제 1 차 년 도
최 종 보 고 서

GOVP1199511008

AI 기술개발(응용 S/W 기술 V)

Development of AI Technology(Application S/W Technology V)

신경망 예측 및 판단 시스템

Development of a Neuro-forecasting and Diagnosis System

연 구 기 관

한국과학기술연구원

시 스 템 공 학 연 구 소

과 학 기 술 처

배 포 선

사본 번호	부수	배 포 처
1/106	1	시스템공학연구소 영구 보존용
2/106	1	시스템공학연구소 도서과 보관용
3/106 – 6/106	4	시스템공학연구소 연구관리과 보관용
7/106 – 8/106	2	시스템공학연구소 인공지능연구부 보관용
9/106 – 11/106	3	과학기술처
12/106 – 106/106	94	필수 배포처

제 출 문

과학기술처 장관 귀하

본 보고서를 "신경망 예측 및 판단 시스템 (1/3)" 과제의 1차년도 최종보고서로 제출합니다.

1995년 9월 3일

주관연구기관명 : 한국과학기술연구원
시스템공학연구소

총괄연구책임자 : 한 문 성

연 구 원 : 정 연 구
박 전 규
서 상 원
이 종 현
황 두 성

참여기업 : (주)포스콘, (주)쌍용컴퓨터

참여기업책임자 : 김 태 수(포스콘), 안 병 덕(쌍용컴퓨터)

연 구 원 : 양 동 육 (포스콘)
이 유
남 한 식
심 상 식 (쌍용컴퓨터)
정 홍 기

위탁연구기관 : 한국과학기술원, 포항공과대학교

위탁연구책임자 : 이 수 영(KAIST), 방 승 양(포항공과대학교)

여 백

요 약 문

I. 제 목

신경망 예측 및 판단 시스템

II. 연구개발의 목적 및 중요성

인간의 정보처리 방식을 모방한 신경망은 적응 학습 기능, 대규모 병렬 처리, 그리고 함수 근사화 및 일반화의 장점을 가지고 있어 기존의 방법으로 해결하기 어려웠던 적응형 예측 및 제어, 음성 인식, 패턴 인식, 로보트 제어, 공정 제어 및 최적화등 다양한 분야에서 매우 활발히 응용되고 있다. 지금까지 산업 및 기업의 예측 및 진단 모델들은 전통적 통계 기법 즉 단순 회귀 또는 다중 회귀 모델, 연립 방정식, 단일 다변량 모델 및 상태 공간 모델 등에 근거하기 때문에 모델의 비적응성, 제약적인 가정의 요구, 처리의 복잡성 및 예측의 비정확성등의 단점이 존재한다. 시계열 예측문제는 관측치로부터 불확실성 및 복잡도가 높아 주어진 문제 해결을 위한 정형화된 예측방법을 얻는것은 어려운 것으로 알려져 있다. 그러나 과거의 축적된 지식과 경험을 바탕으로 다가올 미래 상황을 예측하는 문제는 실세계에 산재해 있어 연구되어야할 과제이다. 주어진 문제의 관측치로부터 문제 현상을 나타내는 결정방정식(deterministic equation)을 알수 있다면 현재의 관측치로부터 다음 미래상황을 예측할 수 있다. 신경망 이론의 예측 문제에서 활용은 미래 결정 방정식을 학습을 통하여 얻으므로서 전통적인 예측 방법보다 우수한 예측기술로 보고되고 있으나, 국내의 경우 체계적인 예측 기술이 확보되고 있지 않아 본 연구를 통하여 예측 및 판단 문제에서 신경망 기술 접근방법 및 응용분야를 확보하고자 한다. 본 연구에서는 신경망 예측 모델 시뮬레이터 개발, 응용영역에서 접근방법에 관한 연구 및 우수한 예측능력을 위한 학습 알고리즘 개발을 연구목표로 하고 있다. 따라서 신경망 기술을 신 인공지능 기법인 퍼지, 혼돈 이론, 유전알고리즘 및 통계적 기법과 융합하는 새로운 예측 이론에 기초한 지능형 예측 모델

을 개발하여 기존의 방법으로 우수한 성능을 얻기 어려웠던 증시 예측, 전력 수요 예측, 기후/기상 예측, 재정 및 경제 분야 예측, 전자 또는 기계 시스템의 센서데이타 예측 및 의사결정등과 같은 여러 응용분야를 도출, 분석 및 모델링을 하고자 하였다.

III. 연구의 내용 및 범위

전통적인 시계열 예측 문제에 적용되는 통계적 기반의 예측 모델를 분석하여 문제점을 살펴 보았다. 분석된 예측 모델로는 Box-Jenkins 이론에 근거한 AR(p), MA(q), ARMA(p,q) 및 신호처리 분야에서 filter 이론에 시간의 정보를 고려한 모델등이 있다. MLP 신경망 모델을 예측 문제에 이용될수 있는 신경망 예측 시뮬레이터를 구현하였으며 다양한 학습 방법을 제공한다. 이 시뮬레이터을 사용하가 위하여 신경망 모델 script 언어를 정의하였다. MLP 신경망 모델의 층과 층의 뉴런들이 fully connected 되는 제한을 두지 않고 사용자가 정의한 topology로 학습이 가능하므로 이 신경망 예측 시뮬레이터를 통하여 FIR 모델 및 TDNN 모델 등이 실험 가능하다. 학습되지 않은 자료에도 robust한 특성을 갖는 신경망 학습 알고리즘을 개발한다. 이를 위해 신경망의 출력단 오차이외에 중간층에도 일반화(Generalization) 성능 및 강건성(Robustness)을 향상시키기 위해서 새로운 오차 함수를 도입한다. 개발된 새로운 신경망 모델을 예측 시스템 개발에 적용하고, 예측 성능의 검증을 위해서 많이 이용되는 SantaFe Competition Data Set을 이용하여 제안한 방법과 기존의 다른 방법들과 예측 성능을 비교한다. 또한, 신경망의 각 뉴런에 이용되는 비선형 함수를 확장하여 보다 우수한 함수 근사화 성능을 가질 수 있는 새로운 신경망 모델을 개발한다. 신경망의 중간층 뉴런의 전달 함수로 많이 이용되는 시그모이드 함수와 이 함수를 M차 미분한 함수들을 함수 근사화를 위한 basis 함수로 이용하는 방법을 연구한다. 신경망의 최적 연결 강도(Interconnection weight) 값 및 basis 함수의 최적 계수를 함께 적응적으로 찾는 학습 알고리즘을 제안한다. 한편, 기존의 신경망 모델은 비구조적인 정보를 수치적으로 표현할 수 있지만, 실제 여러 가지 경우에 구조적인

정보가 존재하고 주어진 문제를 해결하기 위해서는 이를 최대한 이용하는 것이 바람직하다. 따라서, 구조적인 정보를 처리할 수 있는 퍼지 논리를 신경망에 도입하는 새로운 뉴로-퍼지 모델을 개발한다. 개발된 뉴로-퍼지 모델은 임의의 시스템의 퍼지 규칙을 찾을 수 있을 뿐만 아니라, 이미 존재하는 퍼지 규칙을 최적의 모델링 결과를 가지도록 소속 함수(Membership function)의 모양이나 중심값들을 조정할 수 있는 기능을 가진다. 기존의 방법들보다 최적화 문제에서 탁월한 성능을 보이는 유전 알고리즘(Genetic Algorithm)을 이용하여 신경회로망을 통계적 기법으로 학습시키는 학습 법칙을 개발하여 유전 알고리즘을 신경망에 접합시킬 수 있는 가능성을 검토한다.

IV. 연구개발결과 및 활용에 관한 건의

개발된 시뮬레이터는 응용영역에 적용할수 있도록 확장될 것이며 실험된 새 학습 알고리즘이 추가되어 좋은 성능을 낼수 있도록 해야 할것이다. 기존의 신경망 알고리즘에 약간의 변형만으로 보다 우수한 일반화 성능 및 입력 데이타의 변화에 대해 강건한 특성을 갖는 새로운 신경회로망 모델을 개발하므로써 시계열 예측과 같은 문제들을 효과적으로 해결하는 방법을 제시하여 신경망 기술의 실제 문제에의 응용 가능성을 앞당길 것으로 기대된다. 또한, 신경망 모델의 중간층 뉴런의 전달 특성으로 시그모이드(Sigmoid)와 가우시안(Gaussian)과 같은 여러가지 비선형 함수들을 함수 근사화를 위한 basis 함수로 이용하는 새로운 개념의 신경망을 개발하여 보다 우수한 함수 근사화를 구성할 수 있는 가능성을 열었으며, 신경망 모델에 퍼지 논리를 도입하는 새로운 형태의 뉴로-퍼지 모델을 개발하여 비구조적인 정보뿐만 아니라 구조적인 정보까지도 효과적으로 이용하는 새로운 방법을 제시하였다. 그리고, 신경망의 학습을 위해 유전 알고리즘을 이용하는 방법의 연구를 통해 신경망 학습시의 지역 최소화문제를 해결할 가능성을 검토하였다. 신경망, 퍼지 논리, 유전 알고리즘등 인공 지능의 차세대 첨단 기술의 핵심을 형성하는 중요한 기술들을 단일한 접근 방법으로 문제를 해결하는데 이용하는 것이 아니라, 이들의 장단점을 적절히 융합한 통합형 이론(Unified theory) 개발의 기초를 마

련하였으며, 향후 개발 한 알고리즘을 고성능 패턴 인식, 최적화, 로보트나 전자기기의 최적 제어등으로 그 적용 범위를 넓힐 수 있을 것으로 기대된다.

Summary

I. Title

Development of Neuro-Forecasting and -Diagnosis System

II. Objectives

Based on the advantages of parallel processing, adaptive learning and function approximation and generalization capabilities, artificial neural networks that imitate the function of our human brains have been widely applied in various fields such as prediction and adaptive control, speech recognition, pattern recognition, robot control, factory control and optimization, etc. On the other hand, prediction or diagnosis models used for industry or company have been based on only statistical methods such as simple or multiple regression, state-space modelling equations, etc. However, these approaches have defective properties of poor adaptability, requirment of restrictive assumptions, complexity and prediction inaccuracy. It is difficult to find out the formalized method for solving time-series prediction problems due to the uncertainity and complexity of those prediction problems. But we need to solve prediction problem because there are lots of prediction problems in our real-world. If we are given deterministic equations underlying the phenomenon of our problem given a current set of observations form our problem, we can forecast future observations. Recently we know that neural network approaches outperform traditional prediction methods in prediction problems. But in our country the systematic method of solving prediction problems are not satisfactorily devel-

oped.. Through this project we hope to develop the neural network approach in prediction theory and its applications. So, our goal is to implement neural network prediction simulator, apply that simulator to real-world problems, and make new learning algorithms which can outperform the conventional learning methods. Based on statistical method and the unified theory of new artificial intelligent techniques such as fuzzy theory, chaos theory, and genetic algorithm, we extend neural network approach to make up a new intelligent prediction tool. Among the promising possible applications are decision making and prediction in the fields of financial affairs or stock market, electric power consumption and sensors of electrical and mechanical systems.

III. Scope and Contents

First we considered drawbacks which happens to traditional prediction model maily based on statistical approaches. There are AR(p), MA(q), and ARMA(p,q) of the Box-Jenkins theory and one model with time embeding delay in the filter theory. We implement neural network prediction simulator for kinds of MLP neural nework models. The simulator has various learning methods. We use neural newtork script language which helps define the network and the learning parameters. The MLP model doesn't have the restriction that they are fully connected between neurons in two successive layers and is able to simulate kind of models such as TDNN and FIR models. We developed a new neural network learning algorithms which are robust even at uncertain data set and also contain the low pass filter characteristics. To do this, we induced a new error function developed by considering not only output errors but also

hidden neuron errors to improve generalization capacity and robustness. Then we adopted it for developing new prediction model and compared its performance with that of other models using SantaFe competition data set, which is popularly used for verifying the prediction performance. And, a new neural network model which is superior at function approximation problems is developed by expanding nonlinear funtions which have been commonly used as the transfer function of neurons. In other words we adopted sigmoid function and its derivatives as basis functions to improve the function approximation ability of neural network. Learning algorithm for finding the optimum interconnection weights and coefficients of the basis function is suggested. On the one side, the conventional neural network can only express the unstructured data numerically. However, in the real world, there exist some structured data, and it is undoubtedly efficient to make full use of them. So, a new neuro-fuzzy model incorporating the fuzzy logic into the neural network is developed, which can handle not only the structured information but also the unstructured data. This neuro-fuzzy model can modify and creat the fuzzy rules of a system. Finally, the genetic algorithm which is better at optimizing capacity than any other existing methods is used for training of neural network to investigate the possibility of resolving the local minima problem of the learning in neural network.

IV. Results and Recommendations

We developed the neural network prediction simulator and experimented new training algorithms. We hope to extend the simulator to apply to pratical areas and to add developed algorithms in fu-

ture. The developed training algorithms give better generalizing capability and the robustness of output data to the changes of the input data. These results are encouraging in the real-world applications of neural network. Also, a new neural network model which is superior at function approximation problems is developed by expanding nonlinear funtions that consist of sigmoid functions and its derivative functions of the m-th order. This approach may give a possibility for constructing a excellent function approximator. Moreover, we develop a new neuro-fuzzy model so that the neural network can handle a structured knowledge. And, the study on training of neural network by genetic algorithm is finally examined for solving the local minima problem of conventional error back propagation learning algorithm. In this project, a basic research for unified theory of the emerging technology such as neural network, fuzzy logic, genetic algorithm, and chaotic theory was performed for constructing a good function approximator. The results of this research can apply to time series prediction problems, complex pattern recognitions, optimizations and robot control problems.

Contents

여 백

목 차

여 백

1 장

서 론

실세계의 정보환경과 과학기술이 급격히 발전하고 있으며 모든 학문분야에서는 불확실성을 포함한 복잡한 현실적인 문제들을 가능한 빠르고 근접한 해를 얻고자하는 노력들이 이루어지고 있다. 특히 시계열 예측문제에서는 관측치로부터 불확실성 및 복잡도가 높아 주어진 문제 해결을 위한 정형화된 예측방법을 얻는것은 어려운 것으로 알려져 있다. 그러나 과거의 축적된 지식과 경험을 바탕으로 다가올 미래 상황을 예측하는 문제는 실세계에 산재해 있어 연구되어야할 과제이다. 주어진 문제의 관측치로부터 문제 현상을 나타내는 결정방정식(deterministic equation)을 알수 있다면 현재의 관측치로부터 다음 미래상황을 예측할 수 있다. 그러나 결정방정식을 알수 없다면 시스템의 행위변화와 현재 상태정보로부터 미래변화를 알아내는 접근방법이 필요하다. 본 연구는 후자의 문제에 초점을 두어 주어진 관측치로부터 규칙성 및 특성을 파악하여 미래 상황에 대한 행위변화에 대한 예측 및 판단을 하는 문제에 대한 연구이다.

인간의 정보 처리 방식을 모방한 인공 신경망(neural network)은 적응적 학습 기능, 대규모 병렬 처리, 그리고 임의의 함수를 근사화하거나 일반화할 수 있는 장점을 가지고 있다. 또한, 기존의 공학적 접근 방법들이 대상 시스템에 대한 엄밀한 수학적 해석에 기반하여 문제 해결을 위한 프로세스를 설계한 반면, 신경망은 대상 시스템(target system)에 대한 입출력 데이타만으로 적절한 프로세스를 설계할 수 있어, 적응형 예측 및 제어, 음성 인식, 복잡한 패턴 인식, 로보트 제어, 공정 제어 및 최적화등 다양한 분야에서 매우 활발히 응용되고 있다. 또한, 인간의 사고

과정 및 추론 방식을 가능성 이론에 기초한 새로운 수학적 방법으로 표현하는 퍼지 논리(fuzzy logic)는 기존의 논리 체계가 이분법에 의해서는 표현할 수 없었던 전문가의 지식이나 인간의 사고 과정에 따른 정보 처리를 표현하고 설계하는데 획기적인 방법을 제공하고 있다. 그러므로 불확실한 데이타를 처리하는데 효과적이어서 가전 제품 및 공정 제어와 같은 분야에서 우수한 성능을 보이고 있으며 최근에 활발히 이용되고 있는 분야이다. 다아윈의 진화론에 기초한 인간의 유전 현상을 공학적으로 표현하고 이용하는 유전 알고리즘(genetic algorithm)은 최적화 문제에서 탁월한 성능을 보이고 있어 기존의 방법으로 그해를 알수 없었던 복잡도가 높은 NP complete 문제들의 최적 해를 구하는 새로운 가능성의 장을 열고 있으며 최근에 개발된 혼돈 이론(chaos theory)은 혼돈 현상을 갖는 시스템의 분석 및 예측 등에 활발히 이용되어 좋은 실험적 성과를 내고 있다.

지금까지 개발된 산업 및 기업의 시계열 예측 및 진단 모델들은 전통적 통계 기법, 즉 단순 회귀 또는 다중 회귀 모델, 연립 방정식, 단일 다변량 모델 및 상태 공간 모델등에 근거하기 때문에 모델의 비적응성, 제약적인 가정의 요구, 처리의 복잡성 및 예측의 비정확성등의 단점들이 존재한다. 시계열이란 물리적 현상을 나타내는 시간 변화에 따라 정열되는 데이타를 일컫는다. 통계적 시계열 분석에 근거한 예측 및 판단은 관측치 변화에 대한 적응이 어려우며, 자료가 갖는 분포에 대한 가정이 전제 되어야 하며, 계량적인 변수와 정성적인 변수가 혼합되어 있는 경우 성과가 좋지 않다. 시계열 자료를 이용하여 미래 예측을 행하는 전통적인 방법으로 가장 많이 적용되고 있는 Box-Jenkins 방법이 있다[6]. 대표적인 방법으로 autoregressive model, moving average 모델, 그리고 autoregressive-moving average 모델 등이 있다. 시계열 분석을 이용한 예측시스템이 실세계에서 이용하기 위해서는 다음 조건이 만족 되어야 한다[25].

예측(forecasting 또는 predicting) 예측시스템은 short-term 내의 예측치가 정확해야 한다.

모델링(modeling) 예측시스템의 long-term 행위(trend)를 설명할 수 있어야 한다.

특성화(characterization) 환경 변화로 인한 훼손된 입력(noised input)에 대하여 적응이 가능한 특성을 갖고 있어야한다.

최근에, 이들 예측 모델의 개발을 위해 새로운 정보처리 메카니즘인 신경망, 퍼지 논리, 유전 알고리즘, 그리고 혼돈 이론등을 적용하고자 하는 시도가 많이 발표되고는 있으나 아직까지 예측 성능이나 체계적인 설계 방법이 부족한 실정이다. 또한, 최근의 세계적인 기술 개발 추세는 기존의 단일한 문제 해결 접근 방식에서부터 점차로 여러가지 기술들을 통합하여 문제 해결을 추구하고 있으며, 신경망과 퍼지 논리, 유전 알고리즘이나 혼돈 이론과 같은 새로운 기술들의 장점들을 최대한 이용하며, 각각의 단점들을 서로 보완할 수 있는 통합형 이론의 개발이 절실히 필요해지고 있다.

본 연구에서는 시계열 예측을 위한 신경망 예측시스템에 대한 연구가 진행되었다. 인공지능 신경망은 전통적인 폰 노이만 계산방법과는 다른 패러다임을 갖으며 전통적인 계산기법으로 잘 해결되지 않는 문제에 적용되어 좋은 성과를 보여주고 있다. 예측 및 판단 문제에서 신경망 기술을 이용한다는 것은 바로 추정해야 할 함수 및 파라미터들을 신경망으로 대치하는 것으로 볼수 있으며 문제에 따라 신경망 구조(topology) 및 입출력 설계가 다양하게 나타난다. 예측시스템에서 신경망 적용방법은 학습데이타 전처리, 학습 및 파라미터 조정, 그리고 예측 및 판단 단계를 거친다. 학습방법 및 학습데이타 차원에 따라 신경망 구조가 결정된다. 시계열 예측 문제에서 신경망 이용은 과거 시계열 변화 과정을 학습시키므로서 시계열에 내재하는 자료 변동 패턴을 학습하여 수학적인 관계식을 찾아 기억하게 된다. 시계열 자료의 특성을 보여주는 임의변동(randomness), 추세(trend), 계절성 및 주기성 등이 학습과정에서 신경망에 내재된다. 예측 문제에 이용한 신경망 모델로는 지도학습을 하는 backpropagation, recurrent neural network(RNN), finite impluse response(FIR) neural network, radial basis function 모델등이 있다.

우수한 예측성능을 갖는 학습 알고리즘에 관한 연구로 기존의 성능 지수 함수에 중간층 뉴 런에 부가적인 constraint 함수를 사용함으로써 보다 향상된 일반화 성능 및 robustness를 갖는 신경망 모델 하습 알고리즘을 연구개발되었다. 또한 기

존의 신경망 모델은 비구조적인 정보를 통합하기 위한 연구로 신경망 중간층 뉴런의 비건형 함수 확장 방법 및 퍼지 논리를 융합하는 퍼지-신경망 모델을 개발하여 비선형 문제에 적용하여 그 성능을 확인 하였다. 그리고 유전 알고리즘을 이용하여 주어진 문제를 해결하기 위한 신경망의 최적 연결 강도값을 찾는 방법이 연구되었다.

2 장에서 예측 및 판단을 위한 stochastic 방법을 살펴보고, 3 장에서는 예측을 위한 신경망 모델 및 학습규칙 등을 다룬다. 4 장은 신경망, 퍼지논리 및 유전자 통합형 예측모델을 기술하며, 5 장에서는 회전기기 고장 진단 및 수명예측에 대한 이론적인 연구가 기술되었으며, 6 장은 개발중인 신경망 예측시스템을 기술한다.

