Diagnosis of a Chopper Controlled DC Motor by Boosting

Pedro J. Abad; Antonio J. Suárez
Dpt of Electronic, Informatics Systems & Automatic
University of Huelva, Spain
abadhe@diesia.uhu.es; asuarez@diesia.uhu.es

Rafael M. Gasca; Juan A. Ortega
Dpt of Languages & Informatics Systems
University of Sevilla, Spain
gasca@lsi.us.es; ortega@lsi.us.es

Abstract

This paper proposes a methodology to diagnose a transient state of a dynamic system using boosting. The methodology is composed by two steps: one off-line process and another on-line process. The off-line phase begins gathering data from the system, both when it is running free of fault and when the system is running in each fault mode. A segmentation and normalization algorithm is used to reduce the large amount of gathered data. The final step is the generation of a decision tree by a classification tool. The boosting technique is used with the aim of improving the classification results. The on-line process of the methodology consists of evaluating a new reading of the system sensors with the generated decision trees. The diagnosis of the system is the result of this evaluation which has very low computational cost due to the simplicity of the decision trees. Also, the implementation cost is very low due to this simplicity.

1 Introduction

A long variety of techniques, coming from Artificial Intelligence among other, has been applied to diagnosis field from their beginning [8]. Inside the Artificial Intelligence techniques, data mining is about solving problems by analyzing data already present in databases. Data mining is defined as the automatic process of discovering patterns in data. One of the fields of Data Mining is the Machine Learning, which is defined as the ability for a computer system to generate new knowledge based on its past experiences. Machine Learning techniques have been applied to diagnosis field, from a decade ago [3] to current years [9]. Inside classification techniques, the *boosting* is part of ensemble methods. The basic idea is that given a set of classificators, better results could be obtained using a combination of all of them instead of using one alone, even if it is the best. *Boosting* is a general method for improving the performance of any learning algorithm. In theory, boosting can be used to significantly reduce the error of any learning algorithm that consistently generates classifiers which need only be a little bit better than random guessing. *Boosting* works by repeatedly running a given learning algorithm on various distributions over the training data, and then combining the classifiers produced by the weak learner into a single composite classifier. The first provably effective boosting algorithms were presented by [10].

Many diagnosis approaches have been focused to the diagnosis of the induction motors. These motors are very common in industry due to their simplicity, rugged structure, cheapness and easy maintainability. It is very usual that these motors were involved into larger industrial systems. Fault detection and diagnosis of these motors are very important when they are working in on-line monitor conditions. Several techniques has been applied in order to diagnose these motors: techniques based on the signal analysis [11], based on the dynamic modelling of the motor [2] and knowledge based techniques. Within the knowledge based techniques, many tools have been used: expert systems [4], neuronal networks [7] or automatic classification [5].

It is habitual that these motors work in a iterative mode among several known set points. Also, it will be desirable does not add additional sensors to diagnose the system. This is the case treated in the present paper. The goal is to diagnose a DC Motor faults at known transient states. With this purpose, a set of classification rules must be obtained. This classification rules will be used when the system is being monitored to obtain the diagnosis. Classification rules are obtained by boosting of a previous obtained fault modes database. A complete methodology is proposed for this purpose.

This is an important improvement to the methodology presented in a previous work [1]. The complete methodology is illustrated with simulations of a DC motor. Finally, eight different faults have been considered in this system and the results will be discussed.

2 General Methodology for Diagnosing Dynamic System by Classification

In this section, a general methodology for diagnosing dynamic systems will be exposed. First a set of assumptions must be considered and it is necessary to specify the definitions and notations used to explain the methodology.

2.1 Assumptions

The following assumptions are necessary to consider in the present methodology:

- The system will be alternating between steady states and transient states.
- The number of transient states is finite and they are always produced between two known set points.
- When the system begins a transient state, if a fault exists it will be present fully since beginning of the transient to the steady state.
- Faults only will be detected at transient states, independently of the time instant in which the fault occurs.
- All fault conditions must be observed first, before diagnostics can proceed.

