November 28, 2018

Reinforcement Learning as a Basis for Optimization of Operating Systems

Mark Schott

Reinforcement Learning as a Basis for Optimization of Operating Systems

Abstract:

Reinforcement learning techniques are provided that generate initial training data to refine a machine-learning model (e.g., a neural network). The techniques allow a machine-learning system to make a correlation between inputs and outputs, and analyze generated outputs to produce new training data that will produce better outputs. The techniques start by profiling a system (e.g., an operating system) and a mock model of the machine-learning model that provides random outputs or outputs according to a simple heuristic. The techniques can then use the outputs to adjust heuristics of the system to obtain a wide variety of performance reactions of the system. Evaluation of the profiling data of the system can be performed to distinguish good outputs from bad outputs according to a chosen performance metric. In some cases, initial training data is created based on the good outputs. A new machine-learning model can then be trained with the initial training data to produce better outputs than the outputs produced by the mock model. With each iteration of the techniques, the machine-learning model produces better outputs. Generally, these techniques are repeated until outputs of satisfactory quality are achieved.

Keywords: Reinforcement learning, machine-learning, mock model, random outputs, iterative learning, heuristics, neural network, operating system optimization

Background:

Today, existing traditional heuristics are supplemented with a machine-learning model to produce better predictions without sacrificing performance. Typically, the machine-learning model is trained using large amounts of pre-labeled data collected from a large number of devices that have been deployed in the field for an extended period of time. This data is then evaluated based on some performance metric and used to train the machine-learning model to produce better predictions.

Operating systems today use a number of different heuristics to make decisions about which actions to take in a variety of different situations. For example, the operating system may make decisions such as deciding which process to schedule next or when to split and/or combine blocks of memory for page allocation. The heuristics used by the operating system to make decisions must be sufficiently fast in order to achieve acceptable levels of performance. The heuristics used to make a decision, however, are often not the best possible heuristics based on information that may be available to the operating system, which can result in sub-optimal performance of the operating system. With machine-learning, the heuristics can be supplemented to produce better decisions.

Collecting large amounts of pre-labeled data that is useful to train a machine-learning model, however, can be cumbersome to implement, time consuming to collect, and lack a wide variety of data that would be useful for the machine-learning model. As such, it would be beneficial to obtain data useful to train a machine model in an efficient and faster manner without needing as much the data that is conventionally structured or pre-labeled for training the machine-learning model.

Description:

To address the issues related to generating data for training a machine-learning model, reinforcement learning techniques are provided that generate initial training data to be used for training the machine-learning model. Fig. 1 illustrates an example machine-learning system in block diagram form, which can implement one or more aspects of the reinforcement learning techniques. Generally, reinforcement learning is a technique that allows a machine-learning system to make a correlation between inputs received by the machine-learning system and outputs provided by the machine-learning system.

