분류패턴과 신경망을 이용한 시스템의 고장진단

論 文
49D - 12 - 1

Fault Diagnosis for a System Using Classified Pattern and Neural Networks

李 鎭 河* · 朴 省 昱** · 徐 輔 烇***
(Jin-Ha Lee · Seong-Wook Park · Bo-Hyuk Seo)

Abstract - Using neural network approach, the diagnosis of faults in industrial process that requires observing multiple data simultaneously are studied. Two-stage diagnosis is proposed to analyze system faults. By using neural network, the first stage detects the dynamic trend of each normalized date patterns by comparing a proposed pattern. Instead of using neural network, the difference between stored fault pattern and real time data is used for fault diagnosis in the second stage. This method reduces the amount of calculation and saves storing space. Also, we dealt with unknown faults by normalizing the data and calculating the difference between the value of steady state and the data in case of fault. A model of tank reactor is given to verify that the proposed method is useful and effective to noise.

Key Words : Fault diagnosis, Fault Detection, Pattern, Neural Network

1. 서론

최근 산업의 발전과 더불어 시스템이 복잡해지고 빨라지게 되었으며, 이러한 시스템의 신뢰성과 안정성을 유지하는 일이 무엇보다도 중요한 일로 대두되었다. 이를 위해서는 그 시스템에서 발생되는 고장에 대한 보다 빠르고 구체적인 정보를 제때에 얻어야 하며, 이를 활용하여 고장으로 인한 여러 가지 손실을 최소화하여야 한다. 이러한 이유로 최근에는 고장진단 분야에서 많은 연구들이 이루어지고 있다.

고장진단 기법은 여러 가지가 있으나, 일반적으로 수학적 모형을 기반으로 하는 기법과 수학적 모형을 기반으로 하지 않는 기법으로 크게 나뉘어지며, 수학적 모형을 기반으로 하는 기법은 정량적 모형을 기반으로 하는 기법과 정성적 모형을 기반으로 하는 기법으로 나뉘어 진다[1].

수학적 모형을 기반으로 하는 기법은 기본적으로 정확한 모형화가 요구되며, 모형화가 정확하지 않을 경우, 많은 문제를 야기한다. 이 중 정량적 모형을 기반으로 하는 기법은 수학적 모형에 근거하여 데이터를 분석함으로써 고장을 진단하는 해석적 기법으로 동적 시스템의 초기고장 검출에 유용하다. 정성적 모형을 기반으로 한 기법은 시스템 내의 변수간의 상호 작용과 고장의 전파과정 등을 인과관계 표현에 적합한 방향성 그래프, 퍼지 인식맵, 패트리 네트, 사건 트리 등을 사용하여 표현한 결과이며 모두 고장진단을 위해 사용되고 있다[2].

수학적 모형을 사용하지 않는 고장진단 기법으로는 프로세스의 변수 또는 파라미터를 측정기록하고 이들의 통계적 특성에 대한 가설검사에 의해 고장을 검출하는 신호검증기법, 프로세스의 변수 또는 파라미터의 측정치를 도표를 이용하여 관찰함으로서 동적 시스템의 상태를 감시하는 관리도 기법, 진동 및 음향의 주파수 결과에 의해 기계장치의 동작상태를 감시하는 기법 등이 있다[3].

본 논문에서는 수학적모형에 기반을 두고, 그 공정에서 시간에 대해서 순차적으로 나오는 데이터를 패턴으로 생각하여 이를 신경망으로 분석함으로써 고장진단을 수행한다.

신경망과 패턴인식을 결부시켜 고장진단을 하는 방법은 여러 가지 장점들이 있다. 기존의 논문 중에 Yunosuke의 논문[6]에서 이러한 내용들을 다루고 있다. 실제 플랜트에 대해서 정확한 모델링을 하지 않더라도, 그 플랜트의 정상상태에서의 데이터 값과 고장시의 데이터만을 가지고 있다면 이를 이용해서 고장진단을 수행할 수 있을 뿐만 아니라 구현도 비교적 간단하다. 또 정상상태가 아닌 과도적인 상태에서 고장진단을 행하기 때문에 다른 고장진단 시스템에 비해서 빠른 시간 내에 고장을 진단 할 수 있다.

하지만 이 방법은 기준 패턴이 세분화되지 않아서 여러 가지 다양한 패턴이 들어왔을 때 그 값들을 정확히 구별해 내지 못한다. 또 윈도우가 시간에 대해서 뿐만 아니라, 그 때 샘플링한 데이터의 평균치를 정상상태로 설정하고 상하로도 움직이기 때문에 오랜 기간 조금씩 변화하는 고장이나 고장 자체가 정상상태에 도달할 경우 발견해 내기가 힘들다. 진단 후반부에서는 고장패턴을 저장하고 진단하는 부분까지 신경망을 이용했기 때문에 새로운 패턴의 추가가 힘들 뿐 만 아니라, 실제 신경망의 학습을 이용하여 패턴을 저장할 때도 그 수렴을 보장하기 힘들며, 수렴하게 하려면 많은 층과 노드

* 準 會 員 : 慶北大 電氣工學科 博士課程
** 正 會 員 : 구미1大 電氣科 助敎授
*** 正 會 員 : 慶北大 電子電氣學科 敎授 · 工博
接受日字 : 2000年 9月 30日
最終完了 : 2000年 11月 10日

를 가진 신경망을 필요로 한다. 또 예측하지 못한 고장에 대한 대책은 전혀 없다.

본 논문에서는 이러한 단점을 보완하기 위하여 기준패턴을 보다 세분화하였고, 윈도우가 시간에 대해서는 움직이지만, 데이터 값에 대해서는 움직이지 않도록 하였다. 이를 통하여 오랜 시간 서서히 변해 가는 고장이나 예측하지 못한 고장에 대한 대책을 마련하였다. 진단 후반부에서는 신경망으로 고장패턴을 저장하고 진단하는 대신 단순한 저장공간에 패턴을 나타내는 수치를 저장하고 이를 비교하여 고장을 진단하도록 하여, 새로운 패턴의 추가가 쉽고, 수렴성에 대한 단점을 보완 하였다.

