Situation Assessment and Machine Learning

jbayne@stratum4.org

Much attention is being given today to the subject of "machine learning" (ML). ML activities primarily occur in academic research and industrial R&D labs focused on the identification and tracking of unfolding situations in physical and synthetic processes. ML techniques include process modeling, simulation, visualization and data analytics. Applications of ML occur in gaming, pattern recognition in vision and speech, manufacturing process control, bioinformatics, financial trading, autonomous transport, healthcare diagnostics, information security, and environmental data analysis. Consequently, situation assessment and machine learning are central to Stratum 4's approach to cognitive cyber-physical systems, with focus is on intelligent automation and control systems.

Cognition is required for a system to achieve and maintain situational awareness. This ability presupposes that a system is designed and commissioned with a degree of awareness that can improve over time as it operates. Improvements in a system's awareness requires the recognition and assimilation of new knowledge related to changes in the states and behaviors of production processes under its control, and in changes in its operating environment (context).

This note discusses the relation between cognition and machine learning.

Information Processing Pipeline

As discussed elsewhere[1], the technical framework supporting Stratum 4's approach to intelligent (adaptive) cyber-physical systems is predicated on cognitive capabilities implemented along a data acquisition and control processing pipeline, shown below.

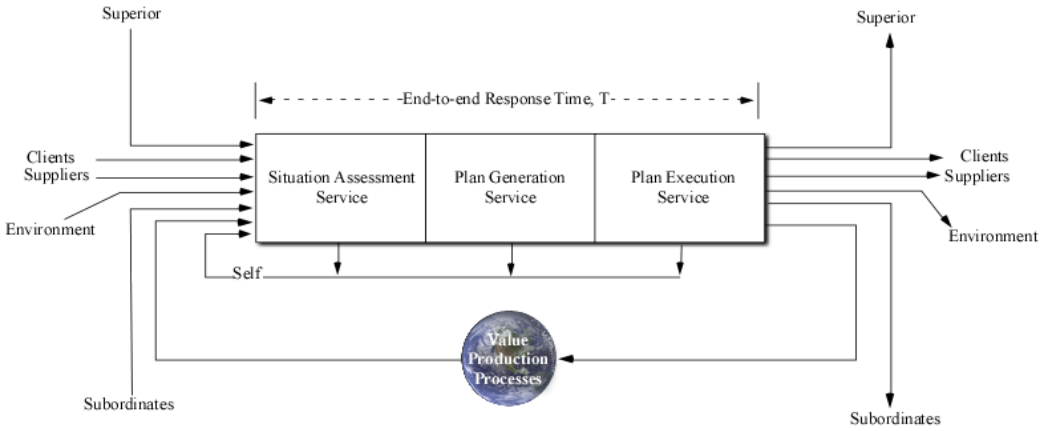

Figure 1 - Information Processing Pipeline

In this feedback control model, information is contained in data streams, expressed in the form of voice, video, telemetry (e.g., position, temperature, pressure, velocity), internet search results, and database queries that flow into a Situation Assessment Service (SAS) where they are continuously processed. SAS identifies and tracks events, within and across streams, that signify occurrence of new or unfolding of

[1] https://stratum4.org/technical-framework/

situations of interest to *value production processes*. Within SAS, recognition of changing situations results in selection of one or more candidate responses, or courses of action, COA. COA are subsequently processed by a Plan Generation Service (PGS) responsible for establishing operational policies ("rules of engagement") governing acceptable responses. If policy-compliant, PGS then attempts to acquire and assign needed operational assets (e.g., human, capital and material) to the COA, a process resulting in creation of an executable plan of action (POA). POA are subsequently handed to the Plan Execution Service (PES) for execution. PES obtains authorization to proceed, effectively converting a POA into a Plan of Record (POR), then schedules, executes and monitors the plan's individual tasks needed to achieve milestones and deadlines, as required. In execution, POR tasks affect the value production processes under control, the effects of which are subsequently visible through various input streams, thus completing the regulatory (autonomic) feedback control loop as shown in the figure.

Event Patterns

Cognition and machine learning are important elements in the functioning of the SAS stage where real-time recognition of event patterns results in situational awareness and the generation of feasible reactions. Tin the figure below, the SAS is modeled as a continuous three-stage computation that serves to detect events in "raw" inputs {i} and produce one or more executable responses {c}.