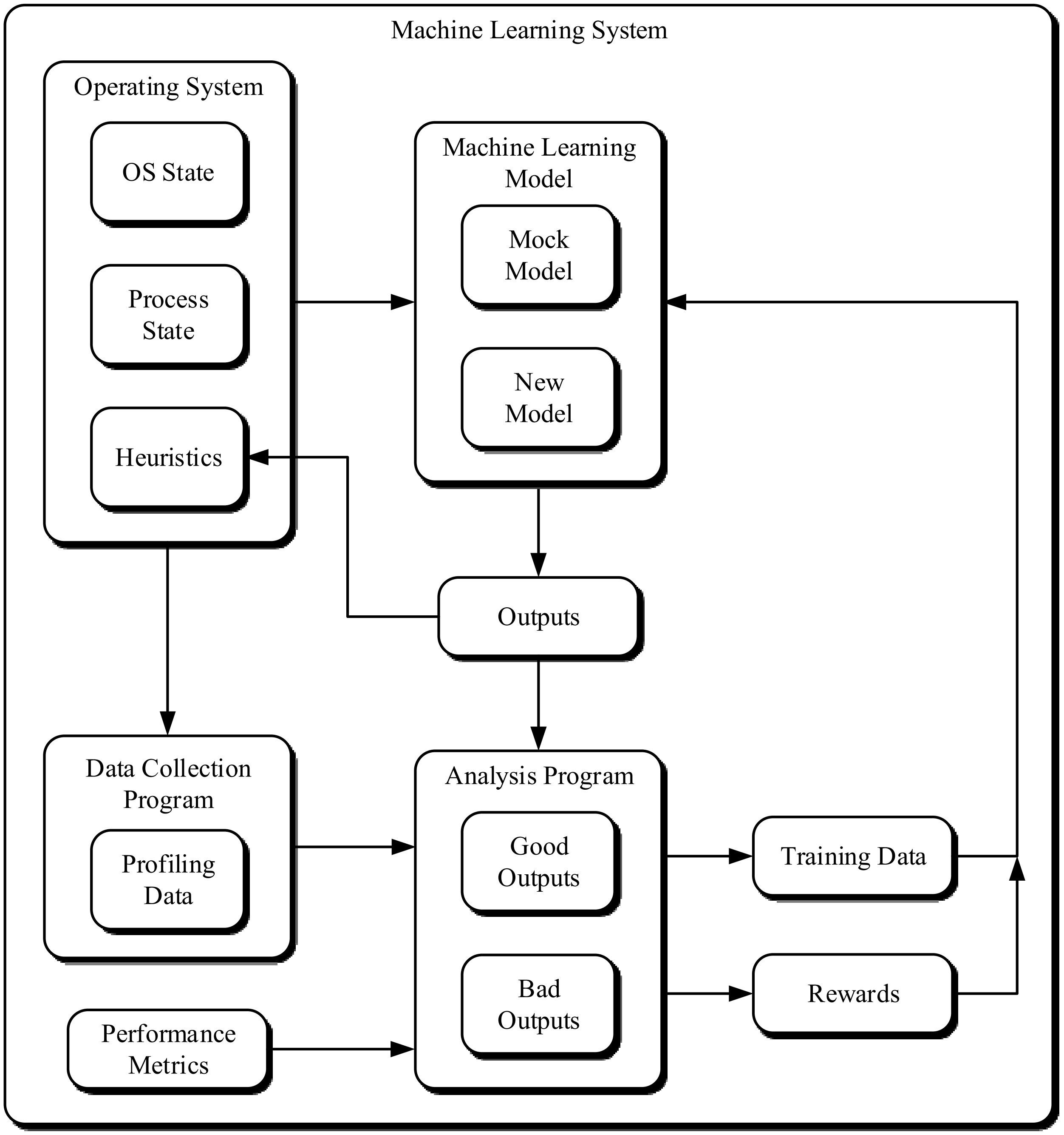

Fig. 1

As illustrated in Fig. 1, the machine-learning system includes a machine-learning model (*e.g.,* a neural network) that generates outputs based on inputs provided to the machine-learning model. The inputs can be provided by an operating system (OS) and/or previous outputs of the machine leaning model. For example, an OS state or process state of the OS can be used as an input for the machine-learning model.

The machine-learning model illustrated in Fig. 1 may include a mock model and a new model. The mock model of the machine-learning model can generate initial outputs that are useful to adjust heuristics of the OS, such as to collect a large variety of useful data in a much shorter duration of time than the time needed to collect data conventionally from devices in the field. This is advantageous because more variations can be introduced in the OS state or process state than are typically seen with devices in the field under normal use or data collection. When a sufficient amount of useful data has been collected to create training data, the new model of the machine-learning model may replace the mock model of the machine-learning model. The new model can then use the training data provided by the mock model to generate better or more-accurate outputs than the previous outputs generated by the mock model.

In some aspects of reinforcement learning, the machine-learning model (*e.g.,* the mock model or new model) generates outputs that are evaluated by an analysis program, which can be configured to sort or select good outputs from bad outputs generated by the machine-learning model. This analysis program, which is illustrated in Fig. 1, may then produce training data or additional training data that is provided as an input to the machine-learning model to generate another set of better outputs, which in turn produces even better training data. The analysis program may also create or track rewards for the machine-learning model that produces the better outputs relative those of other machine-learning models.

The machine-learning system illustrated in Fig. 1 may also include a data collection program that collects profiling data of the OS. For example, the data collection program can collect the profiling data while the OS runs stress-testing benchmarks or other known performance metrics. The analysis program can utilize this profiling data and performance metrics as a basis for sorting or selecting the good outputs from the bad outputs generated by the machine-learning model. Alternately or additionally, the machine-learning model can be trained to optimize any one or subset of performance metrics chosen by a programmer or developer.