2. 고장진단 알고리듬

고장진단 알고리듬의 대략적인 구조는 그림 1과 같다. 대상이 되는 공정의 각 부분에서 데이터를 획득한 다음, 이 데이터를 일정시간동안 저장하게 된다. 저장된 데이터를 정규화 하여 데이터가 신경망을 통과하면서 저장된 데이터의 변화추세를 이미 학습된 기준 패턴과 비교하여 3가지의 수치로 나타낸다. 이렇게 나타낸 변화추세는 이미 분류기의 패턴저장영역에 저장된 고장패턴과 비교함으로써, 고장을 진단하게 된다. 또 정규화 된 데이터는 정상상태에서 벗어나는 정도, 즉 편차를 고려하여 예측하지 못한 고장을 검출하는데도 이용된다. 그림 1에 대한 자세한 내용은 2.1 - 2.5에서 다루겠다.

2.1 데이터의 획득

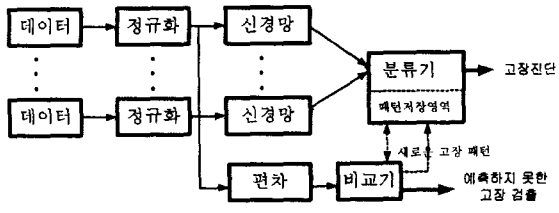

그림 1. 고장진단 알고리듬

Fig. 1. Fault diagnosis algorithm

공정의 각 부분에서 나오는 데이터는 그 공정의 고장을 판단하는 중요한 근거가 된다. 일반적으로 공정 전체의 고장이 없을 경우, 각 부분의 데이터는 정상운전 상태로 남아 있을 것이다. 하지만 어떠한 원인에 의해서 공정의 한 부분에 고장이 발생하게 될 경우, 고장이 발생한 부분은 정상적인 데이터를 내 놓지 못하게 될 것이다.

고장이 발생한 부분의 성질에 따라 차이는 있으나, 고장이 발생하게 되면, 고장난 부분의 데이터가 여러 가지 형태로 이상을 나타내게 된다. 일반적으로 정상동작상태에서의 데이터는 거의 일정한 값으로 정상상태를 유지한다. 하지만 고장이 생기게 되면, 고장의 종류에 따라서 차이는 있으나, 대체로 짧은 시간동안 변화를 일으키다가 새로운 평형 점에 도달하여 비정상적인 정상상태를 유지하게 된다.

또 이러한 한 부분의 고장은 다른 부분에도 여러 가지 파급 효과를 가져오게 된다. 그래서 한 부분의 고장은 여러 부분에서 각 부분 데이터에 여러 가지 문제를 일으킨다.

공정의 각 부분에서 나오는 데이터들을 시간에 대해서 분석해 보면 일정한 패턴의 형태로 생각을 할 수 있다. 즉, 시간에 대해서 일정 길이의 데이터를 분석해 보면 이는 곧 일정한 형식의 패턴이다.

이러한 패턴의 형태를 잘 분석하면 고장의 종류를 파악하는데, 중요한 자료가 될 수 있다. 더구나 한 곳의 고장은 다른 여러 곳에서 파급효과를 일으키는 경우가 많다. 그러므로 한 곳의 고장패턴만을 관찰하는 것이 아니라 여러 곳에서 발생하는 패턴을 동시에 파악한다면, 보다 정확한 결과를 얻을 수 있다.

데이터는 고장을 진단해야하는 공정에 따라서 샘플링 할 시간 간격이나 데이터 개수가 정해진다. 일반적으로 긴 시간동안 많은 데이터를 샘플링하면 할수록 갑작스런 잡음이나 외란에 강하고 비교적 정확한 고장진단이 가능하다. 하지만 데이터를 너무 긴 시간동안 많은 데이터를 고려하게 되면 계산량이 많아질 뿐만 아니라 검출 시간도 늦어지게 된다.

그래서 본 논문에서는 고장발생을 조기에 진단하기 위하여 일반적으로 정상상태에서 분석하는 방법을 취하지 않고, 고장이 시작되어서 정상상태로 진행하기 전의 과도기적인 패턴을 고장패턴으로 받아들여서 저장하는 방법을 이용하였다. 이렇게 해 줌으로써 보다 빨리 고장진단을 수행할 수 있다.

2.2 데이터의 정규화

고장패턴의 경향을 분석할 때, 평균적으로 나오는 데이터의 값은 아주 큰데 반해서, 데이터의 변화 범위가 작은 것이 있을 수 있다. 이는 데이터의 변화 형태를 패턴화하고, 그 패턴을 관찰해서 고장진단을 수행하는 데는 바람직한 데이터의 형태가 아니다.

그러므로 조금 변화된 형태의 데이터가 필요하다. 실제 데이터의 변동분만을 고려하는 형태의 데이터가 있어야 할 것이다. 그러한 형태로 데이터를 바꾸어 줌으로써 공정 각 부분의 데이터 출력이 변함에 따라 패턴의 변화도 뚜렷하게 관찰 할 수 있을 것이다.

본 논문에서는 데이터를 실제로 -1과 1 사이에서 정규화 시킨다. 이 때 정규화 하는 식은 아래와 같다.

$$d_n(t)=\frac{d(t)-d_s}{d_{\max}-d_s} \qquad (1)$$

$d(t)$는 공정의 각 부분에서 시간이 지남에 따라 발생되는 데이터이며, d_s는 평상시 정상상태의 데이터, $d_{\max}$ 는 고장시 최대로 변할 수 있는 값이다. $d_{\max}$ 는 실제로 정확한 값으로 구하기는 어렵다. 하지만 꼭 정확한 값을 구해야 할 필요는 없다. 최대치보다 조금 더 높게 잡아 주어도 시간에 대한 패턴을 나타내는데는 지장이 없기 때문이다.