2 장

통계적인 시계열 예측

2.1 개요

시계열 예측의 목적은 결정 과정에 중요한 변수들의 값들을 정량적인 방법으로 예측하는 시스템의 개발이며 통계적인 시계열 예측은 그 변수의 과정을 확률적 법칙에 따르는 것으로 가정하여 알맞은 확률 과정 (stochastic process) 모델 설정과 통계적인 추정을 통하여 예측 시스템을 형성하는 것을 그 방법론으로 한다.

시계열이란 관심있는 변수에 대한 관측치의 열을 말한다. 변수는 시간 변역에서 보통 일정 간격으로 이산적으로 관측되며 시계열 분석은 그 시계열을 생성하는 현상 또는 과정을 설명하는 일을 맡는다. 시계열 예측을 위해서는 미래로 연장 가능한 수학적 모델에 의해 시계열 과정의 행동을 나타내는 것이 필요하며 그 모델은 현시점과 가까운 시계열의 local segment 에 대해 그 관측치를 잘 나타내어야 한다.

시계열의 몇가지 특징적인 패턴들로는 프로세스의 레벨이 시간에 따라 변하지 않는 경우, 선형 경향을 보이는 경우, 주기적으로 변하는 경우, 또한 이들의 조합형 등이 있으며 이 외에도 배경 프로세스의 변화가 생겨 야기되는 임펄스 패턴, 계단식 변화, 그리고 갑작스런 경향을 경험하는 ramp형 패턴 등이 있다.

시계열을 나타내기 위한 대부분의 모델들은 대수적인 또는 exponential 이나 sinusoidal 함수를 사용한다. 그 일반적인 형태는

$$x_t = b_1 z_1(t) + b_2 z_2(t) + \cdots + b_k z_k(t) + \epsilon_t$$

여기서 $\{b_i\}$ 는 모수들이며 $\{z_i(t)\}$ 은 t 의 함수이다. ϵ_t 은 임의 component. 예로 quadratic 경향 모델이나 12(달) 마다 되풀이 되는 주기를 표현하는 sinusoidal 모델을 들면,

$$x_t = b_1 + b_2 t + b_3 t^2 + \epsilon_t$$

$$x_t = b_1 + b_2 \sin \frac{2\pi t}{12} + b_3 \cos \frac{2\pi t}{12} + \epsilon_t$$

이에 반해 시계열 표현에 있어 겉보기에 상당히 다른 접근 방식을 사용할 수 있는데, 현재의 관측치 x_t 가 prior random component 들, $\epsilon_t, \epsilon_{t-1}, \epsilon_{t-2}, \ldots$ 의 함수로 모델링 된다.

$$x_t = \mu + \psi_0 \epsilon_t + \psi_1 \epsilon_{t-1} + \psi_2 \epsilon_{t-2} + \cdots$$

여기서 μ 와 $\{\psi_i\}$ 는 상수들이다. 이런 형의 모델은 선형 필터 모델(linear filter model) 이라 불리우며 특히 관측치 간에 강한 상관 관계가 있는 경우 우수한 모델이다. 아래 2.4 절에 자세히 기술한다.

시계열 예측의 절차는 알려지지 않은 모수들의 추정 후에 얻어진 모델을 미래 시점으로 투사하여 예측치를 얻는 것이다. 예로서 선형 경향 모델 (linear trend model) 의 알려지지 않은 모수 b_1 과 b_2 에 대한 추정치들을 $\hat{b_1}$ 과 $\hat{b_2}$ 라 하자. 현재 시점 T 에 있다면 미래 시점 $T+\tau$ 에 관측치의 기대값의 예측은

$$\hat{x}_{T+\tau}(T) = \hat{b_1} + \hat{b_2}(T+\tau)$$

이다.

완성된 예측 시스템의 효과의 평가를 위한 기준으로서 많이 쓰이는 것은 *mean squared error*,

$$E[e_t^2] = E[(x_t - \hat{x}_t)^2]$$

이며 예측이 bias 를 갖지 않을 경우 예측 에러 e_t 의 분산과 같다.

2.2 전통적인 방법들

2.2.1 회귀 분석에 의한 예측

반응 변수 x 와 시간의 함수들인 회귀 변수 $z_1, z_2, \cdots, z_k$ 들의 관계를 다변량 선형 회귀분석에의해 추정하여 시계열 예측에 사용하는 것이다. 행열식을 이용하여 간단하게 설명된다.

x 는 관측치의 $n \times 1$ 열 백터, **Z** 는 $n \times p$ 회귀 변수들의 matrix, **b** 는 $p \times 1$ 정해지지 않은 모수들의 열 벡터 그리고 **epsilon** 은 $n \times 1$ 오차들의 열 벡터라 하자.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \mathbf{Z} = \begin{pmatrix} 1 & z_{11} & z_{12} & \cdots & z_{1k} \\ 1 & z_{21} & z_{22} & \cdots & z_{2k} \\ & & \cdots & \cdots & \\ 1 & z_{n1} & z_{n2} & \cdots & z_{nk} \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_k \end{pmatrix} \quad \text{and} \quad \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

그러면 선형 회귀식 모형은

$$\mathbf{x} = \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon} \tag{2.1}$$

으로 된다. 방정식 2.1 은 일반적인 선형 모델 (general linear model) 을 표현하는 형태이다. 어떤 함수적 관계도 알려지지 않은 모수에 대하여 선형이기만 하면 이 형태로 표현된다. 모수 벡터 **b** 를 최소 자승 법으로 추정하는 데 있어 오차 벡터 **epsilon** 에 대한 가정은 보통 오차들의 기대값이 0 이며 상관관계가 없고 분산이

일정하다 등이다. 즉,

$$E(\epsilon) = 0 \quad Var(\epsilon) = \begin{pmatrix} \sigma_\epsilon^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_\epsilon^2 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_\epsilon^2 \end{pmatrix}$$

모수 벡터의 추정치 $\hat{\mathbf{b}}' = (\hat{b_0}, \hat{b_1}, \ldots, \hat{b_k})$ 에 대하여 fitting 된 모델은

$$\hat{\mathbf{x}} = \mathbf{Z}\hat{\mathbf{b}}$$

Fitted 모델의 값과 실제 데이타의 차는 residual 이라 불리우며 residual 벡터 $\mathbf{r}$ 은

$$\mathbf{r} = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - \mathbf{Z}\hat{\mathbf{b}}$$

최소 자승 추정에 의한 $\mathbf{b}$ 의 추정은 오차항의 자승합을 최소화한다.

$$\mathcal{L}(\mathbf{b}) = \epsilon'\epsilon = (\mathbf{x} - \mathbf{Z}\mathbf{b})'(\mathbf{x} - \mathbf{Z}\mathbf{b}) \tag{2.2}$$

방정식 2.2 의 우항을 전개하면

$$\mathcal{L}(\mathbf{b}) = \mathbf{x}'\mathbf{x} - 2\mathbf{b}'\mathbf{Z}'\mathbf{x} + \mathbf{b}'\mathbf{Z}'\mathbf{Z}\mathbf{b} \tag{2.3}$$

방정식 2.3 을 $\mathbf{b}$에 관해 미분하여 정규 방정식을 얻는다.

$$\mathbf{Z}'\mathbf{Z}\hat{\mathbf{b}} = \mathbf{Z}'\mathbf{x}$$

따라서 최소 자승 추정치는 $(\mathbf{Z}'\mathbf{Z})^{-1}$ 가 존재하는 경우

$$\hat{\mathbf{b}} = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{x}$$

Source of variation	Sum of squares	Degrees of freedom	Mean square	F_0
Regression	$SS_R = \hat{\mathbf{b}}'\mathbf{Z}'\mathbf{x} - n\bar{x}^2$	k	$MS_R = SS_R/k$	MS_R/MS_E
Residual	$SS_E = S_{xx} - SS_R$	$n-p$	$MS_E = SS_E/(n-p)$	
Total	S_{xx}	$n-1$		

표 2.1: 회귀분석의 유의미성 검정을 위한 분산 분석

가 되며 이 $\hat{\mathbf{b}}$ 의 기대값과 분산은 각각

$$E(\hat{\mathbf{b}}) = \mathbf{b} \quad Var(\hat{\mathbf{b}}) = (\mathbf{Z}'\mathbf{Z})^{-1}\sigma_\epsilon^2$$

이 됨을 쉽게 알수 있다. σ_ϵ^2 의 추정은 그 불편 추정치인

$$\hat{\sigma_\epsilon^2} = (\mathbf{x} - \hat{\mathbf{x}})'((\mathbf{x} - \hat{\mathbf{x}})/(n-p) = \mathbf{r}'\mathbf{r}/(n-p)$$

을 사용해서 할 수 있다.

ϵ의 확률 분포로서 보통 정규분포를 가정하여 모델 모수들에 대한 통계적 검정을 한다.

$$\epsilon \sim N(0, \sigma_\epsilon^2\mathbf{I}) \text{ 따라서 } \mathbf{x} \sim N(\mathbf{Zb}, \sigma_\epsilon^2\mathbf{I})$$

회귀 모델의 전반적 유의미성에 대한 검정은, 즉

$$H_0 : b_1 = b_2 = \cdots = b_k = 0$$

$$H_1 : \text{ 적어도 하나의 } b_j \neq 0$$

Likelihood Ratio 검정에 의해, 분산 분석 (표 2.1)에 나타나는 통계량인 회귀에 의한 평균 자승치, MS_R 와 Residual 평균 자승치, MS_E 의 비로써 F-검정을 한다 : $F_0 = MS_R/MS_E > F_{k,n-p}(\alpha)$ 이면 귀무가설 $H_0 : b_1 = b_2 = \cdots = b_k = 0$ 를 기각하여 적어도 하나의 회귀 모수가 유의미함을 받아들인다. 모델 모수들 각각의 유의미성은 t 검정을 사용하여 할 수 있다. $(\mathbf{Z}'\mathbf{Z})^{-1}$ 의 j-th diagonal element

를 C_{jj} 라 하면

$$H_0 : b_j = 0$$

$$H_1 : b_j \neq 0$$

의 검정은 통계량

$$t_0 = \frac{\hat{b_j}}{\sqrt{\hat{\sigma_\epsilon^2} C_{jj}}}$$

이 t_{n-p} 분포를 따르므로 $|t_0| > t_{n-p}(\frac{\alpha}{2})$ 이면 $H_0 : b_j = 0$ 를 기각한다. 그런데 이 각각의 모수 검정 절차는 다른 모든 회귀 변수들이 포함된 상태에서 검정되는 변수의 기여도를 측정하는 조건적 검정이기 때문에 회귀변수들 간의 상관이 상당히 존재할 경우 각 변수의 중요성에 대해 오류적인 판정을 할 수 있으므로 유의해서 사용하여야 한다. 또한 각각의 모수 b_j 에 대한 $100(1-\alpha)\, percent$ 신뢰구간은

$$\hat{b_j} - t_{n-p}(\frac{\alpha}{2})\sqrt{\hat{\sigma_\epsilon^2} C_{jj}} \leq b_j \leq \hat{b_j} + t_{n-p}(\frac{\alpha}{2})\sqrt{\hat{\sigma_\epsilon^2} C_{jj}}$$

이 된다.

회귀 분석 모델이 정해지면 그것을 예측 모델로써 사용할 수 있는데, 독립 변수가 시간의 함수 일때 미래 시점 $T+\tau$ 에 대한 예측치는 정해진 모델의 시간 $T+\tau$ 에서의 평균 반응에 대한 점추정치로서 얻어진다. 이 추정치는 시점 $T+\tau$ 에서의 fitting 된 회귀 모델의 값, $\hat{x}$ 이다. 오차에 대한 정규성 (normality) 가정으로부터 $\mathbf{z}' = (z_1, \cdots, z_k)$ 에서의 x 의 평균에 대한 $100(1-\alpha)\, percent$ 신뢰 구간은

$$\hat{x} \pm t_{n-p}(\frac{\alpha}{2})\{\hat{\sigma_\epsilon^2}\mathbf{z}'(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{z}\}^{1/2}$$

이며, 미래 관측치에 대한 $\mathbf{z}' = (z_1, \cdots, z_k)$ 에서의 예측 $\hat{x}$ 은 $\hat{x} = \mathbf{z}'\hat{\mathbf{b}}$ 이며 미래 관측치에 대한 $100(1-\alpha)\, percent$ 예측 구간은

$$\hat{x} \pm t_{n-p}(\frac{\alpha}{2})\{\hat{\sigma_\epsilon^2}[1 + \mathbf{z}'(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{z}]\}^{1/2}$$

이 된다.

관심 변수를 변환하여 모델링하는 경우도 많은데, (예를 들어 오차의 분산을 더

상수에 가깝게 하기 위한 square-root 변환) 이 경우 변환된 모델로부터의 예측치에 역 변환을 적용하여 예측치를 산출하면 일반적으로 편차를 갖는 예측치를 얻게 되며 이 경우에는 변환된 예측치를 (근사적으로) 편차 없는 원래 단위의 예측치로 변환하는 방법들을 사용할 수 있다[7]. 그러나 percentiles 는 변환에 의해 영향을 받지 않으므로 예측 구간은 역 변환을 적용하여 직접 사용할 수 있으며 이때 원래 scale 로 변환된 예측 구간은 일반적으로 예측치에 대해 대칭적이지 않다.

2.2.2 Discounted 최소 자승 추정에 의한 direct smoothing 방법

예측에 있어 회귀 분석을 사용하는 경우의 주된 불리한 점은 추정된 모델 모수들 $\{\hat{b}_i\}$ 이, 가장 최근의 관측치를 고려하기 위해서는, 매 시점 마다 update 되어야 한다는 점이다. 따라서 많은 수의 시계열을 매 시간 분석하여야 한다면 관련 계산량이 너무 커지는 것이다.

그러나 모델의 회귀 변수들이 다항식이나 지수함수 또는 삼각함수 형태의 시간의 함수일때는 모델 모수를 직접 update 할 수 있게 되며 이 방법은 효과에 있어서 전 단계의 모델 모수들을 현 시점의 예측 오차로써 smoothing 하여 새 모수들을 얻는 것이다. 이 절차를 direct smoothing 이라 하며 계산적으로 매우 효율적이다. 이 절차는 discounted least squares 기준에 의한 추정 을 사용함으로써 유도 되어 질 수 있다. 오차의 제곱항들이 시간상으로 현시점으로부터 오래될수록 discount 되는 것이다. 분산의 불균등에 따른 가중 최소 자승 (weighted least squares)의 형태로 볼 수 있으나 여기서는 단순히 오래된 관측치에 대해 가중치를 줄이기 위한 것이다.

이 direct smoothing 을 다항식 모델에 적용할 경우 다중 exponential smoothing의 형태로 나타나게 되며, exponential smoothing 은 그 계산 상의 효율성과 상당한 정확도로 말미암아 근접한 미래를 예측하기 위해 이산형 시계열을 smoothing 하는데 가장 널리 쓰이는 절차 중 하나이다.

편의상 모델의 시간 기준점을 항상 가장 최근 관측치의 시점, T로 하는 모델을

사용하면 시간 T 에 Discounted Least-Squares 기준은

$$\mathcal{L}(\mathbf{a}) = \sum_{j=0}^{T-1} W^2_{T-j,T-j}[x_{T-j} - \mathbf{z}'(-j)\mathbf{a}(T)]^2$$

으로 표현할 수 있다. $\mathbf{a}(T)$ 는 기준 시점에서의 모델 모수 벡터이며 $\mathbf{z}(-j)$ 는 시간 함수 $\mathbf{z}$ 의 값이 $-j$에 산출됨을 나타낸다. 가중치 $W^2_{T-j,T-j}$ 는 보통 다음과 같이 준다.

$$W^2_{T-j,T-j} = \beta^j \quad j = 0, 1, \ldots, T-1$$

Discounting factor β 는 $0 < \beta < 1$ 로써 정한다.

그리고 회귀변수들 $z_i(t)$가 다음의 식을 만족하는 경우를 가정한다.

$$z_i(t+1) = L_{i1}z_1(t) + L_{i2}z_2(t) + \cdots + L_{ik}z_k(t) \quad i = 1, 2, \ldots, k$$

즉, 시간 $t+1$ 에서 한 회귀변수의 값이 그 전 시간 t 에서의 회귀변수 값들의 선형결합으로 나타낼 수 있는 경우다. $\mathbf{L}$ 이 위의 L_{ij} 의 $k \times k$ 행열이라 하면,

$$\mathbf{z}(t+1) = \mathbf{L}\mathbf{z}(t)$$

이 천이 조건을 만족하는 유일한 시간 함수는 다항식이나 지수함수(exponential) 또는 삼각함수 뿐이다. T 에서 회귀변수들의 weighted sum of squares and cross products matrix를 $\mathbf{G}(T)$ 라 하면,

$$\begin{aligned} \mathbf{G}(T) &= \sum_{j=0}^{T-1} \beta^j \mathbf{z}(-j)\mathbf{z}'(-j) \\ &= \mathbf{G}(T-1) + \beta^{T-1}\mathbf{z}(-T+1)\mathbf{z}'(-T+1) \end{aligned}$$

$\{z_i(t)\}$ 가 매우 빨리 decay 하지 않는 경우 $\mathbf{G}(T)$ 는 극한 $\mathbf{G}$ 를 갖는다.

$$\mathbf{G} = \lim_{T\to\infty} \mathbf{G}(T) = \sum_{j=0}^{\infty} \beta^j \mathbf{z}(-j)\mathbf{z}'(-j)$$

따라서 $\mathbf{G}^{-1}$ 는 단지 한번 계산하면 된다. 이 $\mathbf{G}(T)$ 의 steady-state 값을 사용하

여 $\mathbf{a}(T)$ 의 추정을 하면,

$$\hat{\mathbf{a}}(T) = \mathbf{G}^{-1}\mathbf{g}(T)$$

이다. 이때 $\mathbf{g}(T) = \sum_{j=0}^{T-1} \beta^j x_{T-j}\mathbf{z}(-j)$ 이며 다음의 재귀식을 만족함을 보일 수 있다.

$$\mathbf{g}(T) = x_T\mathbf{z}(0) - \beta\mathbf{L}^{-1}\mathbf{g}(T-1)$$

그리고 $\mathbf{G}\hat{\mathbf{a}}(T-1) = \mathbf{g}(T-1)$ 이므로

$$\begin{aligned} \hat{\mathbf{a}}(T) &= \mathbf{G}^{-1}[x_T\mathbf{z}(0) + \beta\mathbf{L}^{-1}\mathbf{g}(T-1)] \\ &= x_T\mathbf{G}^{-1}\mathbf{z}(0) + \beta\mathbf{G}^{-1}\mathbf{L}^{-1}\mathbf{G}\hat{\mathbf{a}}(T-1) \\ &= \mathbf{h}x_T + \mathbf{H}\hat{\mathbf{a}}(T-1) \end{aligned} \tag{2.4}$$

$\mathbf{h} = \mathbf{G}^{-1}\mathbf{z}(0)$ 그리고 $\mathbf{H} = \beta\mathbf{G}^{-1}\mathbf{L}^{-1}\mathbf{G}$.

$\mathbf{H} = \mathbf{L}' - \mathbf{h}\mathbf{z}'(1)$ 이며 방정식 2.4 은

$$\begin{aligned} \hat{\mathbf{a}}(T) &= \mathbf{h}x_T + [\mathbf{L}' - \mathbf{h}\mathbf{z}'(1)]\hat{\mathbf{a}}(T-1) \\ &= \mathbf{L}'\hat{\mathbf{a}}(T-1) + \mathbf{h}[x_T - \mathbf{z}'(1)\hat{\mathbf{a}}(T-1)] \end{aligned} \tag{2.5}$$

가 된다. 여기서 $\mathbf{z}'(1)\hat{\mathbf{a}}(T-1)$ 은 시간 $T-1$ 에 만들어진 x 에 대한 1-step-ahead 예측치 $\hat{x_T}(T-1)$ 인 것이다. 따라서 앞에 설명한대로 시간 T 에 discounted least-sqaures 모수 추정은 전 단계에서 만들어진 추정치와 1-step-ahead 예측 오차와의 linear combination 으로 됨을 알 수 있다. 어느 미래 시점 $T+\tau$ 에 대한 예측 (현재 시점 T 에) 은

$$\hat{x_{T+\tau}}(T) = \mathbf{z}'(\tau)\hat{\mathbf{a}}(T) = \sum_{i=1}^{k} \hat{a}_i(T)z_i(\tau)$$

이 되며, 초기값 $\hat{\mathbf{a}}(0)$ 은 과거 data 를 사용하여 least-squares 방법 등으로 추정해야 한다. τ-step-ahead 예측 오차의 분산은

$$Var[x_{T+\tau} - \hat{x_{T+\tau}}(T)] = \sigma_\epsilon^2 + Var[\hat{x_{T+\tau}}(T)] = \sigma_\epsilon^2 + \mathbf{z}'(\tau)\mathbf{V}\mathbf{z}(\tau)$$

임을 보일 수 있다. 여기서 $\mathbf{V}$ 는 $\hat{\mathbf{a}}(T)$ 의 공분산 행열이며

$$\mathbf{V} = \mathbf{G}^{-1}(\sum_{j=0}^{\infty} \beta^{2j}\mathbf{z}(-j)\mathbf{z}'(-j))\mathbf{G}^{-1}\sigma_\epsilon^2$$

이다.