Figure 2 - Situation Assessment Service

Possibly multiple input data streams enter filter processes (FP) where events of interest are identified. Event objects are defined as the occurrence of a sequence of data objects occurring serially within a given stream, across multiple streams or both. In the figure below, three input streams (S_1, S_2 and S_3) carry data points occurring in the time interval [t_0, t_{10}].

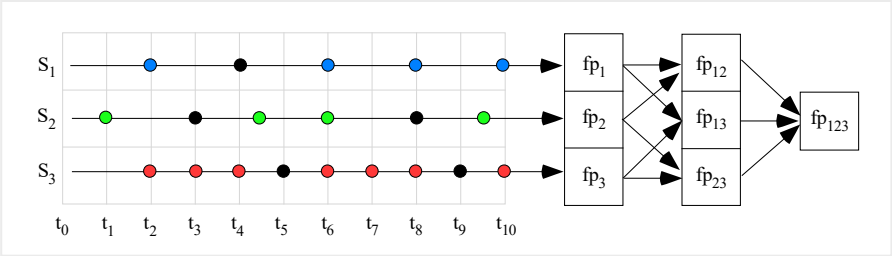

Figure 3 – Concurrent Data Streams

As shown, depending on the number of input streams and their event signatures, the FP process performs pattern matching longitudinally within and transversely across streams to identify known situations of interest. A history process facilitates learning by capturing data streams so that event patterns that recur over time, for example preceding or following known patterns, can be identified as and tracked and responded to with new or modified COA.

In the figure below three events (e_1, e_2, e_3) are defined by the occurrence of individual data $d_i(t_j)$, that occur within and across event streams S_1, S_2 and S_3. In this example, a situation of interest has the signature $\{e_1 \mid e_2 \mid e_3\}$.

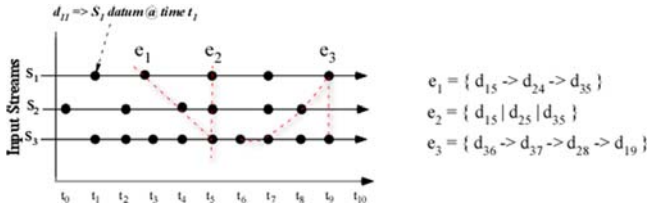

Figure 4 – Situations: Time-sensitive Patterns of Events

Cognition and Machine Learning

The ability of automation systems to identify and respond, with a degree of agility, to evolving situations requires additional *supervisory* services—some performed "out of band" (asynchronously) typically under periodic human supervision, and some "in band" (synchronously) performed automatically as the system runs, with minimal supervision. This additional machinery, shown below, affects all three pipeline stages, but for brevity, we focus here only on the SAS stage.