2.2 Definitions and notation

In order to clarify the phases of the proposed methodology, the following definitions are necessary

Definition 1. Trajectory. A *trajectory* can be defined as a function *s* from a set of time instants *T* to a set of a system sensor values $V \subseteq I\!R$.

$$s : T \rightarrow V$$

where the values of *T* are to regular intervals, from the initial instant t_0 to the final instant t_n

$$T = [t_0, t_0 + \Delta T, t_0 + 2\Delta T, ..., t_0 + (n-1)\Delta T, t_n]$$

t_0 denotes the beginning of a transient state, and t_n denotes the establishment point. The element *i* of a *trajectory* will be denoted as s_i. Thus a *trajectory j* will be represented as:

$$j = [j_0, j_1, ..., j_n]$$

Definition 2. Labelled Trajectory. A *labelled trajectory* is defined as a *trajectory* in which a label has been added. The value of the label is added as the last element in the *trajectory* and it represents the conditions in which the *trajectory* has been obtained. Thus a *labelled trajectory* jl will be represented as:

$$jl = [jl_0, jl_1, ..., jl_n, LABEL]$$

where label is a string(discrete value).

Definition 3. Trajectories Database. A *trajectories database* is a collection of *trajectories*, belonging to the same system and corresponding to the same time instant. All *trajectories* must have the same numbers of elements. This collection of *trajectories* must be stored in a file, where each *trajectory* is represented by one line.

Definition 4. Labelled Trajectories Database. A *labelled trajectories database* is a *trajectories* database where all *trajectories* are *labelled trajectories*.

2.3 Phases of the methodology

The proposed general methodology has two phases clearly different:

1. The first phase (fig. 1) is developed off-line. In this phase the main objective is to obtain a set of decision rules that characterize the system behaviour in the different fault modes which want be diagnosed.

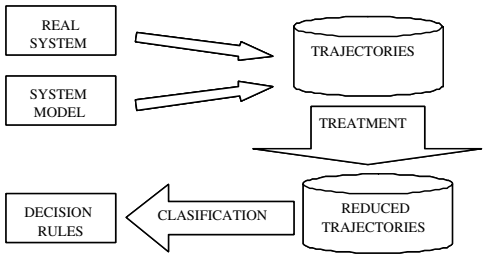

Figure 1. Off-line phase of the proposed methodology

2. The second phase (fig. 2) is the diagnosis phase itself. It is developed on-line, while the system is being monitored. In this phase the generated rules are evaluated with the system sensor measurements in order to obtain a system diagnostic.

Figure 2. On-line phase of the proposed methodology

2.3.1 Off-line phase

The first task of this phase is to select the set of faults to diagnose, and the transient states at which these faults will

be detected. Since the diagnosis process will be performed at known transient states, the process below described must be performed for all transient states in which the diagnosis process must be performed. In this phase there are three consecutive steps: data gathering, data treatment and decision rules generation.

Data gathering

In this step the goal is to obtain a collection of system *trajectories*, for each fault mode and for the free fault running. Obtaining the free fault *trajectories* is not a problem usually, but gathering the fault *trajectories* is another matter. For this purpose, two different options are considered:

- When it is possible to generate the modes, without damaging the system, they are provoked and the *trajectories* are gathered. This is, for example, the running mode in which a system component is accidentally disconnected. Also, it is possible to use this method when the system will be built in a series production. In this scenario, a set of prototypes could be damaged to provoke the faults and the *trajectories* could be obtained. Therefore, the diagnostician will be useful for all products in the series.

- When the fault cannot be provoked in the real system (because the system could be damaged or because it is impossible to stop the running system), a system model must be generated. Then, fault modes, which cannot be provoked in the real system, are simulated in the model, in order to obtain the corresponding *trajectories*.

These two options are not excluding, and both can be used at the same time. For each collected *trajectory*, a representative label will be added. This label represents the running mode of the system in which the *trajectory* has been obtained: a concrete fault or free fault. Therefore, each *trajectory* is transformed in a *labelled trajectory*. All *labelled trajectories* are stored in a labelled database. The result is a *labelled trajectories database*.

Data treatment

Usually, most of the gathered data are not relevant to distinguish among different behaviours. For example, data of *trajectories* which are closer to the steady state will be very similar, even if *trajectories* belong to different behaviours.

In this step, the aim is to reduce the amount of data that represents each *trajectory*. It is performed by means of a segmentation algorithm. The goal of this algorithm is to characterize a *trajectory* by a succession of linear segments. This succession of linear segments approximates the *trajectory* with many less points than the original *trajectory*. In [6] a comparative among segmentation algorithms can be found. In our case, the selected algorithm is the sliding window algorithm 1. The reason for this selection is that it will be needed to perform this segmentation on-line in the next phase of the methodology.