선형 경향 모델 $x_t = b_1 + b_2 t + \epsilon_t$ 의 경우

$$\mathbf{L} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad \mathbf{z}(t) = \begin{pmatrix} 1 \\ t \end{pmatrix}$$

이되며 모수 updating 방정식 2.5 로부터

$$\begin{aligned} \hat{a_1}(T) &= \hat{a_1}(T-1) + \hat{a_2}(T-1) + h_1 e_1(T) \\ \hat{a_2}(T) &= \hat{a_2}(T-1) + h_2 e_1(T) \end{aligned}$$

임을 볼 수 있다.

$$\begin{aligned} \mathbf{G}(T) &= \sum_{j=0}^{T-1} \beta^j \mathbf{z}(-j)\mathbf{z}'(-j) \\ &= \sum_{j=0}^{T-1} \beta^j \begin{pmatrix} 1 \\ -j \end{pmatrix} (1, -j) \\ &= \sum_{j=0}^{T-1} \beta^j \begin{pmatrix} 1 & -j \\ -j & j^2 \end{pmatrix} \end{aligned}$$

이므로 급수의 극한을 정리하면

$$\mathbf{G}(T) = (1-\beta^T) \begin{pmatrix} \frac{1}{1-\beta} & \frac{-\beta}{(1-\beta)^2} \\ \frac{-\beta}{(1-\beta)^2} & \frac{\beta(1+\beta)}{(1-\beta)^3} \end{pmatrix}$$

이 되며 또한 steady-state 값 $\mathbf{G}$ 가 되는 것이다. smoothing vector $\mathbf{h} =$

$\mathbf{G}^{-1}\mathbf{z}(0)$ 는 $\begin{pmatrix} 1-\beta^2 \\ (1-\beta)^2 \end{pmatrix}$ 이 된다. 따라서 모수 updating 공식은 최종적으로

$$\begin{aligned} \hat{a}_1(T) &= \hat{a}_1(T-1) + \hat{a}_2(T-1) + (1-\beta^2)e_1(T) \\ \hat{a}_2(T) &= \hat{a}_2(T-1) + (1-\beta)^2 e_1(T) \end{aligned}$$

으로 표현된다.

이 updating 공식에 의한 모수 추정치는 $(1-\beta)$ 를 smoothing 상수 로 하는 이중 exponential smoothing 방법을 사용한 경우와 같은 것이며 일반적으로 k 차 다항식 모델에 대한 discounting factor β 를 사용한 direct smoothing 은 $(1-\beta)$ 를 smoothing 상수로 한 $k+1$ 차 다중 exponential smoothing 과 동등한 것이다.

2.3 선형 Stationary 시계열 예측 모델

2.3.1 개요

위의 전통적인 방법들에서는 시계열 데이타 $\{x_t\}$에 대해 프로세스의 평균이 시간 t의 결정적 함수로서 가정된다. 그리고 어느 시점에서의 관측치는 그 평균 더하기 random 오차 component 로 구성된다. 여기서 오차들은 일반적으로 통계적으로 상호 독립이라 가정된다. 따라서 관측치들 $\{x_t\}$ 또한 독립적인 변수로 가정된다. 그러나 오차들의 독립 가정은 종종 적당하지 않다. 즉, 많은 경우에 시계열의 연속적인 관측치들이 서로 종속적이다.

이러한 경우들에 exponential smoothing 을 통한 예측 방법은 관측치들 간의 종속성을 효과적으로 이용하지 못하므로 부적절하게 된다. 현실적으로 종종 exponential smoothing 예측 방법들이 상호 종속적인 관측치들의 시계열에 대해서도 어느 정도 좋은 결과를 내며 적용되기도 하나, 이 종속성에 대한 모델링을 개발하여 더 우월한 결과들을 일반적으로 산출하는 예측 기술들이 발전 되었는데 많은 경우 Box 와 Jenkins [6] 에 의해서 확립된 시계열 분석에 바탕을 두고 있다.

시계열 예측 모델 적용을 위한 반복 단계

그림 2.1은 시계열 예측 모델 적용 반복단계를 보여주고 있다.

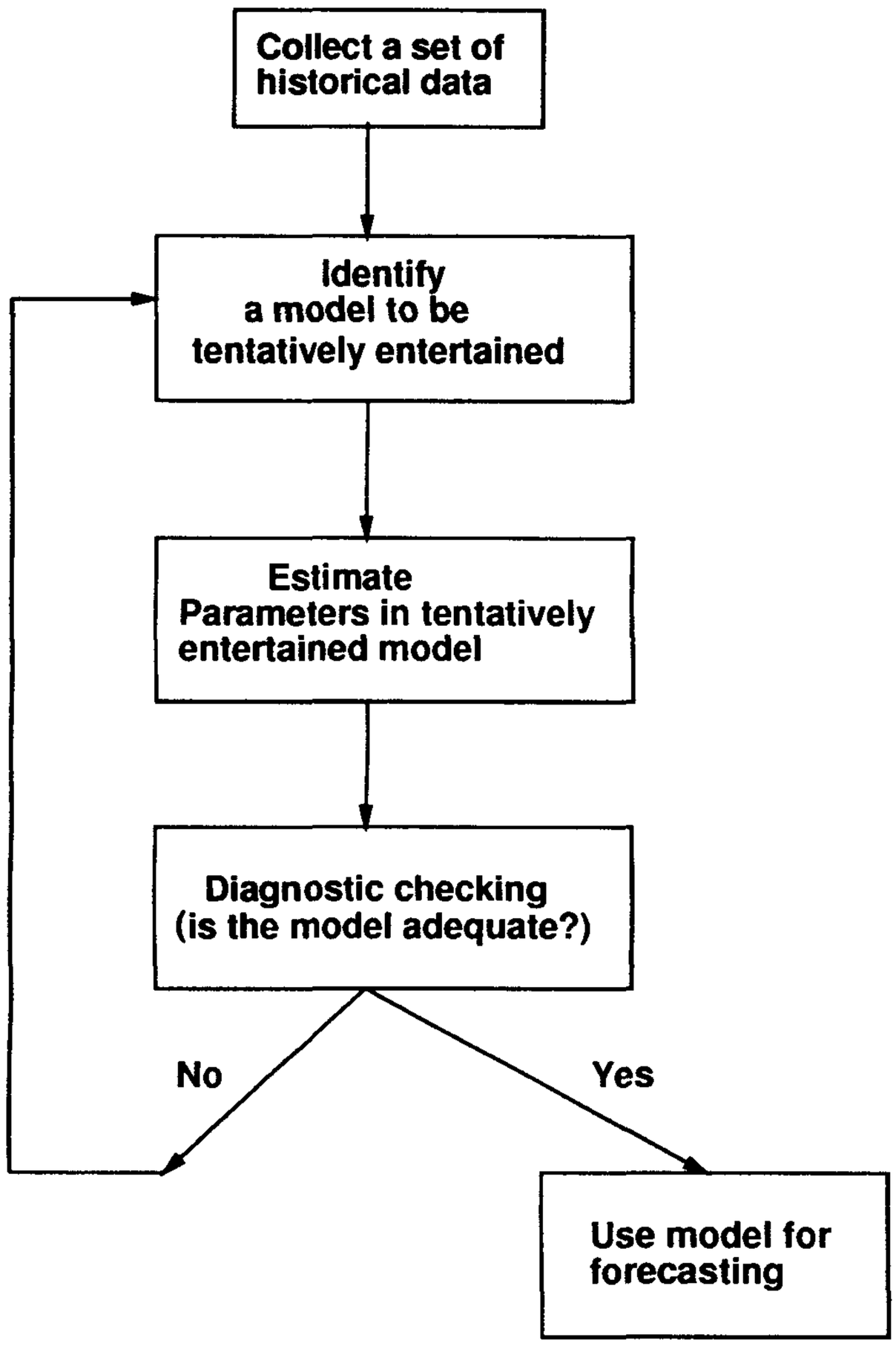


그림 2.1: 시계열 예측 모델 적용을 위한 반복 단계

시계열 모델 classes

시간 t에서 시계열 예측 데이타가 independent random 변수들 $\epsilon_t, \epsilon_{t-1}, \epsilon_{t-2}, \cdots$ 의 linear combination으로 가정한다. ϵ_k의 분포는 $n(0, \sigma_\epsilon^2)$으로 가정하며 white

noise(백색잡음)라 한다.

$$x_t = \mu + \psi_0\epsilon_t + \psi_1\epsilon_{t-1} + \cdots \tag{2.6}$$

또는

$$x_t = \mu + \sum_{j=0}^{\infty} \psi_j\epsilon_{t-j} \tag{2.7}$$

ψ_j는 weight이며 μ는 시계열 데이타의 정도를 결정하는 상수로 본다. 보통 $\psi_0 = 0$. 식 2.6을 backward shift 연산자 B를 정의하여 x_t의 식으로 표현하자.

$$B\epsilon_t = \epsilon_{t-1}$$

일반적으로 연산자 B는 다음과 같이 표현된다.

$$B^j\epsilon_t = \epsilon_{t-j} \tag{2.8}$$

식 2.6은

$$\begin{aligned} x_t &= \mu + (\psi_0 B^0\epsilon_t + \psi_1 B^1\epsilon_t + \psi_2 B^2\epsilon_t + \cdots \\ &= \mu + (\psi_0 B^0 + \psi_1 B^1 + \psi_2 B^2 + \cdots)\epsilon_t \\ &= \mu + \Psi(B)\epsilon_t \end{aligned}$$

여기서 $\Psi(B) = \psi_0 B^0 + \psi_1 B^1 + \psi_2 B^2 + \cdots (\ \psi_0 = 0)$이다. 식 2.6을 linear filter라 하며 다음 특징을 갖고 있다.

- x_t는 $x_{t-1}, x_{t-2}, \cdots$에 dependent하다. 왜냐하면 각 x_t는 이전의 $\{\epsilon_k \mid k = 1, 2, \cdots\}$로부터 결정된다.
- $\{\epsilon_t\}$가 normal distribution을 따르면 $\{x_t\}$도 normal distribution을 따른다.

Linear filter 관점에서 white noise 시계열 예측 모델을 다른 시계열 예측 모델로 함수 변형(function transformation)을 생각할 수 있다. 이러한 linear filter

측면에서 설정된 Box 와 Jenkins 모델이 있다. Linear filter는 stationary 와 nonstationary 시계열 예측에 사용가능하다. Stationary 모델이란 시간의 변화에 대해 통계적 특징(statistical property)이 변화하지 않는 시계열 모델을 말한다[6]. 시계열 데이타 $x_1, x_2, x_3, \cdots, x_t, \cdots$가 같은 확률밀도함수(probability density function)을 갖는다.

$$\begin{aligned} x_1, x_2, x_3, \cdots, x_t, \cdots &\longrightarrow \textit{the same p.d.f} \\ \{x_1, x_2\}, \{x_3, x_4\}, \cdots, \{x_{t-1}, x_t\}, \cdots &\longrightarrow \textit{the same j.p.d.f} \\ \vdots \qquad\qquad &\quad \vdots \end{aligned}$$

그리고 시간의 변화에 대해 통계적 특징(statistical property)이 변화하는 시계열 모델을 nonstationary 모델이라고 한다[20]. Linear filter 에서 weight sequence $\{\psi_i\}$가 infinte 와 divergent하는 모델을 nonstationary 모델이라고 한다. 표 2.3.1는 stationary 와 nonstationary의 특징을 비교한 것이다.

Models	Mean	Sequence of $\{\psi_j\}$
Stationary	Constant mean	Finite and Convergent Infinit and Convergent
Nonstationary	No mean	Infinit and Divergent

표 2.2: Stationary 와 nonstationary 비교

통계치(Statistics)

시계열 데이타 관측치 $x_1, x_2, x_3, \cdots$로부터 시계열 모델을 구성하는데 필요한 통계치(statistic)를 기술한다. 만약 x_i가 확률밀도함수 $p(x_i)$를 따르며 관측치 $x_1, x_2, x_3, \cdots, x_n$ 에 대하여

- 평균(mean) :

$$\tilde{\mu} = \frac{1}{n} \sum_{j=1}^{n} x_j$$

- 분산(variance) :

$$\tilde{\sigma}_x^2 = \frac{1}{n}\sum_{j=1}^{n}(x_t - \tilde{\mu})^2$$

- Autocovariance : 시계열 데이타를 stationary로 가정하자. 일정시간 k만큼 떨어진 모든 (x_t, x_{t+k})에 대한 결합확률분포(joint probability distribution) $p(x)$는 동일하다. 결합확률분포의 특성은 lag k로 구분되는 시계열데이타 $\{x_t, x_{t+k}, x_{t+2k}, \cdots\}$의 scatter disgram으로 부터 알수있다. 또한 시계열데이타의 lag k에서 이웃한 값들의 상관계수(correlation coefficient)을 고려할 수 있다. 시계열 데이타를 stationary로 가정하면 (x_t, x_{t+k})의 covariance는 모든 t에 대하여 같은 값을 갖을때 covariance를 lag k에서 autocovariance γ_k라 한다.

$$\gamma_k = Cov(x_t, x_{t+k}) = E[(x_t - \mu)(x_{t+k} - \mu)]$$

- Autocorrelation Coefficient : Lag k에서 autocorrelation coefficient ρ_k는

$$\begin{aligned}
\rho_k &= \frac{E[(x_t - \mu)(x_{t+k} - \mu)]}{\sqrt{E[(x_t - \mu)]^2 E[(x_{t+k} - \mu)]^2}} \\
&= \frac{E[(x_t - \mu)(x_{t+k} - \mu)]}{\sigma_x^2} \\
&= \frac{\gamma_k}{\gamma_0} \quad (\text{since } \sigma_x^2 = \gamma_0).
\end{aligned}$$

2.3.2 AR(p), Autoregressive Model

식 2.6을 시계열 예측에 사용하기 위하여 ψ_t을 측정하여야 한다. 그러나 infinite인 ϕ_t를 계산하는 것은 가능하지 않다. 시간 t에서 예측치 x_t가 단지 p 개의 이전 관측치 $x_{t-1}, x_{t-2}, \cdots, x_{t-p}$ 와 현재 t시간에서 error ϵ_t로 결정되는 모델이 autoregressive 예측 모델이다.

$$x_t = \xi + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \epsilon_t$$

$$= \xi + \sum_{j=1}^{p} \phi_j x_{t-j} + \epsilon_t \tag{2.9}$$

식 2.9은 p개의 알려진 $x_{t-1}, x_{t-2}, \cdots, x_{t-p}$ 와 unknown parameter ϕ_1, ϕ_2, $\cdots$, ϕ_p의 함수이다. 식 2.8을 이용하여 식 2.9을 변형시켜 보면,

$$\begin{aligned} x_t &= \xi + \phi_1 B^1 x_t + \phi_2 B^2 x_t + \cdots + \phi_p B^p x_t + \epsilon_t \\ &= \xi + (\phi_1 B^1 + \phi_2 B^2 + \cdots + \phi_p B^p) x_t + \epsilon_t \\ &= \xi + x_t \sum_{j=1}^{p} \phi_j B^j + \epsilon_t \end{aligned} \tag{2.10}$$

그러므로,

$$(1 - \sum_{j=1}^{p} \phi_j B^j) x_t = \xi + \epsilon_t. \tag{2.11}$$

여기서 $\Phi_p(B) = 1 - \sum_{j=1}^{p} \phi_j B^j$ 로 정의하면 식 (2.9)는

$$\Phi_p(B) x_t = \xi + \epsilon_t. \tag{2.12}$$

시계열 예측치가 평균으로부터 차이 $\tilde{x}_t = x_t - \mu$로 결정된다고 하자.

$$\Phi_p(B) \tilde{x}_t = \epsilon_t \tag{2.13}$$

식 2.12와 식 2.13로부터 μ는 다음과 같다.

$$\mu = \frac{\xi}{1 - \sum_{j=1}^{p} \phi_j}$$

Note

식 2.13으로 부터

$$\begin{aligned} \Phi_p(B)(\tilde{x}_t + \mu) &= \xi + \epsilon_t \\ \Phi_p(B)\tilde{x}_t + \Phi_p(B)\mu &= \xi + \epsilon_t. \end{aligned}$$

여기서 $\Phi_p(B)\mu = \xi$로 놓고 μ를 계산하면 된다. AR(p)는 stationary 및 nonstationary 시계열 예측 모델에 적용 가능하다. 만약 $\Phi(B)$의 해(root)가 절대값으로 1보다 크면(단위 원의 외부에 존재) 이 시계열 예측은 stationary 경우가 된다 [6, 8]. 표 2.3.2은 AR(1), AR(2)를 비교한 것이다.

	First-order, AR(1)	Second-order, AR(2)
Model	$x_t = \xi + \phi_1 x_{t-1} + \epsilon_t$	$x_t = \xi + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \epsilon_t$
Condition	$\mid \phi_1 \mid < 1$	$\phi_1 + \phi_2 < 1$ $\phi_2 - \phi_1 < 1$ $\mid \phi_2 \mid < 1$
Mean	$\mu = \frac{\xi}{1-\phi_1}$	$\mu = \frac{\xi}{1-\phi_1-\phi_2}$
Variance	$\gamma_k = \phi_1^k \frac{\sigma_\epsilon^2}{1-\phi_1^2},\ k = 0, 1, 2, \cdots$	$\gamma_k = \phi_1 r_{k-1} + \phi_2 r_{k-2}, k > 0$
Autocorrelation	$\rho_k = \phi_1^k,\ k = 0, 1, 2, \cdots$	$\rho_1 = \phi_1 + \phi_2 \rho_1$ $\rho_2 = \phi_1 \rho_1 + \phi_2$
Characteristic		

표 2.3: AR(1) 와 AR(2)의 비교

2.3.3 MA(q), Moving Average Model

식 2.6의 또 다른 변형으로 시간 t에서 예측치 x_t를 이전의 q개의 관측치들과 error ϵ_t의 linear combination으로 구성시킨다.

$$\begin{aligned} x_t &= \mu + \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} \cdots - \theta_q \epsilon_{t-q} \\ &= \mu + \epsilon_t - \sum_{k=1}^{q} \theta_k \epsilon_{t-k} \end{aligned} \tag{2.14}$$

AR(p)에서와 미찬가지로 ϵ에 대해 backward operator를 도입하면 MA(q)는

$$\begin{aligned} B^j \epsilon_t &= \epsilon_{t-j} \\ x_t &= \mu + \epsilon_t - \sum_{k=1}^{q} \theta_k (B^k \epsilon_t) \end{aligned}$$

$$= \mu + \epsilon_t(1 - \sum_{k=1}^{q} \theta_k B^k). \tag{2.15}$$

Linear 항을 $\Theta_q(B) = (1 - \sum_{k=1}^{q} \theta_k B^k)$로 놓으면

$$x_t = \mu + \Theta_q(B)\epsilon_t \tag{2.16}$$

MA(q) 시계열 예측을 하는데는 단지 q개의 weight θ_k만 요구된다. x_t의 평균과 분산은 ϵ_t가 $n(0, \sigma_\epsilon)$ 따른다고 할때 다음과 같다.

$$\begin{aligned} E(x_t) &= E(\mu + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \cdots - \theta_q\epsilon_{t-q}) \\ &= \mu \qquad (2.17) \\ \gamma_0 &= V(x_t) \\ &= V(\mu + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \cdots - \theta_q\epsilon_{t-q}) \\ &= \sigma_\epsilon^2 \sum_{k=0}^{q} \theta_k^2 \qquad (\theta_0 = 1). \qquad (2.18) \end{aligned}$$

시간 k(the kth lag)에서 autocovariance는

$$\begin{aligned} \gamma_k &= E[(\epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \cdots - \theta_q\epsilon_{t-q}) \\ &\quad \times (\epsilon_{t-k} - \theta_1\epsilon_{t-k-1} - \theta_2\epsilon_{t-k-2} - \cdots - \theta_q\epsilon_{t-k-q})]. \end{aligned}$$

그로므로,

$$\gamma_k = \begin{cases} \sigma_\epsilon^2(-\theta_k + \theta_1\theta_{k+1} + \theta_2\theta_{k+2} + \cdots + \theta_{q-k}\theta_q) & k = 1, 2, \cdots, q \\ 0 & k > q. \end{cases} \tag{2.19}$$

지금까지 구한 평균과 분산을 가지고 MA(q)의 autocorrelation 함수는 다음과 같다.

$$\rho_k = \begin{cases} \frac{-\theta_k+\theta_1\theta_{k+1}+\theta_2\theta_{k+2}+\cdots+\theta_{q-k}\theta_q}{1+\theta_1^2+\theta_2^2+\cdots+\theta_q^2} & k = 1, 2, \cdots, q \\ 0 & k > q \end{cases} \tag{2.20}$$

표 2.3.3은 MA(1), MA(2)를 비교한 것이다.

