Generic Prognosis With Evolutionary Approaches

Nicolas Schneider
EADS Innovation Works
nicolas.schneider@eads.net

ABSTRACT

XCS-GP (Generic Prognosis) is a new process to compute system degradation in industrial prognosis activities field. Our approach is based on *XCS* core algorithm [6] extended with an auto-diagnosis module and dynamic learning strategies. We've tested our system on a real *Airbus* use case. We predict the degradation of an *A340* system brake wear based on the aircraft mission plan. The underlying idea of our approach is to find implicit links between aircraft data and environmental data which leads to specifics aircraft degradation. Results are promising, furthermore our approach does not need experts knowledge. It is therefore generic and could be easily applied to other systems.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Evolutionary prototyping*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Industrial automation*; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

Keywords

Prognosis, Aircraft System Degradation, Evolutionary algorithms, Classification, Dynamic Learning, Auto-Diagnosis

1. INTRODUCTION

Prognosis could be defined as the detection of the precursors of a system failure and the computation of the remaining time before its occurrence. Our work belongs to the *health management* research field (which includes *detection*, *diagnosis* and *prognosis* of failure). *Health management* capabilities brings several advantages:

- To increase *safety* by detecting a problem before it becomes critical.
- To increase *reliability*: operational interruption could be avoided thanks to early detection.
- To reduce *delays* and associated *costs* by avoiding unscheduled maintenance tasks.

Moreover, aeronautic and aerospace activities are domains that need to manage dynamic and unexpected situation in which reactivity and ability to understand the current situation is crucial. Regarding our problem, the aircraft fly during its whole life cycle (30-50 years). During this time, many environmental constraints could change (new airports, new flight routes, climatic changing, runaway repairs, etc.) and the system of the aircraft as well (system's ageing laws, new system suppliers, new configurations...). To manage these constraints, the solution should provide dynamics and continuous learning capabilities.

Therefore we propose to build a prognosis capability through learning from in service usage of the system based on the XCS core algorithm [6] upgraded to fit our requirements.

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

2. PROBLEM DESCRIPTION

The use case is focused on aircraft brake system. The objective is to predict the break wear degradation during the aircraft's mission.

Three kinds of historical information are available:

- The *operational data* that only depends on the arrival airport with the following data: *Runway length*, *rolling coeff.* and *condition*[1]; *Cruise severity*[2]; *Weather*.
- The *functional data* directly measured on the aircraft at landing: *Weight*; *velocity*; *Brake op*[3]; *Flap op*[4]; *Flap Angle*: *Brake temperature*.
- The incremental *brake wear* is expressed in millimeters.

3. RESOLUTION PROCESS

Learning Classifier Systems [5] are rule-based systems that automatically build their rules set. These rule sets evolve thanks to two complementary processes: a *genetic algorithm* manages the evolution of the rules set while a *reinforcement process* remunerates the *accurate* rules according to the environment. We used a specific version of the classifiers systems named *XCS* that is accurate for classification and data mining purposes [1], [2], [7].

Contrary to existing *XCS* systems, our process needs to evolve as close as possible to the aircraft behaviors during its whole life. Therefore, our systems needs to be aware of any magnitude of change (also named concept drift (*CD*))

[1]Arrival airport runway status (*poor*, *fair*, *good*).
[2]Flight conditions on the landing airport (*normal* or *high*)
[3]The aircraft spoiler are used during landing or not
[4]The flaps are open during landing or not

[4]. In reality, a concept drift is so frequent that any data mining applications that deal with online data need to treat seriously the problem. In consequence, our system should be able to diagnosticate itself and take the necessary measures to enhance the on-going prediction.

To manage that, we've added an auto-diagnosis module that can detect three different learning states:

- *Stable state*: the system is stable, the prediction rate has converged to an optimal value.
- *Concept drift state*: a change due to a concept drift is observed. Generally this is the result of a degradation of the convergence, an increase of the error and classifiers number.
- *Chaotic state*: the system become chaotic. It is not possible to find any underlying model from the input data.