Table 1. Segmentation on-line algorithm

```
Algorithm Seg_TS = Sliding_Window(T ,max_error)
fix = 1;
while not finished segmenting time series
  i = 2;
  while calculate_error(T[fix: fix + i ]) < max_error
      i = i + 1;
  end;
  Seg_TS =concat(Seg_TS,make_segment(T[fix:fix+(i-1)]);
  fix = fix + i;
end;
```

In this algorithm, a segment is growing until it exceeds a determine error bound. When the error bound is exceeded, a new segment begins to grow. The error is calculated using the expression in equation 1.

$$Error = \sqrt{\sum_{i=1}^{n} \frac{(x_i - s_i)^2}{n}} + \lambda \max |x_i - s_i| \quad (1)$$

The first term of the expression 1 represents the deviation between the *trajectory* and the segment. The second term ensures that the deviation in any point is not greater than error. Lambda factor lets pondering each term.

In the segmentation algorithm, each *trajectory* has been approximated with few segments, but the resulting database is not a *labelled trajectories database*, because all *trajectories* in database have not the same number of elements. After segmentation process, each *trajectory* is represented with a different amount of segments, and these segments start and finish in different time instants. In order to solve this situation, and recover the *labelled trajectories database*, new segments must be generated to homogenize the numbers of elements in each *trajectory*. This will be performed with a normalization algorithm 2.

Each *normalized trajectory* is saved in a new *normalized trajectories* database and the label of *trajectory* is added. Generating new segments from a existing segments is very easy. Only it is necessary to applied the linear equation $y = mx + n$. m and n are calculated for the current segment, and a new y is generated for the new timestamp x.

The normalization algorithm returns a new database in which all *trajectories* have the same number of elements and they correspond to the same timestamp. This new database fulfils the definition of *labelled trajectories database*.

Table 2. Normalization on-line algorithm

```
Algorithm Normalized_DB = Normalize(Segmented_DB)
timestamps = {}
trajectory = read from Segmented_DB
while ∃ trajectories in Segmented_DB
   for each element in trajectory
     if timestamp(element) ∉ timestamps
       timestamps = timestamps ⋃ timestamp(element)
     end
     trajectory = read from Segmented_DB
   end
 end
go top of Segmented_DB
trajectory = read from Segmented_DB
while ∃ trajectories in Segmented_DB
 for each element in timestamps
   if element ∄ in trajectory timestamps
     generate a new segment with element
   end
   add new element to normalized trajectory
 end
 save normalized trajectory to Normalized_DB
 trajectory = read from Segmented_DB
end
```

Many times, fault modes will be more differentiable by the waveform of the evolution of a sensor measurement than the read values by the sensor themselves. In the way that our waveforms are represented by a succession of linear segments, it is important to know how steeply the segments go up or down. With that purpose, the slope of each segment will be calculated and added to the *segmented and normalized labelled trajectories database*.

The cost that must be paid in normalization process is that the new *normalized trajectories* have more segments than the original ones. The increasing is duplicated when slopes of each segments are added, but it is unavoidable if we want use this information. In any case, the global process has reached to reduce the dimension of the training database. This is possible because when trajectories are closer of the steady state, all of them are very similar, and segmentation process reduces a lot of information.

Decision Rules generation

The resulting database of the previous step fulfils all conditions to apply to them a classification tool. Thus, the *normalized and segmented labelled trajectories database* is used as training set with the selected classification tool. The output of this step is the set of generated decision rules.

In order to improve the accuracy of the learning algorithm, boosting will be used. There are two ways of doing boosting:

- Heterogeneous combination of classifiers. Several different classification methods are combined.
- Homogeneous combination of classifiers. An unique classification method is used. This method is named *method* or *base classifier*.

Any way, result of the classification process will generate as rule sets as the number of classifications performed (fig 3).

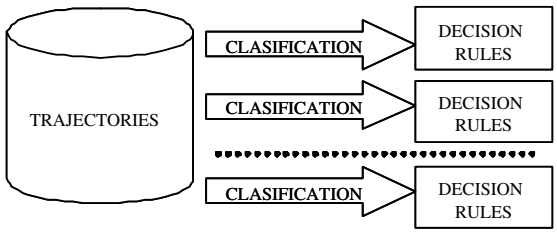

Figure 3. Boosting classification

2.3.2 On-line phase

The on-line phase of the methodology consists of evaluating an observation, of the sensor values of the monitored system, with the set of decision rules obtained in the off-line phase. In order to compare observed data with decision rules, the same treatment must be applied to observed data. Therefore, it is necessary to perform the on-line segmentation of the *trajectory* which is being observed. This is the reason because on-line segmentation algorithm was selected. Also, it is necessary to generate new segments for the normalized timestamps and calculate the slope of each segment. After this treatment, observed values will be directly comparable with decision rules . All process has low computational cost, and due to this the on-line process can be quickly performed.