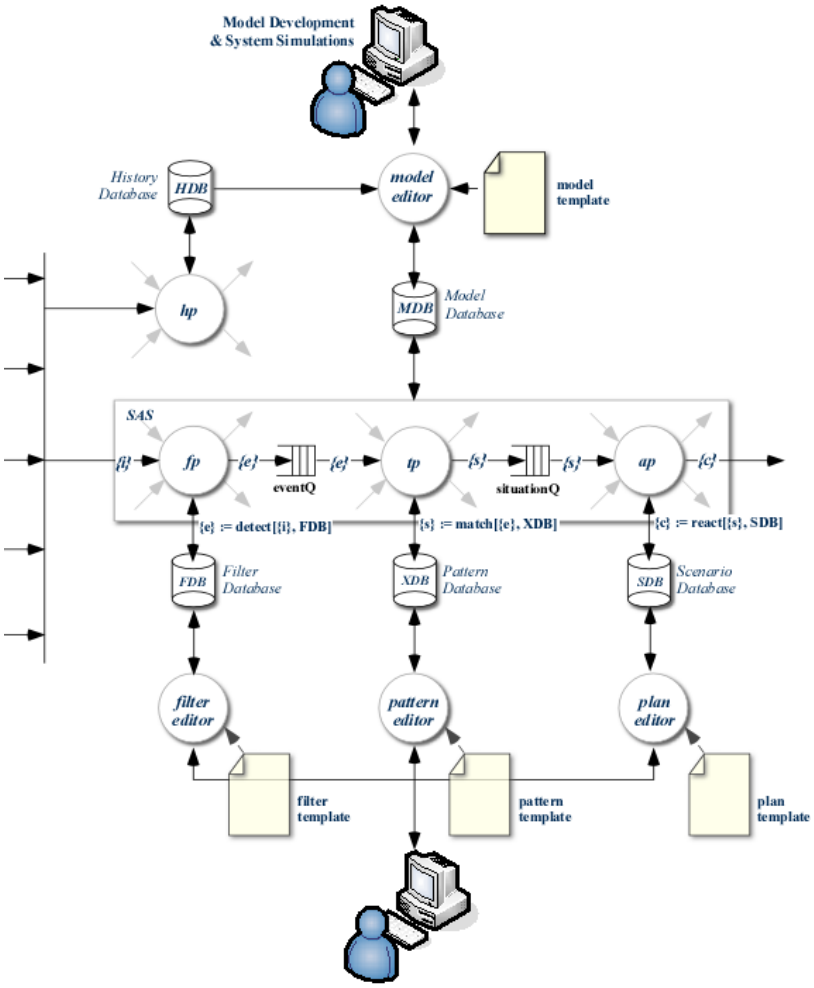

Figure 5 -SAS Learning Infrastructure

As noted above, input streams are presented to the SAS stage's filter process (FP) and simultaneously recorded in a history database (HDB). The HDB serves several critical functions, and is central to the system's model development and machine learning activities. More on this aspect later.

Above the central pipeline (core) are human-directed model development functions where "out of band" data analytics and machine learning activities based on input streams are concentrated. Below the core are supervisory services that guide the in-band learning activities. We begin with a brief description of the in-band activities.

A Word About "Consciousness"

In our mechanistic view, *consciousness* in man and machine is predicated on a hierarchical layering of awareness. At a base level is an awareness of self (see Figure 1), a "protoself," and a system's sense of its own physical embodiment. Above the protoself exists a system's "core consciousness," the operational awareness of its protoself as it interacts with its encapsulating environment—the interplay between a system's states and behaviors and those of external objects in the world. Lastly, there is an advanced awareness of history and the causal role antecedents play in anticipating and planning

responses of potential future behaviors. Our present development efforts are concerned with the implementation of the protoself and core consciousness in intelligent automation systems.

Filter Process (FP)

The FP is responsible for detecting the occurrence of "events," the occurrence of data contained within and across data streams. Events signify changes in the states or behaviors of value production processes.

At the commissioning of a cognitive system, FP's Filter Database (FDB) contains an initial complement of event sequences, the basis for its protoself and core functionality. The event patterns of initial concern involve error conditions for the system's primary services (it's protoself) and essential patterns expected in its automation of essential value production processes.

FP performs the computation {e} = detect[{i}, FDB].

These proto and initial core event patterns are specified in and developed with the aid of a "filter editor" that employs "filter templates" that structure pattern records in the FDB. FP's function is to continuously filter and process input streams looking for event patterns to trigger appropriate responses.

Triage Process (TP)

The TP is responsible for mapping identified sequences of events onto "situations" concerning the performance of value production processes under observation and control of the automation system.

At commissioning, the TP's Pattern Database (XDB) contains scenario patterns (signatures) of situations of interest, and based on specific (optimization) criteria, to prioritize candidate responses.

TP performs the computation {s} = match[{e}, XDB].

Analysis Process (AP)

The AP is responsible for mapping recognized situations (scenarios) occurring in a supervised value production process that requires a formal response by the automation system.

At commissioning, the AP's Scenario Database (SDB) contains a predefined set of possible courses of action (COA) to known situations, reaction plans able in principle to respond to internal (self) and external (process) situations identified by TP.

AP performs the computation {c} = react[{s}, SDB].

End-to-end, the SAS pipeline, in mapping events detected in one or more input streams {i} to candidate response plans {c}, continuously performs a nested computation. For reasons of correctness and stability, the end-to-end computation is required to meet FP, TP and AP timing constraints.

SAS performs the computation {c} = react[match[detect[{i}, FDB], XDB], SDB]]]

Machine Learning

The need for dynamic learning and the subsequent raising of a system's awareness is a necessary property of an adaptive automation system. In the framework presented here, the learning machinery is embodied in the model building process (Figure 5) and their use in the three stages of the SAS pipeline. As information streams are captured by the history process (HP), the data they contain provides the raw material for models of the protoself, the core consciousness, and a system's advanced prediction and planning functions.