To detect the different states, we store three *time windows* (M) with the error M_E, prediction M_P and population size M_{Pop} values. Each *time window* has a size proportional to the parameter θ_{GA}.[5] We compute the standard deviation *stdev* and the linear regression coefficient *lrc* for each window. The different states are defined as following:

$$\Omega = [M_1, M_2], M_i \in M, i \in [1, 2] \tag{1}$$

$$Stable \Leftrightarrow \forall m \in M, stdev(m) < s_1 \tag{2}$$

$$CD \Leftrightarrow \forall m \in M, \forall \Omega \begin{cases} stdev_{t-1}(m) < s_1 \wedge \\ lrc_t(\Omega) < 0 \vee \\ lrc_t(\Omega) > 0 \end{cases} \tag{3}$$

$$Chaotic \Leftrightarrow \forall \Omega, \forall m^{'} \in \Omega \begin{cases} \frac{\sum_1^{\theta_{GA}} m^{'}}{\theta_{GA}} < s_2 \wedge \\ stdev(\Omega) > s_3 \end{cases} \tag{4}$$

Where s_1, s_2 and s_3 are parameterizable thresholds.

Once the system is able to auto-diagnosis itself, we can apply different strategies according to the current state:

- *Stable*: We challenge the classifier's set. We simulate a strong death policy to force the population to get out from local minima. This challenge is repeated n time before stopping the learning. Then we keep the population that have the minimal error.
- *Concept Drift*: We use the *adaptive learning strategy* described by "Hai & co" [3]. *"Hai & co" analysed the magnitude of change in the XCS learning. They proposed an adaptive learning strategy to adjust the learning rate β according to the error's prediction to optimize the convergence*
- *Chaotic*: We stop the learning and inform that no model has been found. Specific user parametrization is required.

Finally, our *XCS-GP* process shall be able to build classifiers from historical data that modeled the relation between *operational data*, *functional data* and the corresponding *brake wear*. Added to that *XSC-GP* shall be able to auto-adapt itself to increase the prediction during the aircraft's life.

[5]Threshold to call the genetic algorithm.

4. TESTS AND RESULTS

We have 126 different flights with the incremental degradation of the break wear. We've divided the data in two sets: the learning set with 90 flights and the testing set with 36 flights in which 4 specifics flights have a strong break wear. This specific testing selection is made to highlight that our model doesn't behave like a linear degradation function.

Results are illustrated in figure 1

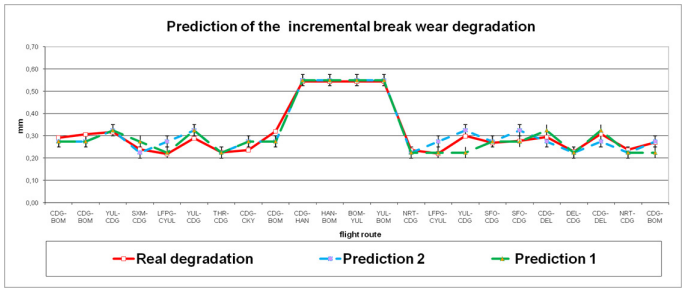

Figure 1: The plain curve is the real break wear and the dotted curves are the predicted break wear.

5. CONCLUSION

Our *XCS-GP* process is a generic process based on *XCS* to predict system degradation. Our process follows the underlying idea that the *operational environment* which a system evolves has an impact on its own *functional environment*. If the *functional environment* is used to compute knowledge (degradation, health measurement, manufacturing time...); this knowledge will accurately be predicted with *XCS-GP* only with the *operational environment*.

6. REFERENCES

[1] Jhon Holmes Alwyn Barry and Xavier Llora. Data mining using learning classifier systems. *Foundations of Learning Classifier Systems*, 2004.

[2] Xavier Llor' Ester Bernad ÌĄ and Josep M. Garrell. Xcs and gale: a comparative study of two learning classifier systems with six other learning algorithms on classification tasks. *ADVANCES IN LEARNING CLASSIFIER SYSTEMS*, 2321/2002:115–132, 2002.

[3] Chris Lokan Hai H. Dam, Hussein A. Abbass. Evolutionary online data mining an investigation in a dynamic environment. *The Artificial Life and Adaptive Robotics Laboratory*, 2005.

[4] Martin V. Butz Hussein A. Abbass, Jaume Bacardit and Xavier Llora. Online adaptation in learning classifier systems: Stream data mining. Technical report, IlliGAL, 2004.

[5] David E Goldberg Lashon B Booker and John H Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235–282, 1989.

[6] Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[7] Stewart W. Wilson. Mining oblique data with xcs. *Tech report number: 2000028*, 2000.