometimes, this overhead is not reported while mentioning the resulting application performance, as in [3], but it is a major and expensive part of the scheme. While insufficient information clearly affects approximation, having too much extra training can be undesirable as well. As it can affect the performance (more calls to original function) as well as the memory used in storing history. The latter may be a concern for memory-constrained systems and applications. DOT, on the other hand, seeks to collect the most relevant information and improve training history during the application lifetime. This self-improving nature of DOT makes it more general and more desirable than the other schemes. In some cases, when its starting parameters are too off to be improved during the application lifetime, it may also require additional attempts, but, in general, they will be fewer than those for other training scenarios.

## 4. A Case Study

This section presents a case study of a Convolutional Neural Network (CNN) application for handwritten digit detection (CNN-HDD) to highlight the importance of DOT. The activation function (tanh) of this application is amenable to approximation. This function is called 8,010,000 times during the application execution with input values in the range -26.5 to 22.5. We experimented with this application using the history-based uniform piecewise scheme [1] and used equidistant inputs. For a history size of 700, the percentage error in detecting images is 91%, if the inputs are in the range: -26.5 to 0. Although 56% of all actual inputs fall in this range but the scheme does very poor if these are used in training. For training inputs in the range 0 to 22.5, which encompasses 44% of all inputs, the percentage error is 79.4%. For inputs in the range -0.5 to 0.5, which consists of 31% of all inputs, the percentage error in image detection is 69%. But if inputs from the range -0.75 to 0.75, which covers about 40% of all inputs, are used for training, the percentage error gets significantly reduced to 6%. Inputs from a certain range seem to contain more useful information for approximation compared to others. A scheme that is treating the function as a black-box and not employing DOT will not be able to figure this out. It will only have to rely on the sequence of inputs during training which may or may not contain sufficient inputs from this useful range depending on the training length. By contrast, DOT will be able to include more pertinent inputs in the training history. By extending the training time, other schemes may be able to collect similar information, but will incur overheads in terms of performance and memory usage. DOT will achieve better results compared to other scenarios and with less history elements.

We enhanced the history-based non-uniform piecewise scheme realized via a binary search tree [1] with a DOT scenario to compare it with its static counterpart. In this scenario, the initial training of some length is performed at the beginning of application execution. The system re-enters the training phase on every $\mathbf{n}^{th}$ function call. It stays in the training phase until a given percentage of inputs in the training window produce acceptable results. The value of **n**, size of training window, percentage of inputs in training window required to produce acceptable results to get out of training, and the acceptable error in results, are configurable parameters. The scheme can optionally improve some of the parameters during the application lifetime. Figure 2 compares DOT with static online training for the CNN-HDD application for the said approximation

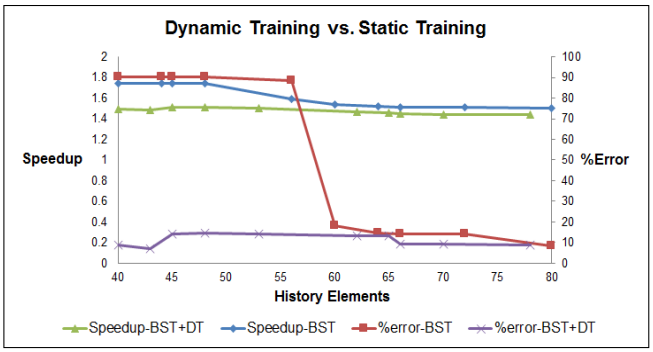


**Figure 2.** DOT vs. static online training for CNN-HDD using history-based non-uniform scheme realized via a binary search tree

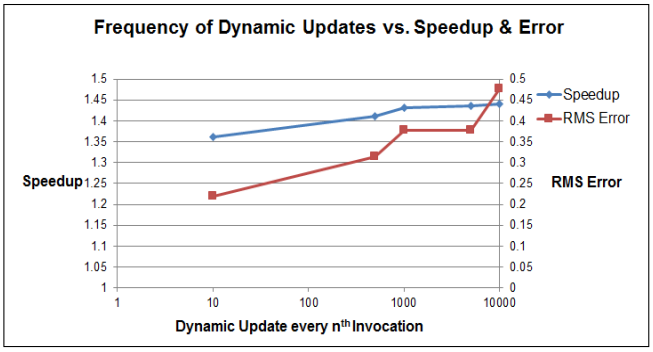


**Figure 3.** Effect of frequency of dynamic update on speedup and RMS error for the Blackscholes application

scheme. DOT helps achieve better approximation compared to its static counterpart for the same number of history elements with a slightly less speedup because of monitoring overhead (as there are more calls to the original function). The parameters used in the scheme are: initial training of length 15; training window of size 3; 70% of inputs in training window required to produced acceptable results; frequency of training (**n**) set to 1,000,000; and acceptable error varied between 0.03 and 0.15.

The DOT scenario described above is just one example and other variants are possible. Even a simple scenario in which training history is updated once every $\mathbf{n}^{th}$ function call after the initial training, may improve approximation for some applications. Figure 3 shows how this scenario can affect speedup and performance as the frequency of update is varied (more update helps improve results at the cost of some loss in performance) for the Blackscholes application from the Parsec Benchmark Suite [2]. But DOT scenarios that are more sophisticated in their mechanism of trying to find the most pertinent and useful training information and are self-improving would be more desirable because of their wider applicability and effectiveness.

## 5. Conclusion

Coarser-granularity black-box approximation techniques rely on training information. The quality and quantity of training information can affect both, approximation results and performance. Choosing a suitable approximation scheme is important and so is choosing a suitable training scenario. DOT should be preferred over other scenarios as it can be general and self-improving. It can also provide pertinent training information that is both sufficient and representative of the application behavior during the production run, thus achieving better results.

## References


[1] Aurangzeb and R. Eigenmann. History-based piecewise approximation scheme for procedures. 2nd Workshop on Approximate Computing (WAPCO), Jan 2016.

[2] C. Bienia and K. Li. *Benchmarking modern multiprocessors*. Princeton University USA, 2011.

[3] M. Samadi, D. A. Jamshidi, J. Lee, and S. Mahlke. Paraprox: Pattern-based approximation for data parallel applications. In *ACM SIGARCH Computer Architecture News*, volume 42, pages 35–50. ACM, 2014.