식 (1)과 같이 데이터를 정규화하여 다음 단계에서 거치게 될 신경망의 입력을 일관되게 만들 수 있다. 또 입력을 그렇게 함으로 수렴이 확실히 보장되지 않는 신경망의 학습을 여러 변수의 범위에 대해서 각각 고려하지 않고, 여러 변수의 입력을 한 번의 학습으로 가중치가 설정된 단 하나의 신경망으로 분석할 수 있다. 또 식 (1)의 정규화 과정으로 이 후에

소개될 예측하지 못한 고장에 대한 검출도 가능하게 하였다.

2.3 신경망의 학습

정규화 된 데이터가 신경망을 통과하게 되면, 그 데이터는 숫자로 바뀌게 된다. 즉, 데이터가 그 데이터의 패턴을 나타내는 고유의 숫자로 바뀌게 되는 것이다. 이는 수많은 데이터를 일일이 저장하거나 데이터 전체를 가지고 비교하는 것이 아니라, 그 데이터를 나타내는 숫자 즉, 신경망으로 패턴을 분석한 값으로써 데이터를 비교하게 하는 것이다.

정규화 된 데이터를 숫자로 바꾸기 위해서는 어떤 기준에 의해서 신경망을 통과시켜야 일관된 패턴분석이 가능할 것이다. 그래서 우선 패턴 분석에 필요한 기준 패턴을 만들어서 그 패턴을 기준으로 신경망을 학습시킨다. 그런 다음, 정규화 된 데이터를 신경망에 통과시킴으로써 패턴을 적절한 형태의 수치로 변화시킬 수 있다.

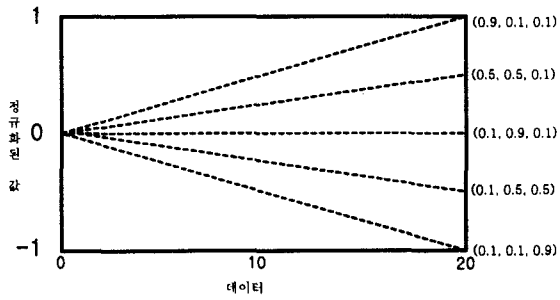

그림 2. 학습해야할 기준 패턴
Fig. 2. Standard pattern to learn

기준이 되는 패턴은 증가정도, 감소정도, 데이터가 일정하게 유지되는 정도에 따라서 우선 3가지의 기준 패턴을 설정한다. 그리고 그 사이에 2개의 기준패턴을 더 추가하였다. 입력 데이터는 -1과 1 사이이며, 출력은 0과 1사이의 값이다. 신경망의 출력은 3개의 숫자로 나타나는데, 첫 번째 숫자는 증가정도, 두 번째 숫자는 일정 상태는 유지하는 정도, 세 번째 숫자는 감소 정도를 나타낸다. 즉, (1, 0, 0)은 가장 증가정도가 큰 패턴이며, (0, 0, 1)은 가장 감소 정도가 큰 패턴이다. 또 (0, 1, 0)는 일반적인 정상상태를 말한다. 하지만 실제 학습에서는 신경망의 수렴의 용이성을 위해서 그 수치를 각각 0.9와 0.1로 설정하고 기준 패턴을 학습을 시켰다. 학습법은 널리 알려진 역전파법[4,5]을 이용하였으며, 그림 2는 학습해야 할 기준패턴을 나타낸다.

2.4 분류기

분류기는 크게 두 부분으로 구분된다. 그 중 한 부분은 고장 패턴을 저장하는 부분이다. 과거의 고장진단은 흔히 능숙한 전문가의 지식이나 경험에 많이 의존했다. 그런 관점에서 본다면 이 부분은 전문가의 지식이나 경험과도 같은 부분이다. 전문가의 지식이나 경험을 데이터의 형태로 저장하는 부분이다.

또 다른 부분은 실제로 고장을 진단하는 부분이다. 이는 실시간으로 들어오는 데이터의 패턴과 저장된 패턴과의 유사성을 비교하는 부분이다. 신경망을 학습시킬 때 오차는 계산상의 편의를 위해서 제곱의 형태를 사용했지만, 분류기의 패턴 비교에서의 오차는 단순히 두 패턴 사이의 유사성만을 비교하면 되기 때문에 단순한 절대값을 사용했다. 이를 수식으로 나타내면 식 (2)과 같다. 만일 저장된 패턴이 여러 개라면 각 패턴에 대해서 식 (2)와 같은 연산을 저장된 패턴의 수만큼 수행해야 할 것이다. 식 (2)에서 s는 각각의 저장된 패턴을 나타내기 위한 첨자이다.

$$D_s(t) = \sum_{i=1}^{3n} | f_{si} - y_i(t) | < \delta_s \qquad (2)$$

$$\mathbf{f_s} = \begin{bmatrix} f_{s1} \\ f_{s2} \\ \vdots \\ f_{s3n} \end{bmatrix} \qquad (3)$$

$$\mathbf{y(t)} = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_{3n}(t) \end{bmatrix} \qquad (4)$$

식 (2)에서 f_{si}는 저장된 고장패턴의 각 원소이며, 임의의 한 패턴에 대한 고장 패턴은 식 (3)과 같이 벡터로 표현된다. 식 (3)과 식 (4)에서 n은 실제 고장진단에 이용되는 데이터를 출력하는 부분의 수 즉, 각각의 패턴을 발생시키는 데이터의 수를 나타낸다. 각 부분마다 증가, 감소, 일정 정도에 따라 3개의 데이터를 만들어 내므로 저장된 고장패턴의 데이터의 수는 $3n$개가 된다. $\mathbf{y(t)}$ 는 시간에 대해서 들어오는 입력패턴이며, 그 형태는 식 (4)와 같다. δ_s는 고장인지 여부를 판단하는 문턱값이다. $\mathbf{f_s}$와 $\mathbf{y(t)}$ 사이의 차이가 작다는 것은 그만큼 두 패턴이 유사함을 나타내므로, 그 수치가 기준치 δ_s보다 작다는 것은 고장이 발생했음을 뜻한다.