As the boosting has been used to generate several sets of decision rules, the new observation needs to be evaluated with all of the classifiers obtained in the off-line process (fig 4). The result will be as classifications as existing classifiers. In order to obtain a unique result, a voting among all results is done. This voting can be as simple as selecting the more voted class, or it is possible to ponder each classifier vote with some criteria.

Output of the voting is the class corresponding to behaviour of the system. This label indicates what is happening in the system.

Because a different set of decision rules have been generated for each transient state, it is necessary to know the current setting point and the new reference, in order to select the appropriate set of decision rules.

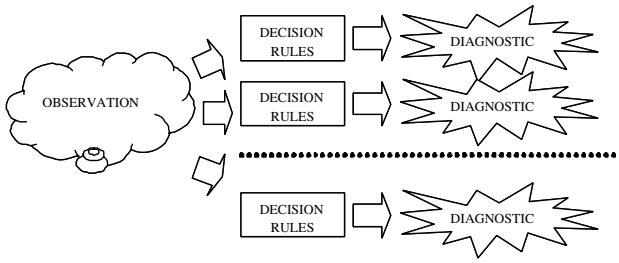

Figure 4. Evaluation with boosting

3 Application to a Chopper Controlled DC Motor

3.1 System description

The proposed methodology has been applied to a separately excited DC motor which is supplied by a three phase rectifier circuit. The DC motor is fed by the three phase rectifier through a chopper that consists of a IGBT transistor, and a free-wheeling diode. The motor torque is controlled by the armature current, which is regulated by a current control loop. The motor speed is controlled by an external PI controller, which provides the current reference for the current control loop (fig. 5).

The DC motor drives a mechanical load, that is characterized by the inertia J, friction coefficient B and load torque T_l. This motor performs a recurrent work, running from the stop state to reach the reference, always with the same load torque. This could be the scenario of a motor driving a centrifugal pump. This system has been implemented with Simulink® using components of the SimPowerSystems® Toolbox (fig. 5).

The DC motor is represented by its equivalent circuit (fig. 6), consisting of inductor L_a and resistor R_a in series with the counter electromotive force (EFM) *E*.

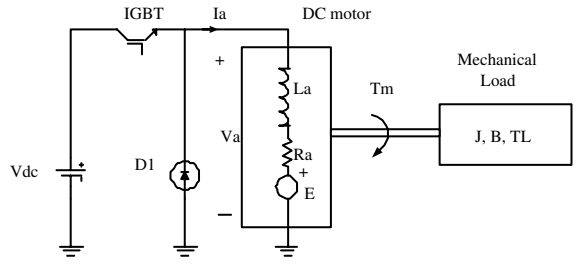

Figure 6. DC Motor equivalent circuit

The DC source (V_{dc}) is the output of the three phase rectifier circuit. The average output voltage is obtained by (2).

$$V_{AV} = \frac{3}{\pi} V_M \sin \frac{\pi}{3} \tag{2}$$

3.2 Faults isolation

The fault modes to diagnose are divided in two types:

Disruptive Faults. When these faults occur, the faulty component stops totally. For the present system, the following disruptive faults will be considered:

- One phase fault. One phase voltage is missing.
- Two phases fault. Two phases voltage are missing.
- IGBT disruption. The IGBT transistor is always on.
- Free-wheeling diode is short-circuiting.
- Free-wheeling diode is open-circuiting.

Ageing Faults. This type of faults occurs when components are loosing its properties gradually, and this produces a progressive alteration in the working system. Next ageing faults will be considered:

- Some turns in the armature winding are short-circuited. This produces a decrement in the inductance and resistance of the winding. The rate between the number of short-circuited turns and the decrease in resistance and inductance are given by equations 3 and 4.

$$\Omega = K \frac{L}{S} \tag{3}$$

$$L = \frac{\mu_r \mu_0 N^2 A}{L_c} \tag{4}$$

- Some turns in the field winding are short-circuited. The fault is the same that the previous one, but applied to the field winding.
- Bearing friction increasing. An excessive bearing friction, due to stress, produces an increasing in the friction torque.