Models	First-order, MR(1)	Second-order, MR(2)
	$x_t = \mu + \epsilon_t - \theta_1\epsilon_{t-1}$	$x_t = \xi + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2}$
Condition		
Mean	$E(x_i) = \mu$	$E(x_i) = \mu$
Variance	$\gamma_0 = \sigma_\epsilon^2(1+\theta_1^2)$	$\gamma_0 = \sigma_\epsilon^2(1+\theta_1^2+\theta_2^2)$
Autocorrelation	$\rho_k = \frac{-\theta_1}{1+\theta_1^2},\ k=1$ $\rho_k = 0,\ k \neq 1$	$\rho_1 = \frac{-\theta_1(1-\theta_2)}{1+\theta_1^2+\theta_2^2}$ $\rho_2 = \frac{-\theta_2}{1+\theta_1^2+\theta_2^2}$ $\rho_k = 0,\ k > 2$
Characteristic		

표 2.4: MR(1) 와 MR(2)의 비교

Inevitability

MA(1)의 시계열 모델을 ϵ_t에 대해 전개해 보자.

$$\begin{aligned} \tilde{x}_t &= \epsilon_t - \theta_1\epsilon_{t-1} \\ &= \epsilon_t - \theta_1 B\epsilon_t \\ &= (1-\theta_1 B)\epsilon_t \\ \epsilon_t &= (1-\theta_1 B)^{-1}\tilde{x}_t \end{aligned} \tag{2.21}$$

$|\theta_1| < 1$로 가정하자.

$$\begin{aligned} \epsilon_t &= (\sum_{j=0}^{\infty}\theta_1^j B^j)\tilde{x}_j \\ &= (1+\theta_1 B^1+\theta_1^2 B^2+\cdots)\tilde{x}_t \end{aligned} \tag{2.22}$$

여기서 $\phi_j = -\theta_1^j$인 infinte-order autoregressive 시계열로 하면 MA(1)은 AR(∞)로 전환된다. 이런경우에 $|\theta_1| < 1$을 invertibility 조건이라한다. 일반

적으로 MA(q) 시계열 예측은 AR(∞)로 전환될 수 있는 조건은 $\Theta_q(B) = 0$ 이다. 정리하면

- MA(q)는 θ_i의 값에 관계없이 stationary 하며 $\Theta_q(B) = 0$를 만족하는 해가 단위 원의 외부에 존재하면 invertible 하다.
- AR(p)는 $\Phi_p(B) = 0$의 해가 단위 원의 외부에 존재하면 stationary 하며 모든 ϕ_i의 값에 관계없이 stationary 하다.

2.3.4 ARMA(p,q), Autoregressive-moving Average Model

시계열 모델을 설정하는데 있어 지금까지 논의된 두 모델 AR(p) 와 MA(q)를 혼합한 시계열 모델을 생각할 수 있다. 즉 시간 t에서 예측치 x_t는 이전에 예측되거나 관측된 $\{x_k \mid k = t-1, t-2, t-3, \cdots, t-p\}$ 와 error 항 $\{\epsilon_j \mid j = t, t-1, t-2, \cdots, t-q\}$으로 부터 추정된다.

$$\begin{aligned} x_t &= \xi + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} \\ &\quad -\theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \cdots - \theta_q \epsilon_{t-q} \\ &\quad +\epsilon_t \end{aligned} \tag{2.23}$$

Linear 항을 갖도록 식 (2.8)를 도입하여 전개하면

$$\Phi_p(B) = \xi + \Theta_q(B)\epsilon_t. \tag{2.24}$$

ARMA(p,q)는 AR(p) 와 MA(q) 모델의 특성을 내포하여 $\Phi_p(B) = 0$의 해가 단위가 1인 원의 외부에 존재하면 AR(p)가 되어 stationary 하며 $\Theta_q(B) = 0$의 해가 단위가 1인 원의 외부에 존재하면 MA(q)가 되어 invertibility 조건을 만족한다. ARMA(1,1)일때 살펴보면 모델은

$$x_t = \xi + \phi_1 x_{t-1} + \epsilon_t - \theta_1 \epsilon_{t-1} \tag{2.25}$$

평균, 분산 그리고 autocovariance는

$$\mu \equiv E(x_t) = \frac{\xi}{1-\phi_1} \tag{2.26}$$

$$\gamma_0 = \phi_1\gamma_1 + \sigma_\epsilon^2[1-\theta_1(\phi_1-\theta_1)]$$

$$\gamma_1 = \phi_1\gamma_0 - \theta_1\sigma_\epsilon^2 \tag{2.27}$$

$$\gamma_k = \phi_1\gamma_{k-1}, \qquad k \geq 2.$$

ARMA(1,1)의 평균은 AR(1)의 평균과 같으며 시간 t lag에 대한 autocovariance는 $(t-1)$ lag의 만 고려한다. Autocorrelation은

$$\rho_1 = \frac{(1-\phi_1\theta_1)(\phi_1-\theta_1)}{1+\theta_1^2-2\theta_1\phi_2} \tag{2.28}$$

$$\rho_k = \phi_1\rho_{k-1}, \qquad k \geq 2. \tag{2.29}$$

일반적으로 ARMA(p,q)의 autocorrelation은 처음의 q lag들은 AR(q)의 영향을 받으며 q 이상의 lag들은 AR(p)의 영향을 받게된다.

2.4 선형 Nonstationary 시계열 예측

Nonstationary인 시계열 예측문제는 임의 연산자를 도입하여 stationary 시계열 문제로 바꾸어 모델링한다. 시계열 데이타가 nonstationary인 경우는 평균에서 nonstationary 인 경우와 평균 및 기울기(slope)에서 nonstationary 인 경우등 다양하게 나타날수 있다[8]. Backward difference 연산자를 다음과 같이 정의하자.

$$\nabla^1 x_t = x_t - x_{t-1} \text{ (first difference)}$$

$$\nabla^2 x_t = x_t - 2x_{t-1} + x_{t-2} \text{ (second difference)}$$

$$\vdots$$

일반적으로 평균에서 nonstationary 인 경우 시계열데이타는 $\nabla^1 x_t$을 적용하면 stationary로 바뀌며 기울기(slope)에서 nonstationary 인 경우는 $\nabla^2 x_t$을 적용함으로써 stationary 모델로 변형된다.

시계열데이타 $\{x_t \mid t = 1, 2, 3, \cdots, n\}$에 d번 difference을 하면 항이 $n - d$인 시계열데이타 $\{w_t \mid t = 1, 2, 3, \cdots, n - d\}$가 된다.

$$\{w_t\} = \{\nabla^d x_t\}$$

그러므로

$$w_t = \nabla^d x_t = \nabla^d \tilde{x}_t.$$

2.4.1 ARIMA(p,d,q), Autoregressive Integrated Moving Average Model

Nonstationary 시계열데이타를 d차 difference하여 p개의 autoregressive 모수 및 q개의 moving average 모수를 가지고 설정되는 시계열 모델을 말한다.

$$\begin{aligned} \Phi_p(B)\nabla^d x_t &= \Theta_q(B)\epsilon_t \\ \Phi_p(B)_p w_t &= \Theta_q(B)\epsilon_t \end{aligned} \tag{2.30}$$

$\{w_t\}$의 0이 아닌 평균을 u_w 라 하면 식 2.30는

$$\Phi(B)_p w_t = u_w + \Theta_q(B)\epsilon_t.$$

실제 적용에서 p, q, d는 2를 초과하지 않으며 ARIMA(1,1,1) 및 ARIMA(2,1,0)를 주로 사용한다.

ARIMA(1,1,1)

$$\begin{aligned} (1 - \phi_1 B)\nabla x_t &= (1 - \theta_1 B)\epsilon_t \\ x_t &= \phi_1 B x_t + x_{t-1} - \phi_1 B x_{t-1} + \epsilon_t - \theta_1 \epsilon_t \end{aligned}$$

$$
\begin{aligned}
&= (1+\phi_1)x_{t-1} - \phi_1 x_{t-2} + \epsilon_t - \theta_1 \epsilon_{t-1} \\
&\quad \text{,since } Bx_t = x_{t-1}, B\epsilon_t = \epsilon_{t-1}
\end{aligned}
$$

ARIMA(2,1,0)

$$
\begin{aligned}
(1 - \phi_1 B^1 - \phi_2 B^2)\nabla x_t &= \epsilon_t \\
x_t &= \phi_1 B^1 x_{t-1} + \phi_2 B^2 x_t + x_{t-1} \\
&\quad -\phi_1 B^1 x_{t-1} \phi_2 B^2 x_{t-1} + \epsilon_t \\
&= x_{t-1} + \phi_1 (x_{t-1} - x_{t-2}) \\
&\quad +\phi_2 (x_{t-2} - x_{t-3}) + \epsilon_t \\
&\quad \text{,since } B^j x_t = x_{t-j}
\end{aligned}
$$

그림 2.2은 시계열 모델간의 관계을 보여준다.

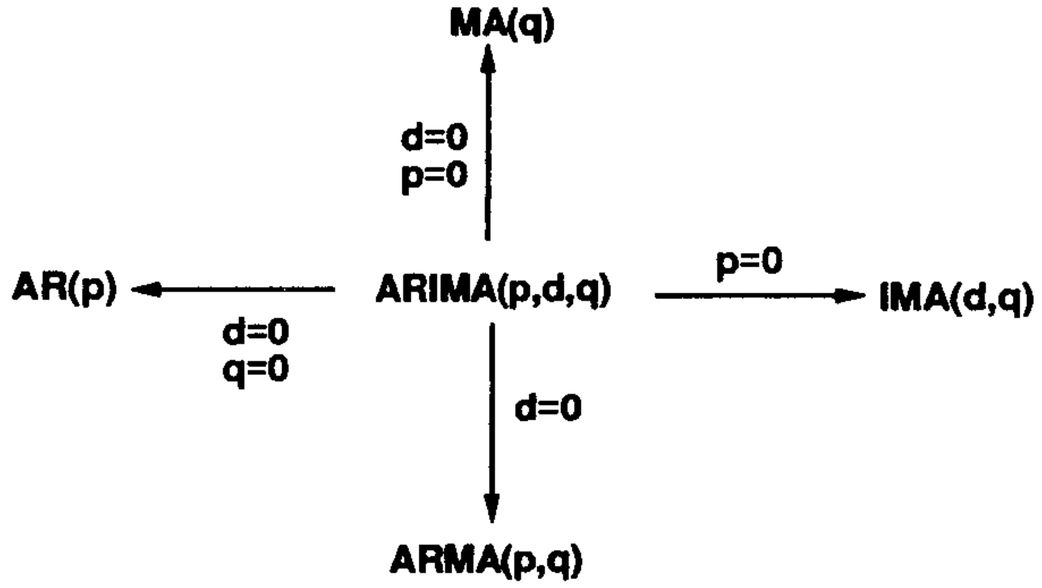


그림 2.2: 시계열 모델간의 관계

2.4.2 Identification

관측된 시계열데이타로 부터 autocorrelation 함수 및 partial autocorrelation 함수를 구하여 이론적 시계열 모델과 비교, 적용할 모델을 선정한다. N개의 시계열데이타에 대하여 K(보통 $K \leq N/4$)인 sample autocorrelation r_k는

$$
r_k = \frac{\sum_{i=1}^{N-k}(x_t - \bar{x})(x_{t-1} - \bar{x})}{\sum_{i=1}^{N}(x_t - \bar{x})^2}, \quad k = 0, 1, 2, 3, \cdots, K.
$$

Partial autocorrelation ϕ_k는 order k의 autoregressive 시계열에서 k 번째 coefficient가 되며 Yule-Walker 방정식을 만족하여야 한다[8, 6, 20].

$$\begin{aligned}
\rho_1 &= \phi_1 + \phi_2\rho_2 + \cdots + \phi_K\rho_{K-1} \\
\rho_2 &= \phi_1\rho_1 + \phi_{k2} + \cdots + \phi_K\rho_{K-2} \\
&\vdots \\
&\vdots \\
\rho_K &= \phi_1\rho_{K-1} + \phi_2\rho_{K-2} + \cdots + \phi_K
\end{aligned}$$

행렬식으로 표현하면,

$$\boldsymbol{\rho}_K = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_K \end{pmatrix}, \mathbf{P_K} = \begin{pmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{K-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{K-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \rho_{K-1} & \rho_{K-2} & \rho_{K-3} & \cdots & 1 \end{pmatrix}, \mathbf{\Phi_K} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_K \end{pmatrix}$$

그러므로,

$$\begin{aligned}
\boldsymbol{\rho}_K &= \mathbf{P_K}\mathbf{\Phi_K} \\
\tilde{\mathbf{\Phi}}_\mathbf{K} &= \mathbf{P}_\mathbf{K}^{-\mathbf{1}}\boldsymbol{\rho}_K
\end{aligned}$$

$\tilde{\rho}_j$의 estimate로 sample autocorrelation r_k으로 대치시켜 ϕ_k를 추정한다. 계산된 sample 과 partial autocorrelation 함수를 plotting하여 이론적인 모델(표 2.4.2)과 비교한다. *tails off*은 함수 값이 k의 변화에 대해 exponential, sinusoidal 혹은 geometric 중 하나의 경향(trend)을 띄며 감소하는 것을 의미하며 *cutts off*은 함수 값이 k의 변화에 갑자기 변화하는 상태를 말한다.

Nonstationary 시계열데이타의 sample 및 partial autocorrelation 함수는 k의 변화에 대해 아주 적게 쇠퇴하거나 또는 증가하는 경향이 있다. 이런 경우 difference 연산자를 이용하여 stationary 시계열로 변화시켜 sample 및 partial autocorrelation 함수를 표 2.4.2와 비교한다.

모델	Autocorrelation 함수	Partial autocorrelation 함수
AR(p)	Tails off	Cuts off after lag q
MA(q)	Cuts off after lag q	Tails off
ARMA(p,q)	Tails off	Tails off

표 2.5: Stationary 모델의 sample 과 partial autocorrelation 함수 비교

2.4.3 Estimation

시계열 모델 identification에서 모델이 잠정적으로 결정되면 모수 estimation를 수행한다. 시계열 모델에서 모수 추정은 목적하는 바에 따라 partial derivative로 부터 추정된다. AR(p)에서 linear least square로 모수 추정을 하면

$$\begin{aligned} x_t &= \xi + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \epsilon_t \\ \epsilon_t &= x_t - \xi - \phi_1 x_{t-1} - \phi_2 x_{t-2} - \cdots - \phi_p x_{t-p} \end{aligned}$$

그러므로,

$$\begin{aligned} \frac{\partial \epsilon_t}{\partial \phi_i} &= -x_{t-i} \\ \frac{\partial \epsilon_t}{\partial \xi} &= -1 \end{aligned}$$

관측치가 n개가 존재할때 AR(p)인 경우를 행렬식으로 표현하면 다음과 같다.

$$x_t = \xi + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \epsilon_t$$

$$\begin{pmatrix} x_3 \\ x_4 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} 1 & x_2 & x_1 \\ 1 & x_3 & x_2 \\ \vdots & \vdots & \vdots \\ 1 & x_{n-2} & x_{n-3} \\ 1 & x_{n-1} & x_{n-2} \end{pmatrix} \begin{pmatrix} \xi \\ \phi_1 \\ \phi_2 \end{pmatrix} + \begin{pmatrix} \epsilon_3 \\ \epsilon_4 \\ \vdots \\ \epsilon_{n-1} \\ \epsilon_n \end{pmatrix}$$

또는

$$\mathbf{x} = \mathbf{Z}\Phi + \epsilon.$$

그러므로 AR(2)의 least square 추정은

$$\tilde{\Phi} = (\mathbf{Z}\mathbf{Z}')^{-1}\mathbf{Z}'\mathbf{x}$$

그러나 MA(q)의 경우에는 모수 추정이 쉽지 않다. MA(1)의 linear least square 방법을 이용한 추정은

$$x_t = \epsilon_t - \theta_1\epsilon_{t-1}$$

또는

$$\epsilon_t = (1 - B\theta_1)^{-1}$$

그러므로,

$$\frac{\partial \epsilon_t}{\partial \theta_1} = B(1 - B\theta_1)^{-2}.$$

또한 sample과 partial autocorrelation 함수의 standard error 측정은 lag k에서 autocorrelation 값을 아는데 도움을 준다. Lag k번째 sample autocorrelation 함수의 standard error, $S(r_k)$는

$$S(r_k) = \frac{1}{\sqrt{N}}\sqrt{1 + 2\sum_{j=1}^{K-1} r_j^{*2}}, \quad \text{where} \quad r_j^* = \begin{cases} r_j & \text{if } \rho_j \neq 0 \\ 0 & \text{if } \rho_j = 0. \end{cases}$$

Lag k에서 sample partial autocorrelation의 coefficient의 standard error, $S(\tilde{\phi}_k)$,는

$$S(\tilde{\phi}_k) \approx \frac{1}{\sqrt{N}}.$$

일반적으로 lag k에서 각 autocorrelation 함수의 절대값($\mid S(r_k) \mid$, $\mid S(\tilde{\phi}_k) \mid$)은 자신의 standard error 의 2배보다 작으면 0으로 간주한다.

MA(1)은 보통의 least square 방법으로 모수 추정을 할수 없다. Nonlinear 시계열 모수측정을 위한 방법으로 Marquardt 방법, Draper 와 Smith 방법, Box

와 Jenkins 방법 등이 있다[6, 8].

2.4.4 Diagnostic Checking

설정된 시계열 모델의 accuracy 측정을 하여 적합성(adequacy)을 판정한다. $\{e_t = x_t - \tilde{x}_t \mid t = 1, 2, 3, \cdots, n\}$로부터 autocorrelation 함수 $\{r_e(k)\}$를 계산하여

- 각 lag k에서 $r_e(k) \approx 0$이면 설정된 시계열 모델은 적합함
- 각 lag k에서 $r_e(k) \gg 0$이면 부적합

을 판정한다. 설정된 모델이 좋은 fittness을 제공하고 모수의 추정이 잘되었다면 residual autocorrelation의 standard error는 $\frac{1}{\sqrt{N}}$안에 존재한다. 만약 residual autocorrelation 값이 0이 아닌 값을 갖게되면 적합성(adequacy)을 판정을 확률밀도 함수 *Chi-square* 검증으로 알수 있다. 모델이 적합하다면 다음 검증 통계치 Q는 자유도 $K - p - q$인 *Chi-square* 내어 있게 된다.

$$Q = (n - d) \sum_{k=0}^{K} r_e^2(k)$$

d는 difference의 차수이다. 만약 Q가 매우 크면 부적합하다고 본다.

2.4.5 Forecasting

적당한 모델이 결정되었으면 미래 관측치에 대한 예측이 진행된다. 현재 시간을 T라 할때 앞으로 $T + \tau$동안 시계열 예측이 목표이며, $\tilde{x}_{T+\tau}$는 현재 시간 T에서 예측된 $T + \tau$에서의 예측치 $\tilde{x}_{T+\tau}(T)$라 하자. 일반적으로 예측치 $\tilde{x}_{T+\tau}(T)$는 예측치 $\tilde{x}_{T+1}(T)$, $\tilde{x}_{T+2}(T)$, $\cdots$,$\tilde{x}_{T+\tau-1}(T)$로부터 연속적으로 예측된다.

x_{T+j}는 시간 T에서 예측치 $\tilde{x}_{T+j}(T)$가 되고 $\epsilon_{T+j}(T)$는 0으로 한다. 그리고 single point 예측 오류 ϵ_{T-j}의 계산은 다음과 같다.

$$e_1(T - j) \equiv x_{T-j} - \tilde{x}_{T-j}(T - j - 1)$$

처음 예측시점 T에서 $\epsilon_{T+j} = 0 (j = 1, 2, \cdots, \tau)$으로 가정한다.

ARIMA(1,1,1)에서 $T + \tau$에서 예측은

$$(1 - \phi_1 B)\nabla x_{T+\tau} = (1 - \theta_1 B)\epsilon_{T+\tau}.$$

그러므로

$$x_{T+\tau} = (1 + \phi_1)x_{T+\tau-1} - \phi_1 x_{T+\tau-2} + \epsilon_{T+\tau} - \theta_1 \epsilon_{T+\tau-1}.$$

$\tau = 1$일때

$$E[x_{T+1}] \equiv \tilde{x}_{T+1}(T) = (1 + \phi_1)x_T - \phi_1 x_{T-1} - \theta_1 e_1(T).$$

여기서 $E[x_{T+\tau}] = 0$, $\epsilon_T \equiv E[e_1(T)] = x_T - \tilde{(x)}_T(T-1)$. $\tau = 2$일때

$$E[x_{T+2}] \equiv \tilde{x}_{T+2}(T) = (1 + \phi_1)\tilde{x}_{T+1}(T) - \phi_1 \tilde{x}_T(T).$$

$\tilde{x}_{T+1}(T)$와 $\tilde{x}_T(T)$는 $\{x_t\}$와 $\{\epsilon_T\}$의 difference equation으로 표현된다. 그러므로 시간 T에서 $T + \tau$의 예측은

$$\begin{aligned} x_{T+\tau} &= \psi_1 \epsilon_{T+\tau-1} + \psi_2 \epsilon_{T+\tau-2} + \cdots + \psi_{\tau-1}\epsilon_{T+1} \\ &\quad \psi_\tau \epsilon_T + \psi_{\tau+1}\epsilon_{T-1} + \cdots + \epsilon_{T+\tau}. \end{aligned}$$

이식을 주어진 예측 모델의 random shock 이라 한다. 그러므로 $t > T$에 대하여 $\epsilon_t = 0$, $t \leq T$인 ϵ_t의 추정은 $e_1(t)$로 하면,

$$\tilde{x}_{T+\tau}(T) = \psi_\tau e_1(T) + \psi_{T+1} e_1(T-1) + \cdots. \tag{2.31}$$

Linear filter 정의로부터

$$\begin{aligned} x_t &= \Psi(B)\epsilon_t \\ \epsilon_t &= \frac{x_t}{\Psi(B)}. \end{aligned} \tag{2.32}$$

식 (2.32)을 ARIMA(p,d,q)에 대입하여 전개하면

$$\Phi_p(B)\nabla^d x_t = \Theta_q(B)\frac{x_t}{\Psi(B)}.$$

또는

$$\Psi(B)\Phi_p(B)(1-B)^d x_t = \Theta(B)x_t.$$

그러므로

$$\begin{aligned}&(\psi_0+\psi_1 B+\cdots)(1-\phi_1 B-\phi_2 B^2-\cdots-\phi_p B^2)(1-B)^d\\ =\ &(1-\theta_1 B-\theta_2 B^2-\cdots-\theta_q B^q).\end{aligned} \tag{2.33}$$

예를 들어 ARMA(1,1)인경우에

$$\begin{aligned}(\psi_0+\psi_1 B+\cdots)(1-\phi_1 B) &= (1-\theta_1 B)\\ \psi_0+(\psi_1-\psi_0\phi_1)B+(\psi_2-\psi_1\phi_1)B^2+\cdots &= 1-\theta_1 B.\end{aligned}$$

그러므로

$$\begin{aligned}\psi_0 &= 1\\ \psi_1 &= \phi_1-\theta_1\\ \psi_2 &= \phi_1(\phi_1-\theta_1)\\ &\vdots\\ \psi_j &= \phi_1^{j-1}(\phi_1-\theta_1).\end{aligned}$$

Random shock을 이용하면 각 예측시간에 대한 예측 보정이 가능하며[6], 앞선 τ 시간 예측에 대한 예측구간을 알수 있다. τ-step-ahead 예측 오류는

$$e_\tau(T) = \epsilon_{T+\tau}+\psi_1\epsilon_{T+\tau-1}+\psi_2\epsilon_{T+\tau-2}+\cdots+\psi_{\tau-1}\epsilon_{T+1}.$$

$e_\tau(T)$의 분산은

$$V[e_\tau(T)] = \sigma_\epsilon^2(1 + \sum_{j=1}^{\tau-1} \psi_j^2).$$

그러므로, $T+\tau$의 예측에 대한 대략적인 $100(1-\alpha)$ 퍼센트 예측구간(prediction interval)은

$$\tilde{x}_{T+\tau}(T) \pm u_{1-\alpha/2}\tilde{\sigma}_\epsilon(1 + \sum_{j=1}^{\tau-1} \psi_j^2)^{1/2}.$$

여기서 $u_{1-\alpha/2}$는 $100(1-\alpha)$ 값을 갖는 표준정규분포로 부터 알수 있다. $\{\tilde{\psi}_j\}$의 추정은 $\{\psi_j\}$을 이용하며 $\tilde{\sigma}_\epsilon$의 추정은

$$\tilde{\sigma}_\epsilon = \frac{SS(\tilde{\phi}, \tilde{\theta})}{r-m}.$$

$SS(\tilde{\phi}, \tilde{\theta})$은 모델 모수들 간의 sum of squares 이며 r은 residual 수 m은 추정된 모수의 수 이다.