여기서 문턱값 δ_s는 아주 중요한 역할을 한다. 만약 δ_s를 너무 크게 잡으면 패턴이 많아지거나, 패턴이 굳이 많아지지 않더라도 유사한 패턴이 많이 존재하게 되면, 다른 패턴과의 혼동 우려가 있다. 또 δ_s를 너무 작게 잡으면, 약간의 잡음만 발생해도 유사한 패턴을 인식하지 못할 우려가 있다. 또 같은 종류의 고장일 지라도 패턴 변화에 정도의 차이가 있을 수 있으므로 고장이 발생해도 이를 인식하지 못하는 경우가 생길 수 있으므로 δ_s를 적절히 잘 선택함으로써 고장의 적절한 진단이 가능하다.

위의 분류 알고리듬은 기존의 신경망을 이용할 때에 비하여 훨씬 더 계산량이 적음을 수치로 보여주지 않더라도 직관적으로 확인할 수 있다.

먼저 새로운 고장패턴이 발생하게 되어 그것을 저장해야 할 때, 기존의 신경망을 이용할 경우에는 앞에서 말한 것과 마찬가지로 전체의 수많은 가중치들을 새로 학습시켜야 할 뿐만 아니라 그 수많은 가중치들을 저장할 공간 또한 많이 필요하게 된다. 또 저장할 패턴이 많아지게 될 경우, 신경망의 층을 늘리거나 혹은 매개변수를 바꾸어 가면서 새로 학습시켜야 하는 단점이 있다. 또 그 학습의 수렴성 또한 보장되지 않는다. 하지만 본 논문에서 제안한 방법을 이용할 경우, 단순히 추가된 패턴을 저장만 하면 되므로 계산량이 줄어들 뿐만, 아니라 일반적으로 가중치를 저장하는 대신 단순히 수치로 기록된 패턴만을 기록하면 되므로 저장공간도 절약할 수 있을 것이다.

또 패턴을 분류하는데 있어서도 신경망을 이용할 경우, 패턴이 들어올 때마다 수많은 가중치와 활성화함수를 계산해야 하는데, 반해서 본 논문에서는 단순히 패턴의 유사성만을 각 패턴에 대해서 계산해 주면 되므로 전체적인 계산량은 훨씬 작다.

극단적인 경우를 가정해서 고장패턴의 수가 많아서 실제 비교할 데이터가 늘어나서 본 논문에서 제안한 알고리듬이 신경망에 비해서 계산량이 많아지고, 이에 반해서 신경망이 가중치의 수가 적은 경우를 생각할 수 있을지 모르나, 실제로 신경망에서 패턴 하나가 추가될 때마다 그 패턴이 수렴하도록 만들기 위해서는 그보다 훨씬 많은 가중치를 추가시키거나 혹은 층을 더 추가해야 한다. 또 그렇게 한다 하더라도 실제 수렴성을 보장하기는 힘들다.

2.5 예측하지 못한 고장의 검출

이제까지 알려진 고장진단 시스템의 가장 큰 과제중의 하나는 예측하지 못한 고장에 대한 검출이었다. 실제로 전문가가 아닌 어떤 고장진단 장치가 고장진단을 수행할 경우, 이제까지의 고장 진단은 저장영역에 고장패턴을 저장함으로써 가능했다. 하지만 이제까지 예측하지 못한 갑작스런 고장이 발생할 경우, 이에 대한 대책은 거의 없는 실정이다.

본 논문의 앞부분에서 신경망에 입력으로 들어가는 데이터를 -1과 1 사이에서 정규화 시키고, 정상상태의 값을 0으로 설정했다. 이렇게 정규화 시킨 첫 번째 이유는 앞에서 설명한 것과 같이 신경망으로 들어갈 입력을 통일시킴으로써 하나의 신경망으로 일관성 있는 패턴 분석을 하기 위해서였다.

여기에서는 데이터를 정규화 한 또 한가지의 이유를 소개한다. 평소에 정상상태를 유지하던 데이터가 정상상태를 벗어난다는 것은 역시 고장을 의미한다. 하지만 이러한 고장이 이전에 발생했던 고장이 아닌 경우에는 유사한 패턴이 없으므로 기존의 장치에서는 고장이 아닌 것으로 판단하게 된다.

그러므로 이에 대한 대책이 필요하다. 그림 1에서 고장진단 알고리듬 내에는 편차를 구하는 부분이 있다. 이를 잘 이용하면 실시간으로 진단은 불가능할 지라도 예측하지 못한 고장에 대한 검출은 가능하다.

$$\sigma_{avg}(t) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\mathbf{d}_i(t)^T\,\mathbf{d}_i(t)} > \delta_f \qquad (5)$$

$$\mathbf{d}_i(t) = \begin{bmatrix} d_{i1}(t) \\ d_{i2}(t) \\ \vdots \\ d_{ia}(t) \end{bmatrix} \qquad (6)$$

식 (5)는 각 공정의 변수에 대한 평균치를 나타낸다. 여기서 $\mathbf{d}_i(t)$는 공정 각 부분의 변수에 대해서 식 (1)로 정규화된 데이터이며 그 형태는 식 (6)과 같다. $\mathbf{d}_i(t)$ 는 고장패턴을 구할 때와 같은 시간동안 데이터를 구하되 실제 고장패턴을 구할 때보다는 적은 수의 데이터로 구성하는 것이 좋다. 실제로 편차를 구할 데이터는 패턴의 형태를 구할 때만큼 많은 데이터는 필요치 않을 것이다. 전체적으로 패턴이 정상상태를 어느 정도 벗어났는가를 파악하는 것이 중요하기 때문이다.

일반적으로 이 데이터에 이상이 생긴다는 이야기는 정규화된 데이터가 0을 벗어난다는 것을 의미하므로 편차가 크다는 것은 고장이 발생했음을 의미한다. δ_f는 고장을 판단하는 문턱 값이다.