3.3 Off-line phase

In order to obtain the *labelled trajectories database*, a set of simulations have been done with the above described system. Transient state simulated is from stop state to a reference of 50 rad/sec. Parameters for simulations are shown in table 3

A tolerance interval is considered in nominal values of each component. These components are considered free of fault if its value persists inside this interval. This represents

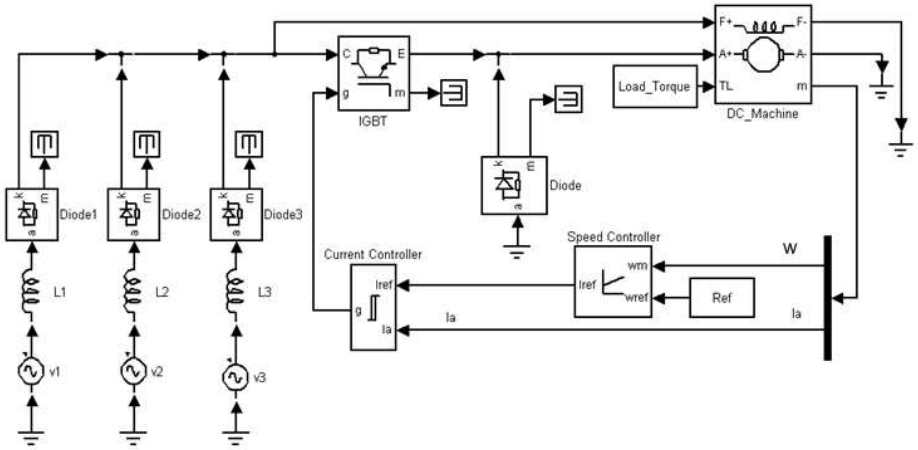

Figure 5. Simulink® model of the system

Table 3. Simulation parameters

Nominal Power	3.9	KW
Armature resistance	0.6	Ω
Armature inductance	0.0012	H
Field resistance	240	Ω
Field inductance	120	H
Mutual inductances	1.8	H
Friction torque	0.5	N.m
Load torque	10	N.m
Inertia	1	$Kg.m^2$
Viscous friction coefficient	0.05	N.m.s
PI Controller Proportional Gain	3	Kp
PI Controller Integral Gain	0.5	Ki
PI Controller current limit	200	A
Current controller hysteresis band	3	A

the little alterations that can be produced in a system without a fault is considered. Tolerance rate has been established in a 4% of the nominal value of the components.

Each disruptive fault simulation has been performed by changing the system model to produce the fault.

For ageing faults, the nominal values of the faulty component are changed. For short-circuiting of turns in windings, a decrease of inductance and resistance until 40% has been considered. Friction torque rising to 10 N.m. has been considered for bearing fault.

Thirty simulations per behaviour have been done by Montecarlo selection of interval values. Data gathered corresponds to readings of the angular speed of the rotor (ω) and the control signal I_{ref} (Fig. 5).

After obtaining the simulations, the segmentation and normalization process (tables 1 and 2) has been applied to *labelled trajectories database*. With the aim of increasing the available information, the value of the slope of each segment has been calculated after the normalization algorithm. These values have been added to the segmented and normalized *labelled trajectories database*.

Finally, the classification tool C5.0® has been applied to the segmented and normalized *labelled trajectories database*.

The considered options have been without boosting and with boosting. A single decision rules tree is obtained without boosting. The size of the single tree is 10 nodes. In the other side, three boosting trials have been tried and three decision rules trees have been obtained. The size of the obtained trees have been 13,11 and 9 nodes.

3.4 On-line Phase: Results

In order to evaluate the validity of the methodology, 10 simulations per behaviour have been performed. The new simulations have been treated to be evaluated with the generated decision rules. This means that the segmentation and normalization algorithms have been applied, and the slopes of the segments have been calculated. For evaluating the new simulations with the boosting classification, a equality voting is performed among the three generated trees. The obtained results with respect to the single classification tree algorithm are shown in table 4.

As can be found in results table, never it is diagnose a fault when it is not happening. In other words, never a false positive is given.

It is important to highlight that faults F2 and DS are non-discriminable, because both give the same sensor readings. This causes that both evaluations return an erroneous diagnostic when the fault F2 is simulated. The rest of disruptive faults are always correctly detected with both options.