2.5 예 제

이 절에서 지금까지 기술한 통계적 시계열 데이타 예측 예제를 다룬다. 예제에서 보이는 시계열 데이타는 stationary와 nonstationary 데이타인 경우로 나누워 다룬다.

Stationary time series

그림 2.3은 화학공정에서 나타나는 viscosity 데이타이다. 이 데이타는 stationary 시계열 데이타로서 시계열 예측 모델을 구성하여 생산품의 질을 높이기 위한 관리을 목표로하고 있다. 그림 2.5는 시계열 예측 모델 fitting 결과를 나타낸다.

단계 1 Sample 데이타 분포

- 100개 점도를 나타내는 수치 데이타
- 데이타 분포를 보아 stationary time series임을 알수 있슴

단계 2 시계열 모델 설정(그림 2.4

- sample autocorrelation 함수 - tails off
- sample partial autocorrelation 함수 - lag 2 후에 cuts off

그러므로 AR(2) 시계열 모델을 잠정적으로 결정

$$x_t = \xi + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \epsilon_t$$

단계 3 Preliminary estimate($\xi,\ \phi_1,\ \phi_2$)

- sample autocorrelation 함수 r_k을 가지고 $\tilde{\phi}_k$를 Yule-Walker 방정식으로 계산. $r_1 = 0.51$와 $r_2 = -0.09$을 대입하면

$$\begin{aligned} 0.05 &= \phi_1 - 0.09\phi_2 \\ -0.09 &= 0.51\phi_1 + \phi_2. \end{aligned}$$

그리고,

$$\xi = E(x_t)(1 - \phi_1 - \phi_2).$$

그러므로 $\phi_1 = 0.6256,\ \phi_2 = -0.3137,\ \xi = 19.64.$

단계 4 Actual estimate($\tilde{\xi},\ \tilde{\phi}_1,\ \tilde{\phi}_2$)

단계 5 Diagnostic Checking

- Residual autocorrelation 함수
- Chi-square 검증

Nonstationary time series

그림 2.6은 지난 100주 동안에 컨테이너에 대한 수요 시계열 데이타를 나타내고 있다. 컨테이너 공정은 몰딩 단계를 통하여 약국에서 처방약 보관에 사용된다. 이 데

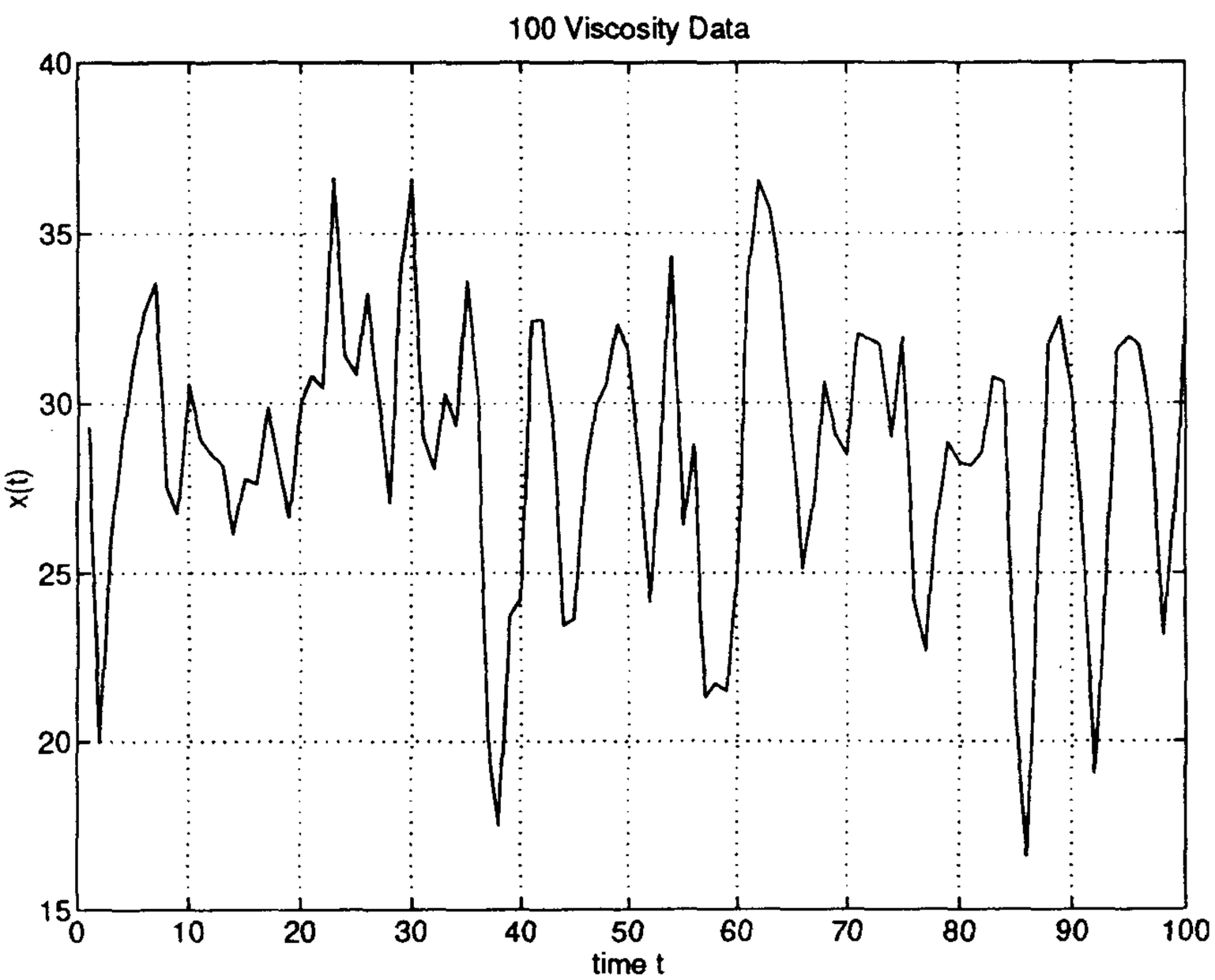


그림 2.3: 화학공정에서 viscosity 시계열 데이타

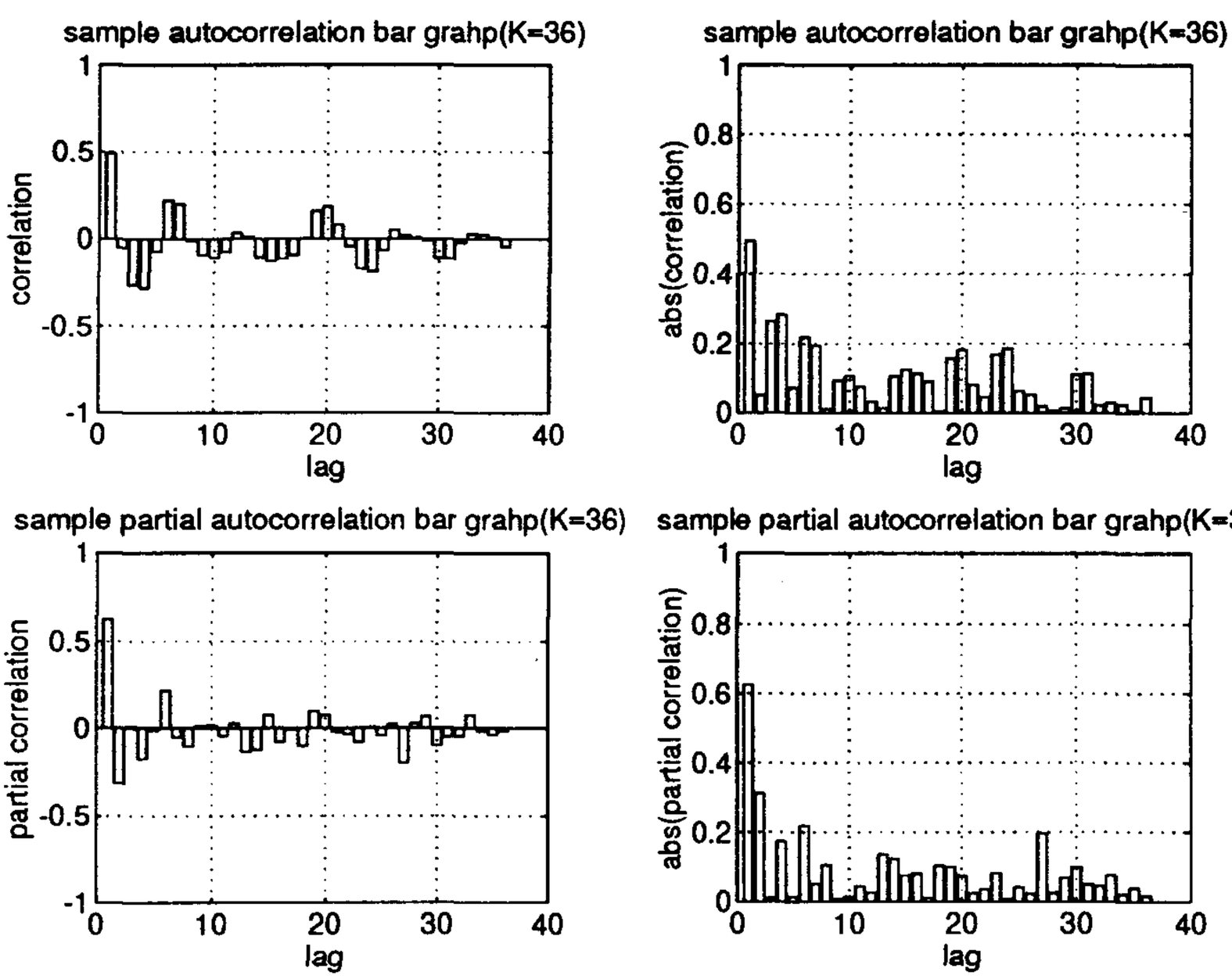


그림 2.4: Sample 와 sample partial autocorrelation

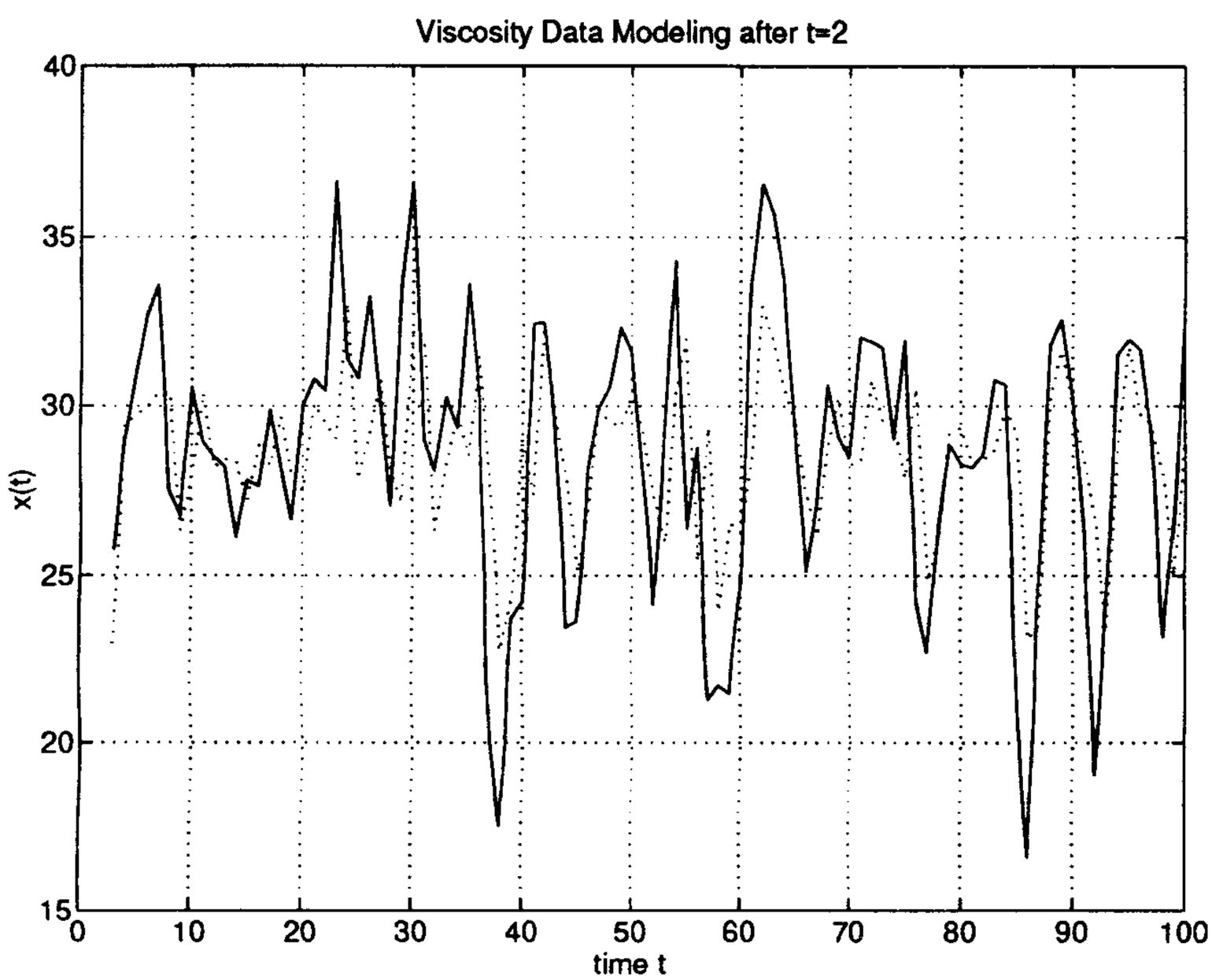


그림 2.5: 시계열 예측 모델 fitting 결과

이타는 nonstationary 시계열 데이타로서 시계열 예측 모델을 구성하여 컨테이너 수요 관리을 목표로하고 있다. 그림 2.8는 시계열 예측 모델 fitting 결과를 나타낸다.

단계 1 Sample 데이타 분포

- 100주의 플라스틱 제품 수요량
- 기울기에서 nonstationary time series임을 알수 있슴

단계 2 시계열 모델 설정

- sample의 autocorrelation 함수 - lag k의 변화에 대해 함수값이 아주 적은 변화를 가지고 쇠퇴, 또는 증가
- 1차 difference을 한후(그림 2.7)
  - sample autocorrelation 함수 - lag 1이후 cuts off
  - sample partial autocorrelation 함수 - tails off

  그러므로 IMA(1,1)을 잠정적으로 결정

$$\begin{aligned} \nabla x_t &= (1 - B\theta_1)\epsilon_t \\ x_t &= x_{t-1} + \epsilon_t - \theta_1\epsilon_{t-1} \end{aligned}$$

단계 3 Preliminary estimate(θ_1)

- r_1을 ρ_1로 대치하여 θ_1을 계산

$$0.4 = -\frac{\theta_1}{1+\theta_1^2}$$

  여기서 $\theta_1 = -1.9176,\ or\ -0.5215$이나 $-1 < \theta < 1$임으로 $\theta_1 = -0.5215$으로 선택

단계 4 Actual estimate($\tilde{\theta}_1$)

단계 5 Diagnostic Checking

- sample 로부터 residual을 계산($\epsilon_1 = \epsilon_2 = 0$)

$$e_k = x_k - \tilde{x}_k$$

- residual autocorrelation
- Chi-square 검증

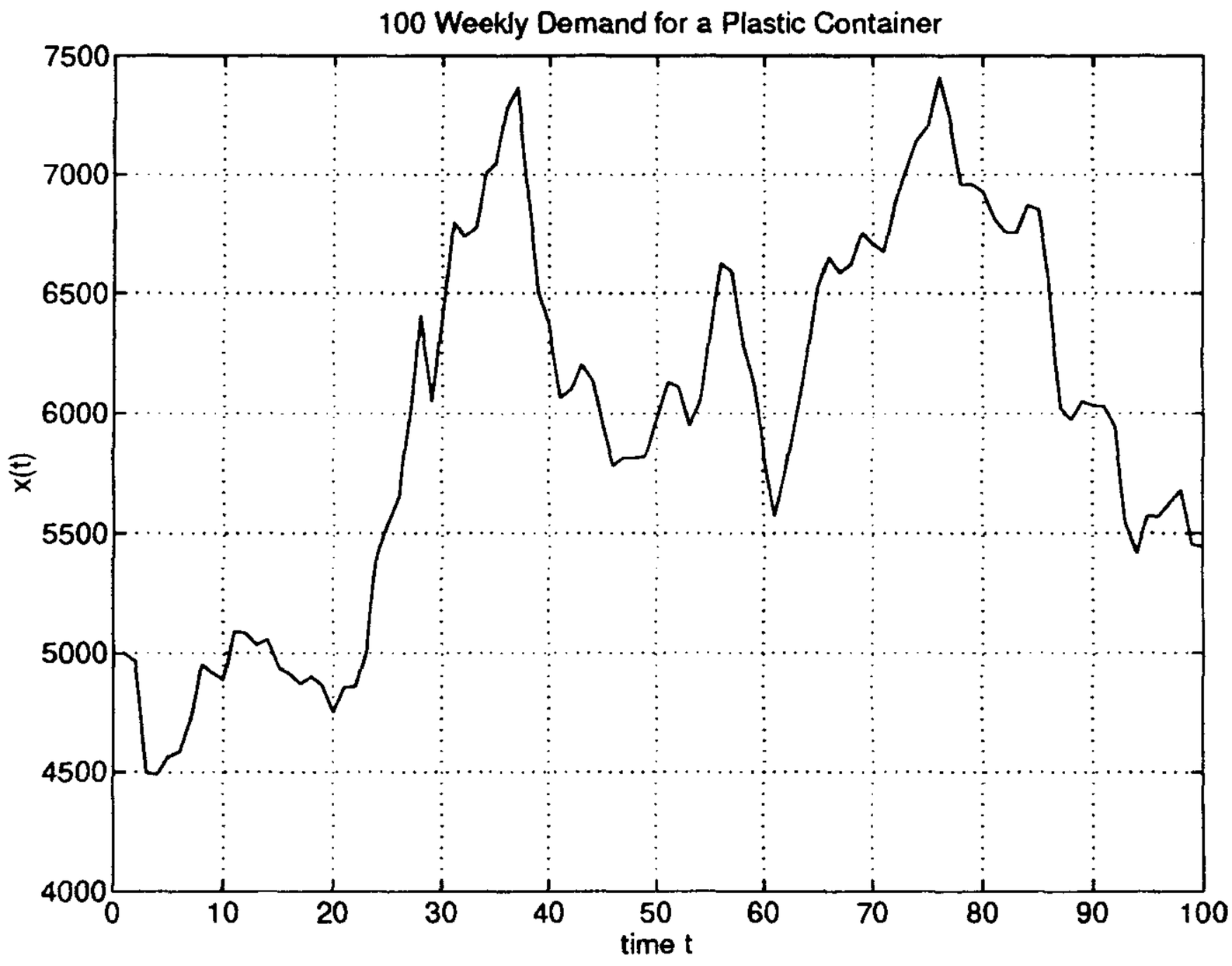


그림 2.6: 컨테이너 수요 시계열 데이타

2.6 Delay Coordinate Embedding을 사용한 시계열 예측

이 절에서는 delay coordinate embedding에 기반한 time series의 short-term 예측 알고리즘을 기술하고자 한다. Filter 된 delay coordinates가 dynamical attractor를 전개할 수 있을 만큼 큰 공간상에 있는 time series를 재구성하는 데