실제로 편차가 δ_f를 넘을 때에는 그 패턴이 지금 현재 저장된 고장패턴에 존재하는 패턴인지 확인하기 위하여 고장분류기의 출력을 조사한다. 이때 고장분류기의 출력에서 이미 알려진 고장으로 출력을 나타내지 않을 경우, 이를 이제까지 예측하지 못한 고장으로 판단하게 된다.

하지만 이는 실제 고장진단이라고는 말할 수 없다. 고장진단이라는 개념은 고장검출과 분류까지를 포함하는 개념이므로 본 논문에서는 예측하지 못한 고장에 대해서는 고장검출만을 가능하게 하였다. 이렇게 저장된 고장패턴으로 고장진단까지 가능하게 하려면, 정확한 원인조사를 통해서 새로 알려진 고장패턴으로 저장해야 할 것이다.

3. 사례연구와 검토

3.1 대상시스템

본 논문에서 예로 든 시스템은 탱크 반응장치이다[6-8]. 전체 구성을 그림 3에 나타내었다. 물질 A를 주입하면 A 물질이 100% 반응할 수는 없으므로 일부는 반응을 거치지 않고 물질 A로, 일부는 반응을 거쳐서 물질 B로 변화되어서 밖으로 배출되는 장치이다. 시스템을 나타내는 식은 식(7)-식(12)와 같다.

$$\frac{dV}{dt} = F_i - F \qquad (7)$$

$$\frac{d(VC_A)}{dt} = F_i C_{Ai} - FC_A - VkC_A \qquad (8)$$

$$\rho C_P \frac{d(VT)}{dt} = \rho C_P(F_i T_i - FT) - UA_H(T - T_c) - \lambda VkC_A \qquad (9)$$

$$\rho_c V_c C_c \frac{dT_j}{dt} = F_c \rho_c C_c (T_{ci} - T_c) + UA_H (T - T_c) \quad (10)$$

$$F = 40 - 10(48 - V) \quad (11)$$

$$F_c = 49.9 - K_C(600 - T) \quad (12)$$

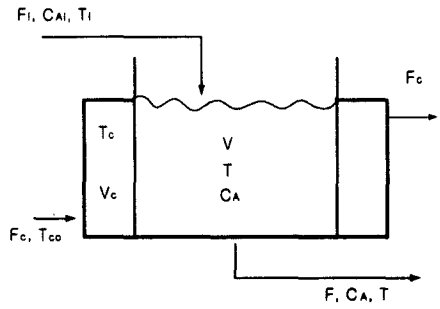

그림 3. 탱크 반응로

Fig. 3. Tank reactor

V : 탱크내부 물질의 부피

F_i : 주입구에서 주입되는 물질의 유입속도

F : 탱크 밖으로 빠져나가는 물질의 배출속도

T : 탱크내부 온도

T_{ci} : 주입되는 냉각수의 온도

T_c : 배출되는 냉각수의 온도

F_c : 냉각수의 유입속도

ρ : 물질의 밀도

ρ_c : 냉각수의 밀도

U : 단위질량당 에너지

A_H : 열전달 면적

탱크 내부의 온도는 시간에 따라 변할 수 있으며, A에서 B로 변화되는 과정은 발열반응이다. 외부로의 열 손실은 무시할 수 있다고 가정한다.

대상이 되는 시스템은 그림 3과 같이 탱크반응로이며, 이를 식으로 나타내면 식 (7)-(10)과 같다. 또 여기에 달려있는 제어기는 식 (11)-(12)와 같다.

3.2 가정한 고장

표 1은 본 논문에서 가정한 고장의 목록이다. 여기에서 가정한 고장들은 모두 8가지이며 이를 기준으로 실제로 고장 상황에 맞게 일부 변수를 변화시켜가면서 시뮬레이션을 행하였다.

표 1. 고장의 목록

Table 1. List of faults

고장 번호	증상	원인
1	냉각수 온도의 증가	냉각수 온도
2	냉각수 온도의 감소	제어 장치 고장
3	입구에서 유입 속도 증가	A 물질의 유입
4	입구에서 유입 속도 감소	속도 제어장치 고장
5	입구에서의 온도 증가	A 물질의 온도
6	입구에서의 온도 감소	제어 장치 고장
7	입구에서 A물질 농도 증가	A 물질의 투입량
8	입구에서 A물질 농도 감소	조절 장치 고장

데이터는 고장이 발생한 시점부터 0.03[h] 간격으로 모두 21개의 데이터를 샘플링하였다. 하나의 고장 패턴에 대해서 나오는 데이터의 개수는 6부분에서 각각 21개씩의 데이터를 나타내므로 모두 126개의 데이터가 된다. 하지만 실제의 경우 정밀한 고장진단을 위해 더 많은 데이터를 필요로 한다면 실제 필요한 데이터 개수는 훨씬 많아 질 수 있다. 그림 4는 가정한 고장 8가지 중, 고장 1의 고장패턴을 나타내었다.

그림 4에서 보듯이 고장의 징후가 나타나기 시작하는 시간은 4[h]부터이다. 표 1에서 고장 1의 원인은 냉각수의 온도 제어장치가 고장난 경우이다. 냉각수의 온도가 증가하면서, 공정 각 부분에서의 변수들이 식 (7)-(12)에 따라서 변화를 일으킨다. 하지만 A 물질의 농도(C_A)나 생성된 물질이 배출되는 속도(F)는 영향을 받지 않음을 알 수 있다.