Models created, validated and updated through simulations are put into service through the FP, TP and AP pipeline functions. Derived models may involve dynamic behaviors expressed as deterministic finite-state machines, finite Markov Processes, network queueing models, mathematical functions (e.g., stochastic differential equations), and tensors and neural networks. While there are many powerful analytical software tools for expressing dynamic and learning models (e.g., R, MATLAB, Mathematica, TensorFlow, PyTorch), the examples listed below are expressed in the Wolfram Language.

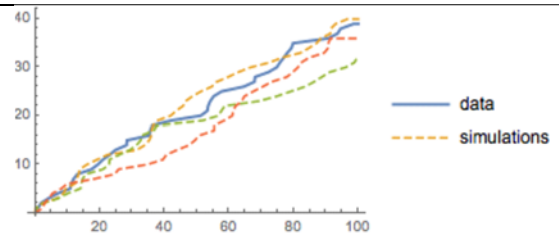

Figure 6 – Estimating a Process from Data

```
data = RandomFunction[PoissonProcess[.4], {0,10^2}];
eproc = EstimatedProcess[data, PoissonProcess[λ]]
sims = RandomFunction[eproc, {0,10^2},3];

ListLinePlot[Join[data["Paths"], sims["Paths"]], PlotStyle
→ {Thick, Dashed, Dashed, Dashed}, PlotLegends
→ {"data", "simulations"}]
```

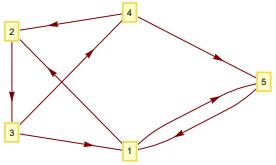

Figure 7 - Finite State Machine

```
FSM= {1→2, 1→5, 2→3, 3→4, 3→1, 4→2, 4→5, 5→ 1};

FindCycle[FSM]
{{1 → 2, 2 → 3, 3 → 1},
{2 → 3, 3 → 4, 4 → 2}}
```

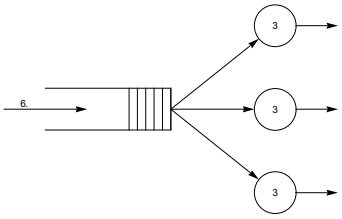

Figure 8 – 3-Server Queuing Model

```
Server = QueueingProcess[6.,3,3];
QueueProperties[Server, "QueueDiagram"]
QueueProperties[server, "MeanQueueSize"]
0.8888
```

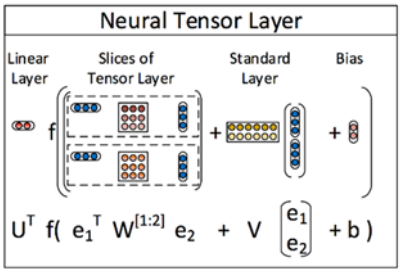

Figure 9 – Tensor Network Models

Tensors are symbolic arrays of arbitrary dimensions.

```
$Assumptions = m ∈ Matrices[{4,4}, Reals,
Symmetric[{1,2}]]

TensorRank[m] = 2
TensorDimensions[m] = {4,4}
```

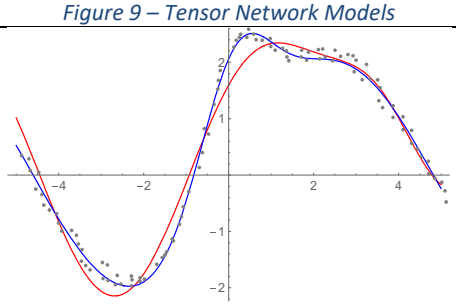

Figure 10 - Curve Fitting Optimization

```
model=a Exp[-b (x-c)^2]+d Sin[ω x+φ];
data=Table[{x, model/.{a->2,b->1,c->0,d->2,ω->0.67,φ-
>0.1}},{x,-5,5,.1}]+RandomReal[.25,101];

f1 = FindFit[data, model, {a, b, c, d, ω, ϕ}, x]
f2 = FindFit[data, model, {a, b, {c, 0}, d, ω, ϕ}, x

Show[Plot[Evaluate[model/. {f1, f2}], {x, −5,5}, PlotStyle
                → {Red, Blue}], ListPlot[data, PlotRange
                → All, PlotStyle → Gray]]
```