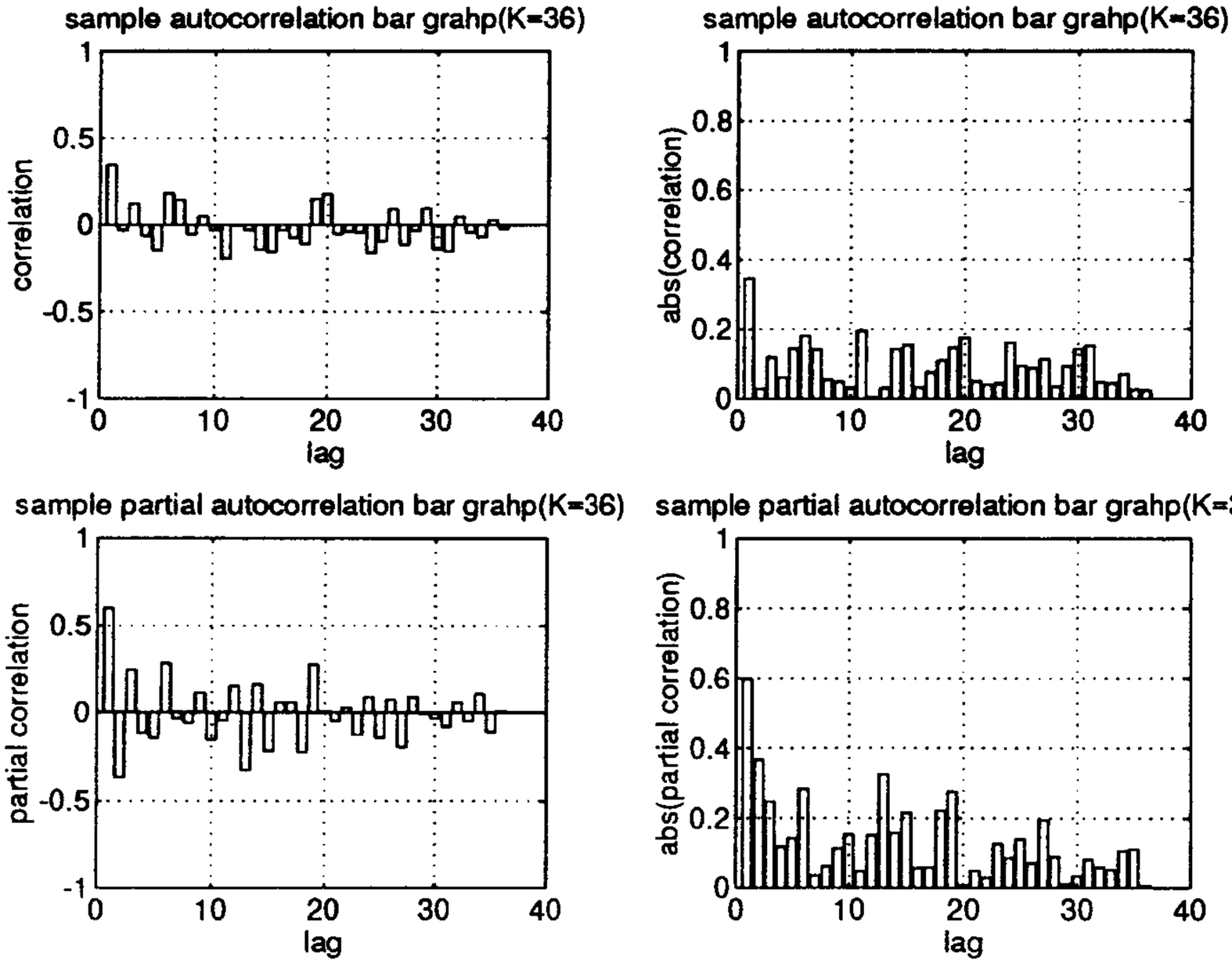


그림 2.7: Sample 와 sample partial autocorrelation

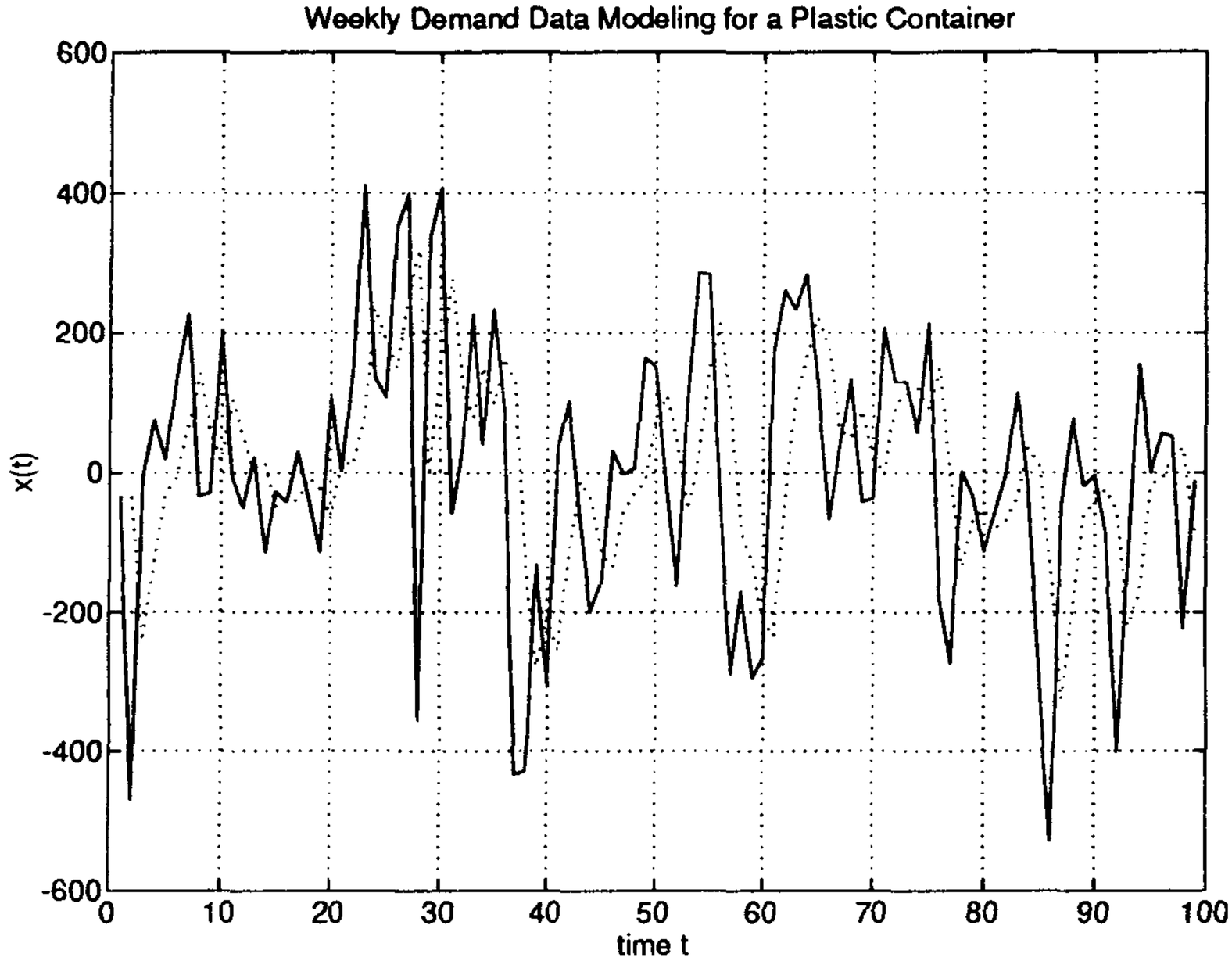


그림 2.8: 시계열 예측 모델 fitting 결과

사용된다. 이 공간안에서 국부적인 선형 모델은 embedded된 데이타를 상위 몇개의 singular value decomposition 모드로 투영함으로서 만들어 지게 되는데, 이 모델의 차원은 attractor의 차원 또는 그보다 작게 된다. 이렇게 만들어진 저차원 선형모델이 미래의 time series를 예측하는 데 사용되게 된다[25].

2.6.1 Embedology

Deterministic한 dynamic 시스템의 state는 이 시스템의 미래의 전개 상황을 완벽하게 결정하는데 필요한 정보이다. 한가지 기억할만한 유용한 paradigm은 n 개의 변수를 갖는 n 개의 ordinary differential equation으로 구성된 시스템을 들 수 있다. 이경우 smoothness에 대한 최소한의 조건이 만족된다면 존재 및 유일성(existence & uniqueness)의 성질이 만족되게 된다. 다시말하면, R^n 공간상의 한 점 $\mathbf{a} = (x_1, ..., x_n)$, 을 초기값으로 하는 unique trajectory가 한 개 존재한다는 것이다.

Time series는 시간이 경과함에 따라 관측가능한 측정치라 여겨지는 숫자들의 list 이다. 따라서 dynamic system 이며 time series를 생성하는 관측치는 내재되 있는 시스템의 state의 함수라 볼 수 있다. 만약에 이 시스템이 일정시간 경과후에 동일한 state로 돌아오게 되면, 관측치는 동일한 값을 생성하게 될 것이다.

이것이 이 절에서 설명하는 방법의 기초이다. 자세히 설명하기 위해 다음과 같은 notation 을 도입하기로 하자.

$\mathbf{a}$: 시스템의 현재 state

$h(\mathbf{a})$: 현재시간에서의 관측치 h(a).

그러면 가정에의해 현재 상태 $\mathbf{a}$는 t 시간후에 상태, $F_t(\mathbf{a})$,를 생성할 때 필요한 모든 정보를 가지고 있다. 그러면 예측 문제란 $\mathbf{a}$ 가 주어졌을 때,

$$P_t(\mathbf{a}) = h(F_t(\mathbf{a})) \tag{2.34}$$

를 찾는 문제 이다. 여기서 P_t는 예측 함수 이다. 이와 같은 관점에서 보았을 때, 예측 문제는 표현의 문제(representation problem)와 function approximation

문제로 나눌 수 있게 된다. 표현의 문제의 경우는 이론적인 state a를 대치할 b를 구하기 위해 주어진 데이타를 이용하게 된다. 한가지 해결책이 delay coordinate embedding 방법으로 Euclidean space R^m 상에서 각 state a 는 유일한 b로 mapping 된다. 두번째 문제, function approximation 문제는 이용가능한 데이타로 부터 효율적인 근사 함수 $P_l : R^m \rightarrow R$ 을 찾는 문제로 앞으로 자세하게 기술되게 될 것이다.

2.6.2 Delay Coordinates

Delay coordinate를 재구성하는 방법은 time series 데이타로 부터 유도된 벡터를 사용하여 시스템의 dynamical state의 set을 재 생성하는 것을 말한다. A를 시스템의 compact한 유한 차원의 state 의 set 이라 하자. 예를 들어 이와 같은 set 은 평형점(equilibrium), periodic orbit, chaotic attractor 등으로 구성될 수 있다. 그리고 $g : A \rightarrow R$ 를 observation function 이라고 하고 r 을 0보다 큰 실수라하면 다음과 같은 m 차원 벡터를 구성할 수 있다.

$$b = \left[g(\mathbf{a}), g(F_{-r}(\mathbf{a}), ..., g(F_{-(m-1)r}(\mathbf{a})\right] \tag{2.35}$$

위의 벡터를 시스템의 관측치로 부터 시간 지연된 값으로 부터 나온 값들로 구성이 되어 있어서 delay coordinate vector라고 부른다.

벡터 b 는 measurement function g로 부터 생성된 등간격 time series 의 segment 이므로 다음과 같이 쓸 수 있다.

$$b = \left[x_t, x_{t-r}, ..., x_{t-(m-1)r}\right] \tag{2.36}$$

여기서 $x_t = g(\mathbf{a})$ 는 시간 t, state a 일때의 time series의 값이다.

2.6.3 Filtered Delay Coordinates

실세계에서 발생될 수 있는 noise 를 처리하기 위해 다음과 같이 정의된 filtered delay coordinate embedding 을 소개하기로 한다.

$$b = M\left[g(\mathbf{a}), g(F_{-r}(\mathbf{a}), ..., g(F_{-(w-1)r}(\mathbf{a})\right]^T \tag{2.37}$$

위 식에서 M은 rank m 인 $m \times w$ matrix 이다. 위의 식을 time series로 변환하면 아래와 같이 쓸수 있다.

$$b = M\left[x_t, x_{t-r}, ..., x_{t-(w-1)r}\right]^T \tag{2.38}$$

원래의 delay coordinate map과 마찬가지로 이 filtered 된 delay coordinate map 또한 state a 와 벡터 b 사이에 1:1 대응이 성립함을 알 수 있다. 특히 m 이 attractor A 의 box-counting 차원의 2 배 보다 크면 거의 모든 관측 함수 g는 1:1 대응하게 된다.

여기서는 $M = M_3M_2M_1$ 을 다음과 같이 정의 한 선형 operator를 사용하였다.

1. $M_1 = w$ 차 FFT (Fast Fourier Transform).

2. M_2 는 $m/2$ 보다 큰 주파수 성분을 제거한다.

3. M_3 는 나머지 $m/2$ 주파수 성분으로 m 차 IFFT 를 수행한다.

결론적으로 말하면 matrix M 를 곱하는 것이 time series 데이타에 대해 길이가 w 인 window 로 low-pass filtering 한 효과를 내는 것으로 low-pass embedding 이라 할 수 있다.

2.6.4 Prediction

이전 절에서 예측 문제의 반,즉, 계산가능한 양으로 현재의 상태를 표현하는 법을 해결했다. 이절에서는 입력으로서 현재의 state 를 가지고 미래의 time series 값을 예측하는 문제를 풀고자 한다. filtered delay coordinate embedding 을 통해서 내재된 dynamics 를 재구성할 수 있다. 즉, training set 로 부터 만들어진 state b로 부터, t 시간후의 값을 $X = P_t(b)$ 을 구할 수 가 있다.

Prediction Funcition 구하기

b^b 를 시스템의 상태를 나타내는 R^m space 상에 있는 벡터라고 가정하자. 여기서 우리가 원하는 것은 $P_t(b^*)$ 를 구하는 것이다. 여기서 P_t 는 t 시간 경과 후에 time series 값을 나타내는 함수 이다. 이 함수를 구하기 위해 먼저 R^m space 상에서 b^* 와 가장 가까운 k 개의 neighbors $b_1, ..., b_k$ 를 찾는다. 그러면 각 b_i 는 training set 으로부터 얻어졌으므로, state b_i 로 부터 t 시간 후의 값, $X_i = P_t(b_i)$ 을 구할 수 있다. 여기서 주의해야 할 것은 X_i 은 원래 time series 로 부터 얻어진 값이지 filtered된 값이 아니라는 것이다. 그러면 해야할 남은 과정은 이렇게 해서 얻어진 X_i 로 부터 구하고자 하는 $bf\ b$ 상태로 부터 t 시간 후의 값을 구하는 과정 만이 남게 된다. 지금까지 설명한 과정을 요약하면 다음과 같다.

1. 구해진 neighbors $\mathbf{b_1}, \ldots, \mathbf{b_1}$ 의 무게 중심 $\mathbf{c}$ 를 찾는다.

2. 주어진 dimension $l < m$,에 맞고 $\mathbf{c}$ 를 통과하는 최적의 저차원 linear space R^l 을 찾는다. 여기서 최적이란 말은 이웃들간의 거리가 최소인 것을 의미한다. Singular Value Decompostion (SVD) 을 사용하면 이 subplane R^l 을 쉽게 구할 수 있다. 좀더 자세히 설명하기위해 각 row 가 $\mathbf{b_1} - \mathbf{c}, ..., \mathbf{b_k} - \mathbf{c}$ 로 구성된 matrix A 를 정의 하자. 이 matrix A 를 SVD 하면 A 는 다음과 같이 분해된다.

$$A = USV^T \tag{2.39}$$

여기서 U 와 V 는 orthogonal matrix 이고S 는 diagonal 이며 증가하지 않으면서 음수가 아닌 값을 가지고 있다. l 개의 dominant right singular vector(matrix V 의 처음 l columns)가 우리가 원하는 R^l space를 span 하게 된다.

3. $\mathbf{b_1} - \mathbf{c}, ..., \mathbf{b_k} - \mathbf{c}$ 데이타 포인트를 이 R^l space로 투영하고 다음의 데이타 포인트에 가장 잘 맞는 상수와 선형의 형태로 구성된 affine Model $L : R^l \rightarrow R$ 을 구성한다.

$$T(b_1 - c, X_1), \ldots, T(b_k - c, X_k)$$

여기서 $T : R^{(m)} \rightarrow R^l$ 는 R^l에 맞는 최적의 투영이다. 그러면 모델 L 은 다음의 형태를 가지게 될 것이다.

$$L(x) = \mathbf{a} * \mathbf{x} + d \tag{2.40}$$

여기서 $\mathbf{a}$ 는 l 차원 벡터이며 d 는 상수이다.

4. $\mathbf{b}^* - \mathbf{c}$를 R^l space에 투영하고 모델을 구한다. 그러면 결과는 $L(T(b^* - c) = X^*$ 이며 우리가 원하는 예측치가 된다.

Discussion

예측 함수 P_t 를 추정하는데 있어서 몇가지 해결되지 않은 문제들이 있다. 실제 문제에 당면했을 경우에 최적의 조건은 구하기 힘이 들며, 예측치는 예측 함수를 구하는 방법에 따라 매우 민감하게 마련이다. 예측 함수를 구하는 방법에 있어서 고려할 수 있는 사항은 다음과 같은 것들이 있다.

Weighted Regression

일단 선형 mapping $L : R^l \rightarrow R$ 선택되고 neighbor의 숫자가 결정되면, 어떻게 주어진 data를 모델에 filtting 하느냐 하는 선택의 문제가 남게 된다. 이 경우 weighted regression 방법을 이용하면 예측 성능이 향상되게 됨을 알 수 있다. 다시 말하면 neighbor $\mathbf{b_i}$에 $(1 - (1/2)d_i^2)^3$ 을 가중하게 하는 방법이다. 여기서

$$d_i = \frac{dist(\mathbf{b_i}, \mathbf{b}^*)}{dist(\mathbf{b_k}, \mathbf{b}^*)} \tag{2.41}$$

이고 $\mathbf{b_k}$ 는 $\mathbf{b}^*$ 로 부터 가장 멀리 떨어진 neighbor를 말한다. 비록 이 가중치 주는 방법이 최적이라고 할 수는 없지만, 동일한 값을 주는 $d_i = 1$ 의 경우에 비해

향상된 결과를 얻을 수가 있다.

직접적인 방법과 순환적인 방법 (Direct vs Iterated Prediction)

우선 다음과 같은 time series 가 주어졌고 그다음을 예측하고자 한다고 가정하자.

$$x_1, \ldots, x_N$$

그러면 위에서 기술된 방법으로 one time unit 예측치 $\hat{x}_{N+1}$을 구할 수 있다. 그 다음 예측치 $\hat{x}_{N+2}$ 를 구하기 위해 두 가지 방법이 있을 수 있다. 그 첫번째가 직접적인 방법(direct prediction) 으로 주어진 $x_1, ..., x_N$ 로부터 two time unit 앞을 예측하는 방법이다. 다음으로는 이와는 반대로 주어진 $x_1, \ldots, x_N$와 $\hat{x}_{N+1}$을 이용하여 one time unit 예측치를 구하는 방법이 있을 수 있다.

어떤 방법이 나은 지는 충분한 연구 및 토의가 있어야 하겠지만 이들 방법은 각각 다음과 같은 약점이 존재하게 된다. 우선 직접적인 방법의 경우에는 현재 시점보다 상당히 먼 미래를 예측할 경우에 예측치에 대한 신뢰도는 매우 회의적이 되며, 순환적인 방법의 경우에는 예측된 값을 미래의 예측에 사용하므로 사용된 예측치가 이미 좋지 않은 예측치가 될 수도 있다는 위험부담을 감수해야만 한다. 비록 실세계에 적용할 때에 이상적인 조건은 구현하기도 힘이 들지만, 순환적인 방법이 좀더 우수한 성능을 보인다고 보고도 되었으나 본 연구에서는 한 방법만 사용하는 방법을 배제하였다. 즉, 두가지 방법으로 산출된 예측치의 평균을 취하는 방법을 사용하였으며, 이로 인해 예측치의 분산에 대한 최소화를 도모하였다.

Algorithm

이절에서는 이전 절에서 지금까지 설명해온 것들을 기반으로 예측 algorithm을 설명하고자 한다. 먼저 고려해야 할 파라미터들은 다음과 같은 것들이 있다.

1. w : window 길이 (filter embedding 하기 위한 입력 window 의 길이)
2. m : low-pass filtering용 embedding dimension
3. l : model dimension (SVD로 결과를 이용하여 투영할 모델의 차원)

4. k : 1 차원 선형모델로 fitting 하기 위해 사용할 neighbor의 수

좀더 자세히 설명하기 위해 우선 주어진 $x_1, \ldots, x_N$로부터 예측치 $x_{N+1}, x_{N+2}, x_{N+3}, \ldots$ 을 구하는 문제를 생각해 보자. 그러면 예측치 $x_i, i > N$ 를 계산하기 위해서, 수 많은 예측치가 존재한다는 사실을 이용할 수가 있다. 다시 말하면, $\mathbf{b}_j, j < i$가 시간이 j 일 때에 filtered delay coodinate vector 라고 하면, $P_{i-j}(\mathbf{b_j})$는 x_i의 예측치가 된다.