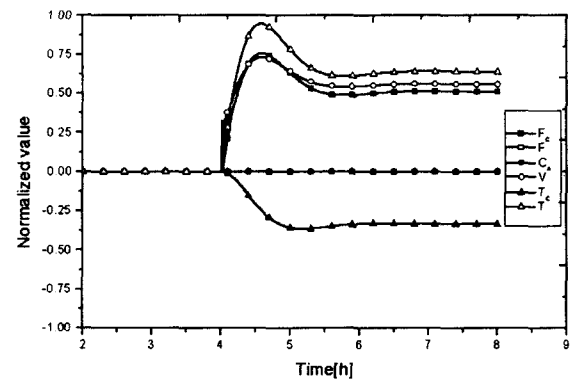

그림 4. 고장 1의 고장 패턴

Fig. 4. Fault pattern of fault #1

3.3 신경망을 통한 패턴의 분석

신경망은 이미 앞에서 말한 바 있는 역 전파법으로 학습을 시켰으며, 4층의 노드로 구성되어 있다. 각 층의 노드수는 21, 5, 15, 3개로 구성했다. 고장패턴으로 잡은 구간은 고장 발생 직후부터 0.03[h] 간격으로 21개의 데이터를 샘플링하였고, 그 21개를 패턴으로 학습시킨 신경망에 통과시켰을 때의

결과는 표 2와 같다.

표 2. 신경망에 적용된 고장패턴

Table 2. Fault pattern applied to neural network

변수	고장 1	고장 2	고장 3	고장 4
F_c	0.7990	0.0965	0.1321	0.2384
	0.1977	0.3747	0.6415	0.7665
	0.0987	0.6416	0.2978	0.0967
F	0.0983	0.0983	0.7792	0.0966
	0.8974	0.8974	0.2175	0.2387
	0.1068	0.1068	0.0987	0.7724
C_A	0.0983	0.0983	0.0613	0.1823
	0.8974	0.8974	0.9408	0.7684
	0.1068	0.1068	0.0981	0.1344
V	0.8638	0.0973	0.1653	0.1022
	0.1325	0.2375	0.7230	0.9013
	0.0993	0.7721	0.1827	0.0968
T_c	0.2255	0.0473	0.3177	0.1161
	0.7354	0.9537	0.6878	0.5620
	0.1237	0.1000	0.0965	0.4062
T	0.8879	0.0960	0.1094	0.4422
	0.1085	0.1788	0.5248	0.5600
	0.0995	0.8307	0.4590	0.0975

3.4 고장진단 시뮬레이션

표 2는 신경망에 적용된 고장패턴을 나타낸다. 이 표는 이미 빈번하게 발생한 고장에 대해서 사전에 만든 테이터로 신경망을 거쳐 그 패턴을 각 변수마다 3개의 수치로 표시하였다. 공정중의 6곳의 데이터를 이용하게 되므로 하나의 고장에 대해서 18가지의 데이터로 실제 고장여부를 판단하게 된다.

그림 5와 그림 6은 각각 기존의 Yunosuke의 방법[6]과 본 논문에서 제안한 방법으로 고장진단을 수행했을 때의 결과이다. 다른 요소들 즉, 패턴 추가의 용이성, 계산량 등과 같은 요소를 생각지 않고 단지 고장진단 결과만 보더라도, 그림 5에서처럼 윈도우를 상하로 이동시키면서 기준패턴을 단순하게 잡게되면 각각 다른 패턴에 대해서도 뚜렷한 차이를 나타내지 못함을 알 수 있다. 그림 6에서는 고장 1에 대해서 고장 1의 그래프에서만 기준치 이하로 떨어진 수치를 볼 수 있는 반면에, 그림 5에서는 기존의 방법이 다른 패턴에 대해서도 정확히 구분하지 못하고 고장 1과 고장 5, 6, 7, 8이 모두 그래프 상에 나타남을 볼 수 있다. 이는 수치가 적을수록 그 고장에 가까워진다는 것을 나타내는 그래프임을 감안할 때, 그림 5(기존의 방법)에 비하여 그림 6(제안한 방법)에서 성능의 개선이 뚜렷하게 그래프를 통해 나타나고 있다.

또한 제안한 방법은 윈도우를 고정시킴으로써 오랜 시간 진행된 고장이나 예측하지 못한 고장에 대한 대책을 마련하였고, 패턴의 저장에는 굳이 신경망을 이용하지 않고, 단순히 저장 비교하는 알고리즘을 이용함으로써 패턴을 많이 저장시킬 수 있고, 패턴의 추가도 쉬워졌다.

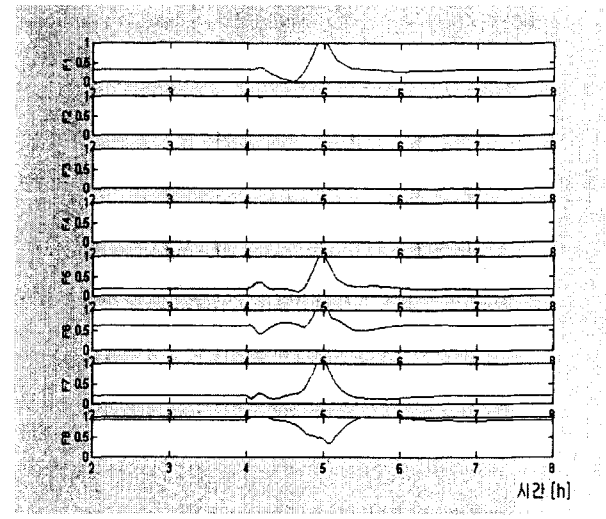

그림 5. 기존의 방법을 이용한 고장 1의 진단

Fig. 5. Diagnosis of fault #1 using conventional method

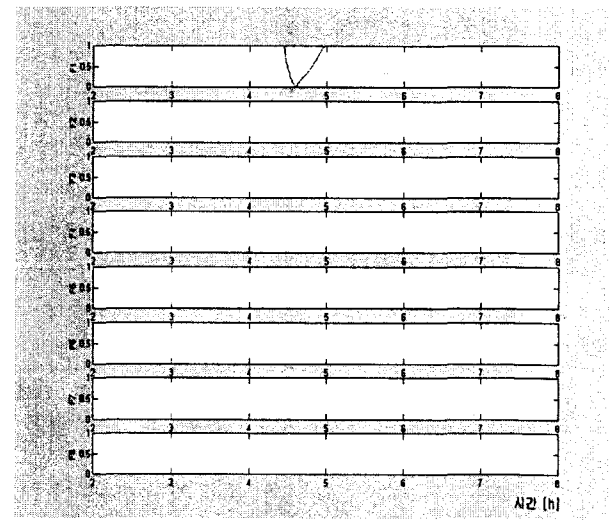

그림 6. 제안된 방법을 이용한 고장 1의 진단

Fig. 6. Diagnosis of fault #1 using proposed method

그림 6에서는 문턱값 δ_s를 1로 하고 고장 1이 발생시켰을 때, 각 고장 분류기의 $D_s(t)$의 수치를 나타낸 것이다. 그림 5에서 보면 고장 1이 4.46[h]에서 고장이 발생했음을 알 수 있다.