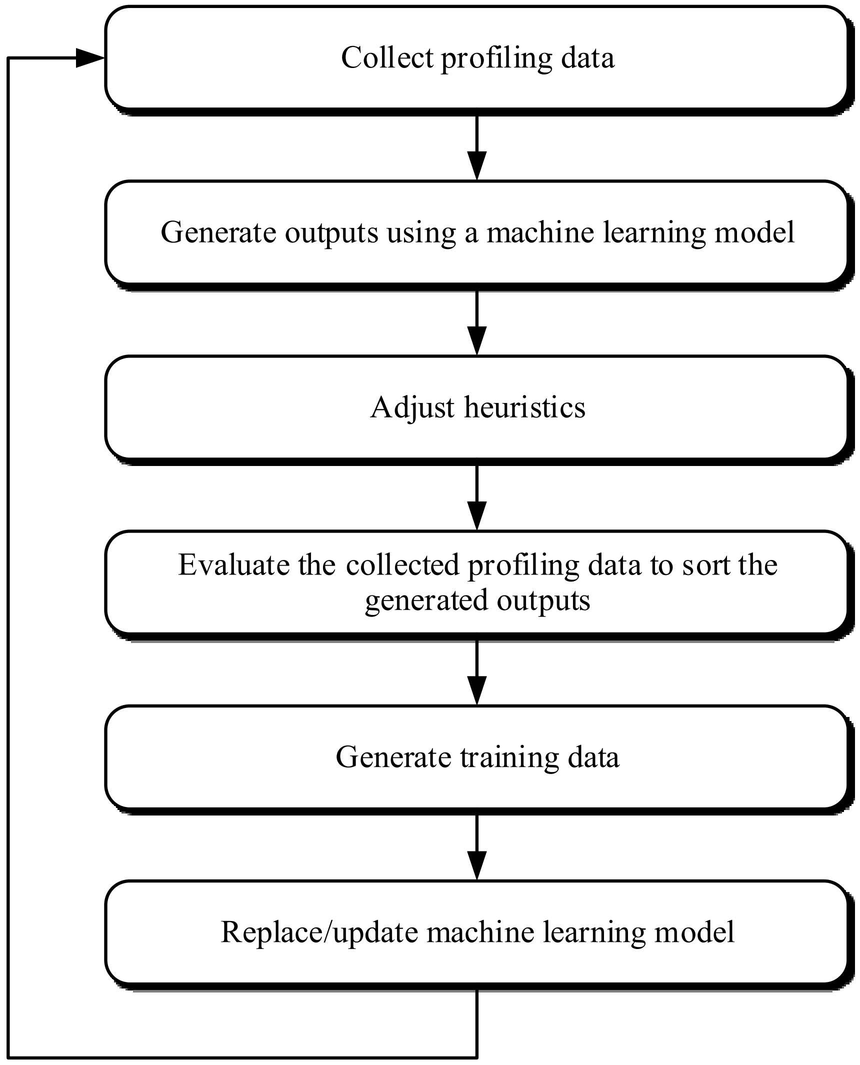
Collect profiling data
Generate outputs using a machine learning model
Adjust heuristics
Evaluate the collected profiling data to sort the generated outputs
Generate training data
Replace/update machine learning model

Fig. 2

Fig. 2 illustrates steps of an example method to perform the reinforcement learning techniques. In some aspects, the machine-learning system shown above in Fig. 1 performs the reinforcement learning techniques.

The reinforcement learning techniques may start with a data collection program that collects profiling data to correlate an operating system (OS) state or process state to heuristic outputs, such as those that would best improve the OS performance. For example, rather than collecting profiling data from a large number of devices in the field, the data collection program of Fig. 1 collects profiling data from development devices in a development lab or in virtual machines.

A mock model of the machine-learning model can then generate random outputs or other outputs according to a simple heuristic. Based on these outputs, the techniques can adjust heuristics of the OS, such as kernel heuristics or memory heuristics, to obtain a wide variety of performance reactions of the OS. By introducing more variation into the OS state or process state that provide inputs to the mock model, a large number of outputs can be generated by the mock model, which in turn enables collection of a numerous variety of useful data in a much shorter amount of time than attempting to coordinate and collect data from devices in the field.

After generating outputs, an analysis program may evaluate the collected profiling data, such as in accordance with one or more chosen performance metrics in order to distinguish, sort, or filter good outputs from bad outputs. In some cases, the analysis program of Fig. 1 evaluates the outputs generated by the mock model and sorts out which outputs have a positive effect on OS performance based on the profiling data and performance metrics (*e.g.*, selected or default metrics). For example, outputs that are determined to have a positive effect or impact can be considered as good outputs and outputs that have a negative or negligible effect can be considered as bad outputs.

The analysis program can then produce initial training data using the good outputs generated by the mock model.

Based on the initial training data, a new model of the machine-learning model can be trained using inputs that includes at least some of the initial training data produced by the analysis program. Alternately or additionally, the machine-learning model can be trained using corresponding OS state or process state information from which the initial training data is generated. This new model may then replace or update the mock model of the machine-learning model by which the initial training data is created. After replacement or update with the new model, the machine-learning model may start or perform another round of the data generation, collection, and/or profiling as described above. This round of profiling, however, may produce better outputs than the random outputs used at the start (*e.g.*, an initial state with more random outputs), which in turn generates more useful data for the analysis program of the machine-learning model to evaluate. This analysis program can then generate new and better training data than the initial training data. Accordingly, with each iteration of the steps described above, the techniques or machine-learning model can generate better outputs than a previous iteration of outputs. This process can be repeated until the machine leaning model generates outputs of sufficient quality according to the performance metrics, such as those chosen or selected by the system designer.