이러한 사실을 이용하여 예측치 $x_i, i > N$는 아래의 식 2.42과 같을 이용하여 구할 수 있다.

$$x_i = \frac{1}{w} \sum_{j=i-w}^{i-1} P_{i-j}(\mathbf{b_j}) \qquad \text{-(2.42)}$$

위의 식 2.42은 과거 w 시간전의 예측치를 평균한 것임을 알 수 있고 다름아닌 직접적인 방법과 순차적인 방법의 혼합임을 알수 있다.

2.6.5 실험 결과

이 장에서는 전 장에서 설명한 알고리즘을 이용하여 실제로 예측 문제에 적용하고자 한다. 실험에 사용한 파라미터는 다음과 같이 설정되었다.

1. $w = 32$: window 길이
2. $m = 32$: lowpass filtering 용 embedding dimension
3. $l = 1$: model dimension
4. $k = 4$: 1 차원 선형모델로 fitting 하기 위해 사용할 neighbor 의 수

실험에 사용한 데이타로는 SantaFe Competition에 사용한 Laser data set을 사용하였다(그림 3.6). 1000개의 훈련용 데이타로 부터 400개의 미래의 데이타를 예측하는 것으로 식 2.42에서 w를 1로 사용했을 경우, 다음의 정규화된 error 계산식 2.43에 의해서 2273.3의 성능을 얻었으며 실제 값과 예측치의 값은 그림 2.9에 나타나 있다.

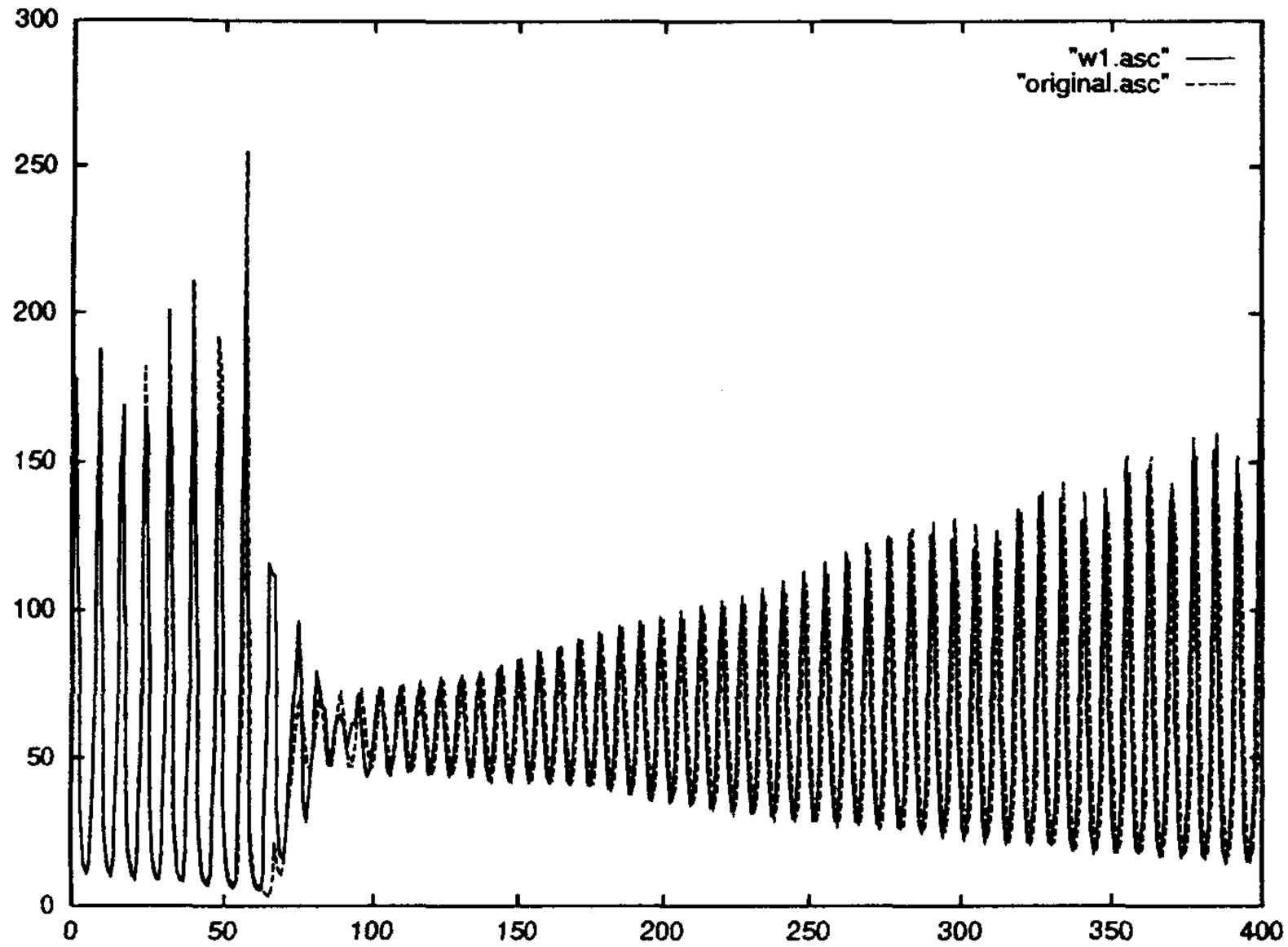


그림 2.9: $w = 1$일 경우의 실제값과 예측치

그림 2.10: $w = 2$일 경우의 실제값과 예측치

$$NMSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{2.43}$$

다음으로 식 2.42에서 $w = 2$ 로 수정하여 미래의 값을 예측하기 위해 과거의 2 time unit prediction 값을 이용하였을 경우에는 error 값이 613.7187로 줄었으며 그림 2.10에 나타난 것과 같이 더욱 향상된 결과를 얻을 수 있게 되었다.

여 백

3 장

신경망 예측 모델

신경망은 상호간의 연결강도(weight)를 갖는 여러개의 뉴런들로 이루어진 병렬분산처리 시스템이다. 신경망은 여러개의 뉴런으로부터 들어오는 입력신호들을 수학적인 식에의해 계산되는 과정을 거친후 하나의 출력으로 내보낸다. 이러한 과정은 생물학적 신경망의 수학적 모델이라고 할 수 있다. 신경망은 병렬처리를 함으로 처리속도가 빠르다. 몇개의 뉴런의 고장에 어느 정도 저항성을 갖고 있으며, 훼손된 입력에 대해서 근접한 해를 제공한다.

예측 문제에 이용되는 신경망 패러다임인 feed-forward 신경망 모델이 있다. Feed-forward 신경망은 뉴런(단순처리 소자)들이 정렬된 층(layer)에 배치되어 구성된다. 3층으로 구성되는 신경망을 $I \times H \times O$으로 표시하자. I는 입력층에 속한 뉴런의 수, H는 중간층에 속한 뉴런의 수, O는 출력층에 속한 뉴런의 수이다. 그림 3.1은 $3 \times 4 \times 3$ 신경망 모델의 한 예이다. 입력층에 속한 뉴런은 입력을 중간층에 제공한다. 중간층과 출력층에 속한 뉴런들은 하위층에 속한 뉴런의 출력값과 연결강도의 sum of products인 net_j을 계산후 전달함수를 통하여 출력값 o_j을 계산한다(여기서 j는 층내 뉴런 인덱스).

$$o_j = f(net_j) = f(\sum_i w_{ji} x_i) = \frac{1}{1 + \exp(-\alpha net_j)}$$

α는 전달함수의 기울기를 조정하는 상수값이다. Feed-forward 신경망 학습은 관측된 시계열 데이타를 가지고 진행된다. 학습패턴 집합 $\{(x_p, t_p) \mid p = 1, 2, \cdots, P\}$에서 선택된 (x_p, t_p)의 x_p를 신경망 입력으로하여 출력 o_p를 계산한 다음 target

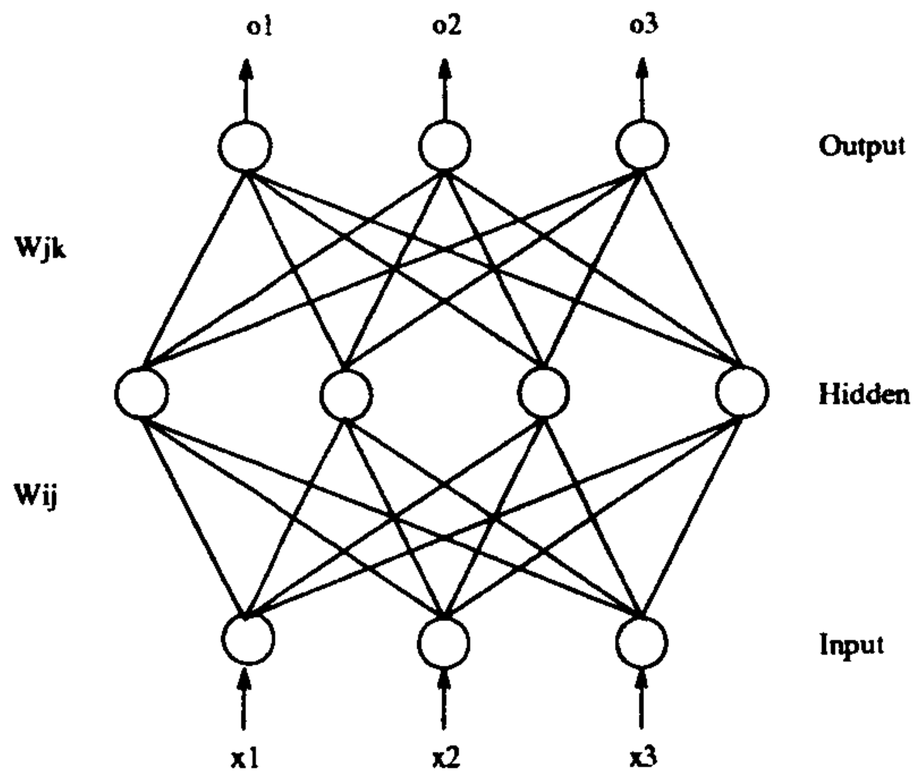


그림 3.1: $3 \times 4 \times 3$ feed-foreward 신경망 예

값 t_p의 error에 대한 미분치(derivative)를 역전파 시킨다. 본 연구를 통하여 적용된 feed-forward 신경망 학습 방법은 다음과 같다.

- Multilayer Perceptron(MLP) Learning
  - Backpropagation 학습방법 [22]
    - 입출력 데이타의 1차 미분제한을 갖는 학습방법
    - 입출력 데이타의 2차 미분제한을 갖는 학습방법
- Temporal Backpropagation Learning [24]
- Radial Basis Function Network [13, 16]

3.1 Backpropagation

MLP에서 입력 학습패턴에 대한 출력를 구한다음 target 패턴과의 차를(error 함수)를 weight 함수로 생각할수 있다. 이런 배경에서 weight에 대한 gradient가 존재한다. 그림 3.2은 weight W와 total error E간의 관계를 보여준다.

입력층 및 출력층 노드가 1인경우 주어진 학습패턴 p에 대하여 error E_p는 weight의 함수가 되며 target 패턴 t와 출력패턴 값 o에 대하여 E_p는 다음과 같

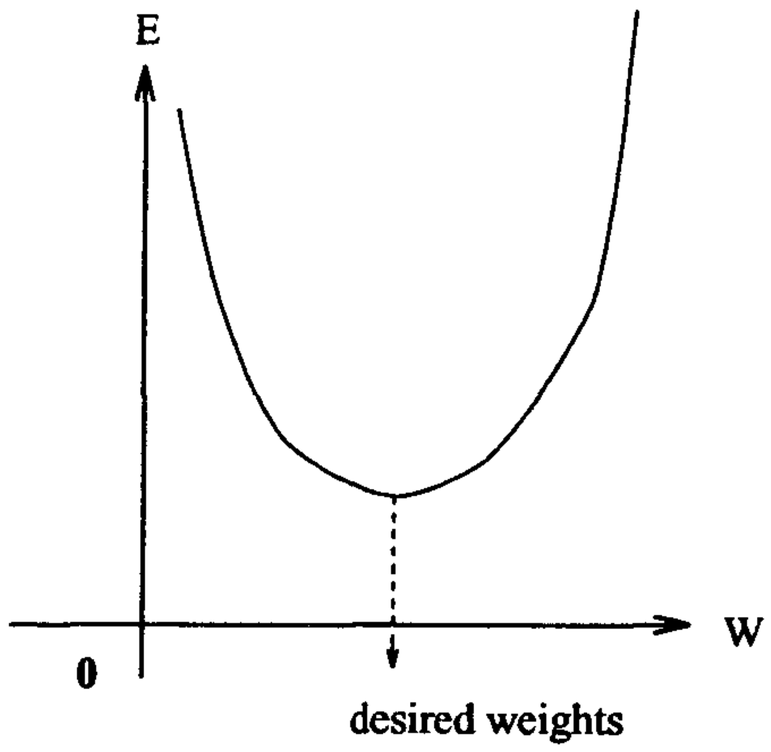


그림 3.2: Gradient descent - E vs w

다.

$$E_p = \frac{1}{2}(t-o)^2$$

그러므로 total error는

$$E = \sum_p E_p.$$

다음은 $n-$component 입력패턴 $I = [I^1 \;\; I^2 \;\; \cdots \;\; I^m]$, $q-$component target 패턴 $T = [T^1 \;\; T^2 \;\; \cdots \;\; T^m]$에 대한 MLP 학습방법을 보여준다[22]. 이 학습 알고리즘은 backpropagation 학습으로 알려져 있으며 (절 3.2)와 (절 3.3)에서 논의될 우수한 예측 성능을 갖는 알고리즘 개발은 backpropagation 학습 알고리즘으로부터 확장된 것이다.

Gradient descent for linear system

신경망이 입력층과 출력층으로 구성되었고 출력층 뉴런의 출력함수가 linear transfer 함수를 사용한다고 가정하자(그림 3.3). 입력에 대한 target 패턴 T^p의 error는

$$E_p = \frac{1}{2}\sum_{j=1}^{q}(T_j^p - o_j)^2, \quad p = 1, 2, \cdots, m$$

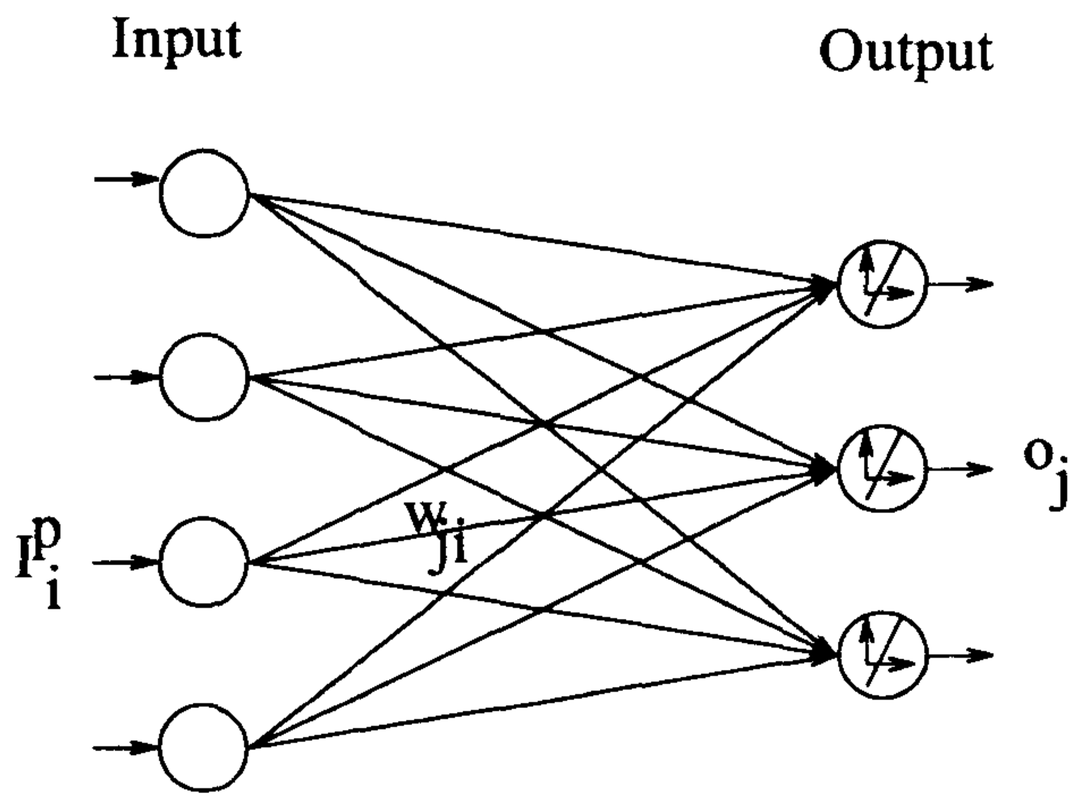


그림 3.3: 2 층으로 구성된 신경망 모델

여기서 o_j는 출력층 j번째 뉴런의 출력값이다. 입력층 뉴런i와 출력층 뉴런j을 연결하는 connection을 w_{ji}라 할때 error E_p의 변화에 대한 w_{ji}의 변화량은

$$\frac{\partial E_p}{\partial w_{ji}} = \sum_{k=1}^{q} \frac{\partial E_p}{\partial o_k} \frac{\partial o_k}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_j} \frac{\partial o_j}{\partial w_{ji}}$$

왜냐하면 출력층 j 뉴런은 입력층 i 뉴런과의 연결선만 남는다($k = j$). 첫번째 미분항은

$$\frac{\partial E_p}{\partial o_j} = -(T_j^p - o_j) \equiv -\delta_j.$$

두번째 미분항은

$$\begin{aligned} o_j &= \sum_i^n w_{ji} I_i^p \\ \frac{\partial o_j}{\partial w_{ji}} &= I_i^p. \end{aligned}$$

그러므로

$$\frac{\partial E_p}{\partial w_{ji}} = -\delta_j \times I_i^p.$$

I^p에대한 w_{ji}의 변화량은 $\Delta_p w_{ji} = -\delta_j \times I_i^p$가 된다. 그러므로 입력패턴 I^p에 대한 학습룰(learning rule)은

$$w_{ji} = w_{ji} + \eta \Delta_p w_{ji} \ (\eta\text{는 learning rate})$$

또한 각 학습패턴에 대한 total error로부터 학습률을 고려할수 있다. Total error에 대한 gradient는 각 패턴의 gradient의 합에 비례한다.

$$\frac{\partial E}{\partial w_{ji}} = \sum_p \frac{\partial E_p}{\partial w_{ji}}$$

$$w_{ji} = w_{ji} + \eta \Delta w_{ji}$$

Gradient Descent for semilinear transfer function

신경망이 2개 이상의 층을 갖는 다층으로 구성되고 각층 뉴런의 activation 함수가 semilinear(differentiable 이고 $f'(x) \geq 0$) 함수를 사용하면 각 층에서 뉴런의 출력값 o_j는

$$\begin{aligned} net_j &= \sum_i w_{ji} o_i \\ o_j &= f_j(net_j). \end{aligned}$$

Error E_p에 대한 w_{ji}의 gradient는

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}.$$

여기서, 뉴런의 층이 3 이상이고 각층의 뉴런이 linear transfer 함수를 사용하는 경우, $net_j = \sum_k w_{jk} o_k$ 임으로 두번째 미분항은

$$\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} (\sum_k w_{jk} o_k) = o_i.$$

첫번째 미분항을 δ_j로 놓으면

$$\delta_j = -\frac{\partial E_p}{\partial net_j}.$$

출력 뉴런이 $o_j = net_j$임으로

$$-\frac{\partial E_p}{\partial net_j} = \delta_j \times o_i$$

$$\Delta_p w_{ji} = \eta \delta_j o_i$$

Semilinear 함수를 사용하는 경우 δ_j와 o_j는

$$\begin{aligned}\delta_j &= -\frac{\partial E_p}{\partial net_j} = -\sum_k \frac{\partial E_p}{\partial o_k}\frac{\partial o_k}{\partial net_j} = -\frac{\partial E_p}{\partial o_j}\frac{\partial o_j}{\partial net_j} \\ &= -\frac{\partial E_p}{\partial o_j} f_j'(net_j)\end{aligned}$$

왜냐하면

$$\frac{\partial o_j}{\partial net_j} = f_j'(net_j) \quad (\text{from } o_j = f_j(net_j)).$$

어떤 뉴런 j의 출력은 같은 층내의 뉴런들의 출력과 독립관계에 있으므로 $k \neq j$인 k에 대하여는 모두 0이된다.

$\frac{\partial E_p}{\partial o_j}$를 계산은 출력층과 중간층(hidden layer)의 경우로 나누어 볼수 있다.