기존의 방법과 제안한 방법의 고장 패턴수에 따른 정량적 계산량을 비교, 검토한다. 패턴 분류를 위해 사용한 기존

의 방법에서는 신경망을 4층 구조로 각 기준패턴에 대해 수렴시켰다. 실제 기준패턴이 8개 일 때, 18-7-20-9의 노드로 수렴시켜 신경망을 이용하여 계산량을 측정하였고, 나머지 이보다 더 큰 수의 기준패턴을 가진 경우에는 단지 출력 노드의 수만을 증가시켜서 계산량을 측정하였다. 그러므로 실제로 기준패턴수가 8보다 큰 경우에는 노드 수나 층수가 이를 수렴시키기 위해 증가하게 될 것이므로, 기존의 방법을 이용할 경우, 표 3에서 나온 수치보다 훨씬 더 많은 계산량의 차이를 나타내게 될 것이다. 제안한 방법과 기존의 방법의 계산량 결과는 표 3에 나타내었다. 계산된 수치는 matlab의 cputime이란 함수를 이용하였으며, 사용된 컴퓨터기종은 PentiumII 266 MHz이다. 기준이 된 샘플링 횟수를 300회이며, 1회로 하기에는 수치가 너무 작아서 300회를 기준으로 계산된 시간을 측정하였다.

표 4. 제안한 방법과 기존의 방법의 계산량

Table 4 The amount of calculation between proposed method and conventional method

기준패턴수	4	8*	12	16
제안한 방법	0.38[sec]	0.82[sec]	1.18[sec]	1.59[sec]
기존의 방법[6]	3.99[sec]	4.83[sec]	5.50[sec]	6.32[sec]

3.5 외부 잡음 신호가 있는 경우의 고장진단

어떤 시스템에서든지 약간의 외부 잡음이 있기 마련이다. 그래서 본 논문에서는 각 실제 데이터 최대 변화량의 15% 내에서 균일 분포된 잡음(uniform noise)을 각 변수마다 발생시킨 다음, 고장진단이 가능한지를 시험해 보았다. 그림 7은 잡음 발생시의 고장 1의 고장패턴이며, 그림 8은 그 때 고장을 진단한 결과이다. 이런 실험은 다른 고장 패턴에 대하여도 행해 졌으며, 15% 정도의 잡음 신호에서 대부분의 고장패턴에 잡음이 없을 때보다 약간 늦거나 비슷한 시간에 별다른 이상 없이 고장이 진단됨을 알 수 있었다. 하지만 실제 잡음 신호가 높아지면 높아질수록 정확한 진단은 힘들어 질 것이다.

3.6 예측하지 못한 고장의 검출

그림 9에서는 실제 앞에서 언급한 적이 없는 즉, 예측하지 못한 고장패턴에 대해서 실험한 결과이다. 여기서 말하는 예측하지 못한 고장이란 이제까지 빈번하게 발생한 적이 없는 새로운 고장을 의미한다.

예측하지 못한 고장을 검출하는 기준은 앞에서도 말한바와 같이, 공정 각 부분에서 받은 데이터를 정규화 할 때에 정상상태를 기준으로 정규화 하였기 때문에 이들의 편차를 구하여, 이들이 기준치를 많이 벗어나게 되면, 이 또한 고장이라 할 수 있다. 공정의 각 부분 데이터가 정상상태를 벗어나면 앞에서 식 (5)에서의 $\sigma_{avg}(t)$의 값이 증가함은 굳이 그래프로 보여주지 않더라도 직관적으로 알 수 있다.