Table 4. Diagnosis results

Fault	Behaviour	Single tree diagnosis	Boosting diagnosis
(OK)	No Fault	100% OK	100% OK
(AW)	Armature Windings	80% AW 20% OK	90% AW 10% OK
(FW)	Field Windings	90% FW 10% BF	90% FW 10% BF
(BF)	Bearing Friction	90% BF 10% FW	100% BF
(F1)	1 Phase Fault	100% F1	100% F1
(F2)	2 Phase Fault	100% DS	100% DS
(IG)	IGBT Fault	100% IG	100% IG
(DS)	Diode Short-circuit	100% DS	100% DS
(DO)	Diode Open-circuit	100% DO	100% DO

With respect to the ageing faults the diagnosis is generally more imprecise. This imprecision is because sensors readings are very similar when ageing values are very closer to the correct ones. For example, AW fault is not detected in 20% of cases when it is evaluated with the single decision tree, and in the 10% of cases when it is evaluated with boosting. Also, the faults FW and BF are mistaken when the sensor readings are very similar. Nevertheless, boosting classification improves the obtained results by the single classification tree.

4 Conclusions and Future Works

An automatic and general methodology to diagnose dynamic systems by boosting has been proposed. The proposed technique is a general method for improving the accuracy of any given learning algorithm and it has no parameter to tune. The presented segmentation and normalization algorithms are able to reduce significantly the dimension of the gathered data to treat.

This methodology is able to diagnose dynamic systems when they are running in transient states that have been previously trained.

Models are not necessary when experimental data are available; they are only used when real data are not available.

Obtained results with the chopper controlled DC motor are very accurate, and it have been proven that boosting is able to improve the results of a single classification.

When the rules have been generated, the diagnosis process is very simple, and it can be performed with a linear computational time with respect to the numbers of nodes in the tree. This simplicity allows that it can be implemented with low cost components.

Currently this methodology is been applied to a real implementation of the modelled system. Some different faults are being added, as eccentricity fault or brush fault. Also, improvements in data treatment and classification step are being tested.

5 Acknowledgments

The authors gratefully acknowledge the contribution of the *Ministerio de Ciencia y Tecnologia de España* (DPI2003-07146-C02-01) and the European Regional Development Fund (ERDF/FEDER).

References

[1] P. J. Abad, A. J. Suárez, R. M. Gasca, and J. A. Ortega. Using supervised learning techniques for diagnosis of dynamic systems. In *13th International Workshop on Principles of Diagnosis*, Semmering, Austria, 2002.

[2] C. W. Chan, K. C. Cheung, H. Y. Zhang, and Y. Wang. Fault detection of dc-motors using non-linear observer based on current b-spline neurofuzzy network. In *Proc 14th IFAC World Congress*, volume B, Beijing, China, 1999.

[3] C. Feng. Inducting temporal fault diagnostic rules from a qualitative model. In *Inductive Logic Programming*. S. Muggleton, editor. Academic Press, 1992.

[4] F. Filippetti, M. Martelli, G. Francesschini, and C. Tassoni. Development of expert system knowledge base to on-line diagnosis of rotor electrical faults of induction motors. In *Conf. Rec. 27th IEEE-IAS Annu. Meeting*, Oct 1992.

[5] M. Hajiaghajani, H. A. Toliyat, and I. M. S. Panahi. Advanced fault diagnosis of a dc motor. *IEEE Trans. On Energy Conversion*, 19(1), March 2004.

[6] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. *IEEE Intl. Conf. Data Mining*, pages 289–296, May 2001.

[7] X. Q. Liu, H. Y. Zhang, J. Liu, and J. Yang. Fault detection and diagnosis of permanent-magnet dc motor based on parameter estimation and neural network. *IEEE Trans. On Industrial Electronics*, 47(5), October 2000.

[8] C. J. LopezToribio, R. J. Patton, and F. J. Uppal. Artificial intelligence approaches to fault diagnosis for dynamic systems. *Applied Mathematics and Computer Science. Special Issue: Advances in Computational Intelligence for Fault Diagnosis Systems*, 9(3):471–518, 1999.

[9] D. Roverso. Fault diagnosis with the aladdin transient classifier. In *System Diagnosis and Prognosis: Security and Condition Monitoring Issues III*. AeroSense2003, Aeroespace and Defense Sensing and Control Thecnologies Symposium, 2003.

[10] R. E. Schapire. The boosting approach to machine learning: An overview. In *Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA, March 2001.

[11] R. R. Schoen, T. G. Habetler, F. Kamran, and R. G. Bartheld. Motor bearing damage detection current monitoring. *IEEE Trans. Ind. Applicat*, vol. 31, Nov-Dec 1995.