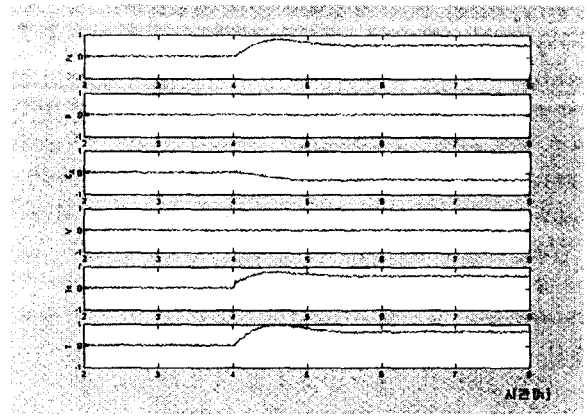

그림 7. 잡음이 있는 경우의 고장 1의 고장패턴

Fig. 7. Fault pattern of fault #1 in the case of noise

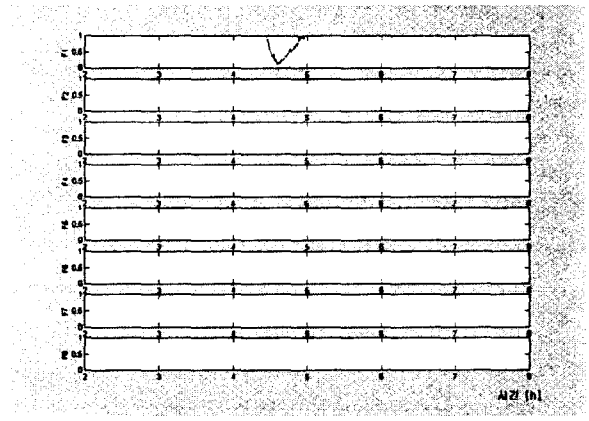

그림 8. 잡음이 있는 경우의 고장 1의 진단

Fig. 8. Diagnosis of fault #1 in case of noise

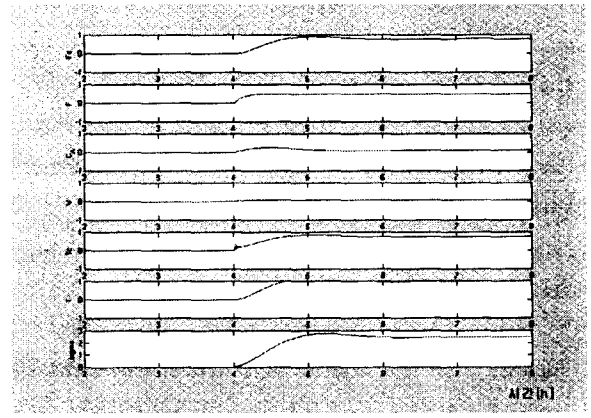

그림 9. 예측하지 못한 고장에서 $\sigma_{avg}(t)$ 의 값

Fig. 9. Value of $\sigma_{avg}(t)$ in unknown fault

4. 결론

본 논문에서 제안한 고장진단 알고리듬은 실시간으로 제공되는 데이터를 정규화하여 이를 신경망을 통해 분석하였으며, 분석된 데이터를 이미 저장되어 있는 고장패턴과 비교함으로써 고장을 검출하고 분류했다.

제시한 신경망의 기준패턴은 초기의 정상상태에서 고장시의 과도상태로 패턴이 변화한다는 사실에 착안하여 보다 현실적인 기준패턴을 제시하였다.

패턴을 저장, 분류하는 부분에 있어서는, 기존의 신경망[6]의 경우 새로운 고장패턴을 추가할 때마다 많은 수의 가중치를 변화시키면서 많은 횟수의 학습을 반복해야 하고, 그 과정에서 많은 시간적인 손실과 계산량의 증가를 초래할 뿐만 아니라, 수렴성도 보장할 수 없었다. 또한 그 많은 가중치들을 저장할 공간도 많이 필요하였다. 또한 패턴의 분류에 있어서도 많은 가중치와 활성화 함수를 거쳐서 패턴 분류가 이루어지므로 그 계산량은 패턴저장이나 분류에 있어서 엄청나다.

하지만 제안한 방법으로 고장패턴을 분류할 경우, 많은 학습 대신에 단순히 고장패턴을 한번 저장하고, 그 저장된 각각의 패턴에 대해서 단순히 실시간으로 들어온 데이터에 대해서 유사성만을 비교하여 고장진단을 수행하므로 계산량이나 저장공간 면에서 장점이 있다.

본 논문에서 제안한 데이터 정규화를 통하여 예측하지 못한 고장에 대한 검출이 가능함을 시뮬레이션을 통하여 검증하였다. 외부 잡음신호가 있는 경우에 대해서도 고장진단을 수행할 수 있음을 시뮬레이션을 통해서 검증하였다.

참 고 문 헌

[1] 이기상, "고장검출진단 및 고장허용제어의 개념과 동향", *전기학회지*, VOL. 48, No. 4, 1999.

[2] Moharned El Hachemi Benbouzid, "Induction Motors' Faults Detection and Localization Using Stator Current Advanced Signal Processing Technique" *IEEE trans. on Power Electronics*, VOL. 14, No. 1, pp. 14-22, 1999.

[3] Richard Dorr and Frederic Kratz, "Detection, Isolation, and Identification of Sensor Faults in Nuclear Power Plants", *IEEE trans. on Control System Technology*, VOL. 5, No. 1, pp. 42-60, 1997.

[4] Jacek M. Jurada, *Introduction to Artificial Neural Systems*, PWS, pp. 163-206, 1992.

[5] Kumpati S. Narendra and Kannan Parthasarathy "Identification and Control of Dynamical systems Using Neural Networks", *IEEE trans. on Neural networks*, VOL. 1, No. 1, pp. 4-27, 1990.

[6] Yunesuke Maki and Kenneth A Laparo, "A Neural-Network Approach to Fault Detection and Diagnosis in Industrial Process", *IEEE Trans. on Control Systems*, VOL. 5, No 6, pp 529-541, 1997.

[7] Luyben, *Process, Modeling, Simulation, and Control for Chemical Engineers 2nd Edition*, McGraw-Hill, 1989.

[8] Ray, *Advanced Process Control*, Mc-Graw-Hill, 1981.

저 자 소 개

이 진 하 (李 鎭 河)

1975년 11월 7일 생. 1998년 경북대 공대 전기공학과 졸업. 2000년 동 대학원 전기공학과 졸업(석사). 현재 동 대학원 박사과정.
Tel : 053-940-8804
E-mail : imustrun@palgong.knu.ac.kr

박 성 욱 (朴 省 昱)

1964년 11월 16일생. 1987년 경북대 공대 전기공학과 졸업. 1989년 동 대학원 전기공학과 졸업(석사). 1996년 동 대학원 전기공학과 졸업(공박). 1991.2.~1992.1. 국방과학 연구소 2분부 근무. 1998.2.~1998.12. 영국 버밍엄 대학 Post-Doc. 연수. 1992년~현재 구미1대학 전기과 조교수. 1998년~IEEE Control/Neural Network member
Tel : 054-440-1203, Fax : 054-440-1209
E-mail : swpak@blue.kumi.ac.kr

서 보 혁 (徐 輔 焃)

1952년 3월 11일생. 1957년 서울대 공대 전기공학과 졸업. 1980년 동 대학원 전기공학과 졸업(석사). 1985년 동 대학원 전기공학과 졸업(공박). 현재 경북대 전자전기공학부 교수
Tel : 053-950-5604, Fax : 053-950-6600
E-mail : bhsuh@bh.knu.ac.kr