1. **출력층 뉴런**의 경우(그림 3.4)

$$\frac{\partial E_p}{\partial o_j} = -(T_j^p - o_j)$$

그러므로 뉴런 j의 gradient는

$$\delta_j = (T_j^p - o_j) f_j'(net_j)$$

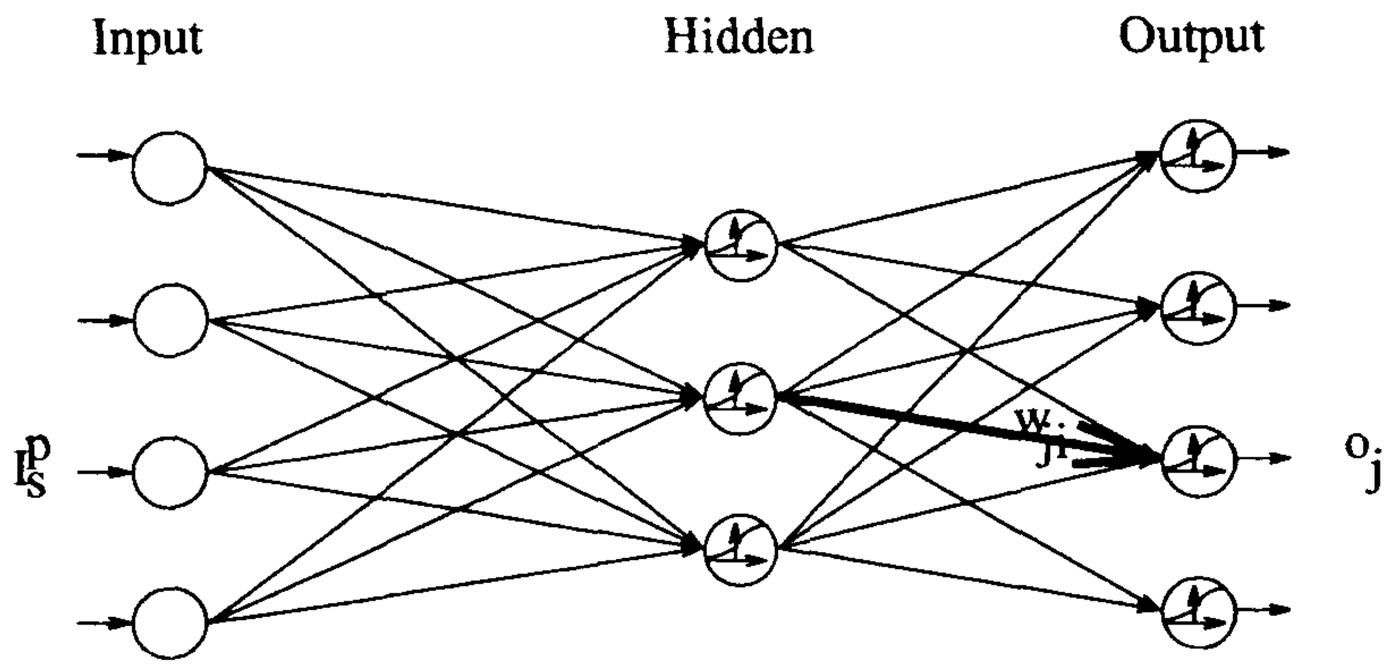


그림 3.4: MLP의 출력층

2. 중간층 뉴런의 경우(그림 3.5)

$$\begin{aligned}
\frac{\partial E_p}{\partial o_j} &= \sum_k \frac{\partial E_p}{\partial net_k}\frac{\partial net_k}{\partial o_j} \\
&= \sum_k \frac{\partial E_p}{\partial net_k}\frac{\partial}{\partial o_j}(\sum_i w_{ki}o_i) \\
&= \sum_k \frac{\partial E_p}{\partial net_k} w_{kj} \\
&= -\sum_k \delta_k w_{kj}
\end{aligned}$$

$\frac{\partial E_p}{\partial net_k}$은 중간층 뉴런 j에서 다음 상위층 뉴런 k의 weight 변화량이 된다. 그러므로 뉴런 j의 gradient는

$$\delta_j = f_j'(net_j)\sum_k \delta_k w_{kj}$$

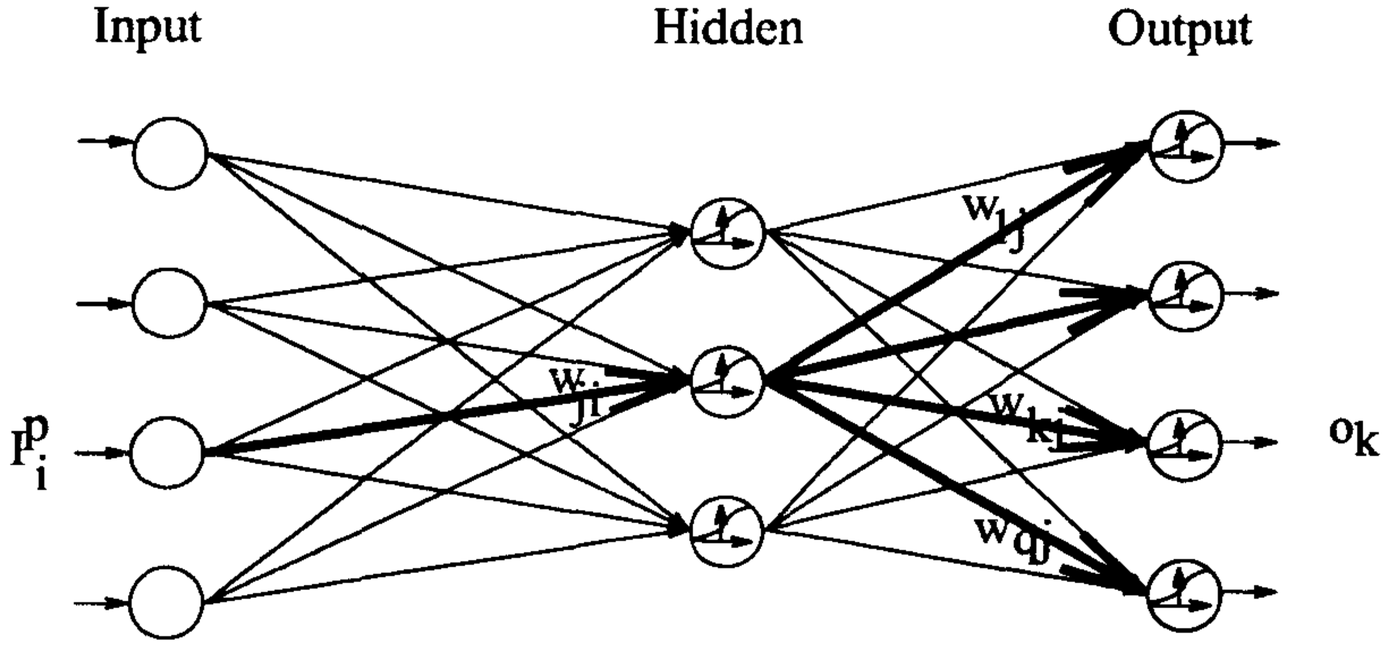


그림 3.5: MLP의 중간층

Transfer 함수를 다음 logistic 함수를 사용하는 경우 학습률은 다음과 같다.

$$\begin{aligned}
o_j &= \frac{1}{1+\exp[-(\sum_i w_{ji}o_i + \theta_j)]} \\
&= \frac{1}{1+\exp(-net_j)}
\end{aligned}$$

θ_j는 뉴런 j의 bias이다. $f_j'(net_j)$의 계산은

$$o_j = \frac{1}{1+\exp(-net_j)}$$

$$\begin{aligned}\frac{\partial o_j}{\partial net_j} &= -(1+\exp -net_j)^{-2}(-\exp(-net_j)) \\ &= \frac{1}{1+\exp -net_j}\frac{\exp(-net_j)}{1+\exp(-net_j)} \\ &= o_j(1-o_j)\end{aligned}$$

그러므로 학습률은

1. 출력층 뉴런의 경우

$$\begin{aligned}\delta_j &= (T_j^p - o_j)f_j^{'}(net_j) \\ &= (T_j^p - o_j)o_j(1-o_j)\end{aligned}$$

2. 중간층 뉴런의 경우

$$\begin{aligned}\delta_j &= f_j^{'}(net_j)\sum_k \delta_k w_{kj} \\ &= o_j(1-o_j)\sum \delta_k w_{kj}\end{aligned}$$

Transfer 함수가 logistic 함수인 경우

1. 출력층 뉴런의 값이 0.5에 가까운 뉴런의 weight adaptation이 크게 발생하므로 빠른 학습이 진행될수 있슴
2. logistic 함수에서 0 와 1의 뉴런의 출력값을($w_{ji} \approx \infty$ 또는 $-\infty$) 기대하기는 사실상 불가능 함으로 보통 (0.1, 0.9)의 뉴런의 출력값을 기대.

Momentum

MLP 학습의 학습속도를 높이기 위한 전략으로 t 시간에서 weight 변화량을 $t+1$에서 weight adaptation에 고려하는 방법이다. 또한 momentum 항은 잡음으로 인한 연결강도의 급작스런 변화를 억제하는 smoothing 기능을 준다. Momentum은 가능한 weight의 변화가 이전의 움직이던 방향을 고수하므로서 빠른 시간내에 최적의 weight에 도달하도록 하는 역활을 한다.

1. Rumelhart, Hinton 와 William 방법

$$w_{ji}(t+1) = w_{ji}(t) + \eta\Delta_p w_{ji}(t+1) + \alpha\Delta_p w_{ji}(t)$$

α는 momentum coefficient이다.

2. Exponential smoothing

$$w_{ji}(t+1) = w_{ji}(t) + \alpha\Delta_p w_{ji}(t+1) + (1-\alpha)\Delta_p w_{ji}(t)$$

α는 smoothing coefficient로서 1에 가까운 값을 갖게 되면 momentum 항의 역활이 사라진다. 만약 0에 가까운 값을 갖게 되면 학습을 오래 동안 한 방향으로 진행시키는 역활을 한다.

그러나 steepest descent 학습을 하는 경우 momentum 항의 역활은 학습패턴의 feature를 구분시키는데 도움을 줄수 있는 역활을 하게 된다.

MLP 신경망 모델을 예측시스템에 적용하는 3 단계는 다음과 같다[14].

Data Analysis: 예측문제를 분석하여 비선형 dynamics를 비교적 잘 내포하는 time delay 및 embedding dimension을 통계이론 혹은 경험적 방법으로 찾음

System Identification: 신경망 모델을 결정하고 학습을 진행한다. Data analysis로부터 신경망의 입출력 뉴런수가 결정되면 문제의 비선형 dynamics를 학습시킨다. 이 단계에서는 cross validation을 통하여 중간층 수 및 각 층의 뉴런 수와 학습률이 결정된다. 가장 좋은 비선형 dynamics 를 보여주는 모델이 선택된다.

On-line Prediction: 학습된 신경망 모델을 적용하여 예측을 진행한다. 필요하다면 예측치에 대한 error 보정 방법을 고려한다.

응용시스템 구성시 단계1은 좋은 학습패턴을 얻는데 매우 중요하며, 단계 2, 3은 신경망 모델 및 학습률 등을 변경시켜 학습이 반복 진행된다. 예측문제의 비선형

dynamics를 잘 보여주는 신경망 및 학습률이 short term 및 long term내 예측치를 얻는데 이용된다.

다음 시계열 데이타가을 $3 \times 5 \times 1$ 인 MLP에서 학습시킨다 하자.

$$x_{t+1}, x_{t+2}, x_{t+3}, \cdots, x_{t+T}, x_{t+T+1}, \cdots$$

설정된 신경망의 입력 뉴런수가 3, 출력 뉴런수가 1 이므로 위 시계열 데이타로부터 학습데이타 다음과 같이 4-component 학습 벡터가 된다.

$$(x_{t+1}, x_{t+2}, x_{t+3}, x_{t+4})$$
$$(x_{t+2}, x_{t+3}, x_{t+4}, x_{t+5})$$
$$\vdots$$
$$(x_{t+K}, x_{t+K+1}, x_{t+K+2}, x_{t+K+3})$$

학습시 설정된 벡터의 1, 2, 3 번째 항은 신경망 입력, 4 번째항은 target 값으로 하여 학습이 진행한다.

3.2 일차 미분 제한을 갖는 신경망의 개발

우수한 예측 성능을 가지는 신경망 모델을 개발하기 위해서는 보다 나은 일반화 성능을 가지는 신경망 모델이 필요하다. 신경망 모델의 일반화 성능은 학습해야 할 자료의 양과 해결해야 할 문제의 복잡도, 그리고 신경망 자체의 자유도 등에 크게 영향을 받는다. 일단 신경망의 은닉층 뉴런의 수가 정해지면 일반화 성능을 고려할 때 필요한 데이타의 최소량에 대한 이론적 연구 결과들이 발표되고 있지만, 실제 문제를 해결하기 위한 신경망 모델의 크기나 구조에 대한 설정 방법들은 아직 해결되지 않고 있다. 따라서, 보다 우수한 일반화 성능을 갖는 신경망을 개발하기 위해서는 아직도 데이타의 양과 문제의 복잡도, 그리고 신경망 모델의 크기 사이의 절충이 필요하다[5].

최근에 일반화 성능을 향상시키기 위한 여러가지 방법들이 개발되고 있다. 신경망 모델 자체의 복잡도를 줄여서 일반화 성능을 향상시키려는 시도로는 Pruning

algorithm[21], Synaptic weight elimination[23], Weight decay[19][2][4], Weight sharing[12][27][3]등의 방법들이 있지만 보다 나은 일반화 성능을 위해서 체계적인 설계 방법의 연구가 필요한 실정이다. 한편, 문제의 복잡도를 증가시켜서 일반화 성능을 향상시키려는 시도로는 Noise injection[11]이나 학습을 위한 성능 지수 함 수에 부가적인 제한(constraint) 함수를 이용하는 방법에 대한 연구들이 있다[18][17].

본 연구 과제에서 개발하고자 하는 방법은 학습 과정에서 소요되는 계산량의 큰 증가없이 보다 우수한 일반화 성능을 가지는 학습 알고리즘을 개발하고 이를 예측 모델을 개발할 때 흔히 비교 자료로 사용되는 SantaFe Competition Data Set A[25](Chaotic intensity pulsations of a NH3 laser)에 적용하여 개발한 알고리즘의 우수성을 보이고자 한다. 실제로 많은 예측 문제에서 출력값의 고주파 특성이 Overfitting을 일으키고, 따라서 좋지 않은 예측 결과를 주는 경우가 종종 있으므로 출력단에 저주파 여과기를 이용하는 경우가 있다. 본 연구에서 개발하고자 하는 신경망 모델은 이 Post-processing 과정을 신경망 모델내로 포함시켜 보다 간단한 구조로서 우수한 일반화 성능을 얻을 수 있도록 개발된다. 한편, 시계열 예측(Time series prediction)문제는 크게 Single step prediction과 Iterative prediction의 두가지 모드가 존재한다. Single step prediction은 n개의 과거와 현재 입력값을 이용하여 이후 한 Step의 미래치를 예측하고 난후, 실제로 얻은 값을 다시 입력으로 사용하는 예측 방법이고, Iterative prediction은 n개의 값으로 미래치를 하나씩 예측해 나갈때 예측된 결과를 다시 입력으로 사용하여 미래치를 예측해 나가는 것으로 초기의 예측 오차가 누적되어서 이후의 예측 결과에 큰 오차를 유발할 가능성이 큰 특징을 가지 고 있다[25]. 따라서, 초기 오차에 민감하지 않은 입출력 특성을 가지는 신경망 모델, 즉 입력의 변화분에 대해 강건한(Robust) 특성을 가지는 신경망 모델의 개발이 필요하다. 다음 절에서 입출력 데이타에 대해 강건한 특성을 갖는 신경망 모델의 개발에 관해 다루고, 본 절과 다음절에서는 신경망내에 입출력 데이타의 Low pass filter 특성을 포함한 새로운 신경망 모델의 개발에 관해 설명한다.

많은 지도 학습 알고리즘에서 학습을 위한 오차 함수는 출력 오차들의 제곱의 합으로 정의된다. 신경망은 오차 함수를 최소화하는 과정을 통해서 훈련 데이타로 정의되는 입출력 관계(mapping)를 학습하게 된다. 이제 이 오차 함수에 입출력 관계 함수의 일차 미분을 고려해보자. 입출력 관계의 일차 미분은 민감도(Sensitivity)와 관련이 있고, 이차 미분은 주파수 특성과 관련이 있다. 입력 데이타의 변화에 대한 강인성(robustness)과 fault-tolerance 성능을 향상시키기 위해서는 입력값에 대한 출력값의 민감도(sensitivity)를 줄여야 한다. 특히, 잡음이 많이 섞인 입력이 들어온다거나 제대로 동작하지 않는 시냅스 또는 뉴런이 존재할 때에는, 출력값의 낮은 민감도가 올바른 분류 (classification)와 연상(association)을 수행하는데 크게 유리하며 그리고, 높은 민감도를 갖는 입출력 관계에서 흔히 나타나는 오버피팅 (overfitting)을 피할 수 있다[18].

L개의 층을 갖는 전방향(feedforward) 신경망에 chain rule을 적용하면 다음과 같은 민감도에 관한 식을 얻을 수 있다.

$$\frac{\partial y_i}{\partial x_k} = \sum_{j_1,\cdots,j_{L-1}} W^{(L)}_{ij_{L-1}} W^{(L-1)}_{j_{L-1}j_{L-2}} \cdots W^{(1)}_{j_1 k} \dot{f}_L(\hat{y}_i) \dot{f}_{L-1}(\hat{h}^{L-1}_{j_{L-1}}) \cdots \dot{f}_1(\hat{h}^1_{j_1}) \quad (3.1)$$

여기서 x_k와 y_i는 각각 입력 벡터의 k번째 성분과 출력 벡터의 i번째 성분이고, $W^{(L)}_{j_L j_{L-1}}$ 은 L번째 층의 시냅스 연결을 나타낸다. 출력층이 L번 째 층이라면 i는 j_L 과 같게 된다. $\dot{f}_L(\cdot)$은 L번째 온닉층의 시그모이드(Sigmoid) 함수 $f_L(\cdot)$의 미분이고, $\hat{h}_{j_L} = \sum_{j_{L-1}}^{N_{L-1}} W^{(L)}_{j_L j_{L-1}} h^{L-1}_{j_{L-1}}$ 는 L번째 온닉층의 j_L 번째 뉴런의 시냅스 통과후 활성화 신호이다. 그리고 $h^L_{j_L} = f_L((\hat{h})^L_{j_L})$ 는 시냅스 통과전 활성화 신호이다. 설명의 편의를 위해서 L번째 층의 모든 뉴런이 모두 같은 sigmoid 함수 $f_L(\cdot)$ 을 갖는다고 가정하였다. 합은 모든 은닉층의 모든 뉴런들에 대해 행해진다. 향상된 fault-tolerance 성능을 위한 추가 오차 함수는 모든 i, k 인자에 대한 민감도의 제곱의 합으로 정의되는데, 이때 계산의 효율과 해석의 편리를 위해 간단한 하나의 항을 쓰도록 한다. 식 3.1로부터 온닉층의 활성화 신호를 포화(saturation)시켜서 민감도를 줄일 수 있다. 즉, 신경망 의 학습 과정에서 출력측 오차를 줄이는 것 뿐만 아니라, 동시에 $\dot{f}_L(\hat{h}_{jL})$ 도 아주 작은 값이 되도록 신경망의 연결 강도(Interconnection weight)를 구한다. 온닉층 뉴런들로 표현되는 hyperplane은

훈련 자료들로부터 멀어지게 되고, 시냅스 연결 강도 (synaptic weight)와 자료의 작은 변화는 큰 차이를 만들지 않으므로 강인한 분류(robust classification) 성능을 가질 수 있을 것으로 기대 된다.

일반적인 오차 함수대신에 다음 식 3.2의 새로운 오차 함수를 정의한다.

$$E = E_o + \gamma E_h = \frac{1}{M}\sum_{s=1}^{M} E_o^s + \frac{\gamma}{M}\sum_{s=1}^{M} E_h^s \tag{3.2}$$

여기서 $E_o^s = \frac{1}{2N_0}\sum_i^{N_0}(t_i^s - y_i^s)^2$ 는 s번째 훈련 패턴의 정규화(normalized)된 출력 오차이다. 새로 추가된 은닉층의 벌칙(penalty)항은 다음과 같다.

$$\begin{aligned} E_h^s &= \frac{1}{2}\prod_{l=1}^{L-1}\frac{2}{N_l}\left[\sum_{j_l=1}^{N_l}\dot{f}\left(\hat{h}_{j_l}^{s_1}\right)\right] \\ &= \frac{2^{L-2}}{N_1 N_2 \cdots N_{L-1}}\sum_{j_1,j_2,\ldots,j_{L-1}}\dot{f}\left(\hat{h}_{j_1}^{s1}\right)\dot{f}\left(\hat{h}_{j_2}^{s2}\right)\cdots\dot{f}\left(\hat{h}_{j_{L-1}}^{s(L-1)}\right) \end{aligned} \tag{3.3}$$

여기서, t_i^s 와 y_i^s 는 각각 s번째 저장된 패턴에 대한 목표출력값과 실제 출력값이다. 그리고 $\hat{h}_{j_1}^{s1}$ 는 1번째 은닉층의 j_l 번째 뉴런의 시냅스 통과후 값이다. 여기서 $M, N_0(=N_L), N_l$ 은 각각 저장된 패턴의 수, 출력 뉴런의 수, l번째 은닉층의 뉴런의 수이다. 두 개의 오차항은 이러한 수들로 정규화되어 있다. 만일 은닉층의 활성화 신호가 시그모이드 (Sigmoid) 함수의 선형 구간에 있으면 시냅스 연결을 통과한 값에 매우 민감해지고 은닉층 오차 E_h^s 가 커진다. 출력 오차 E_o 와 은닉층 오차 E_h 는 모두 비슷한 값들로 정규화되어 있음을 주목할 필요가 있다. 즉, 매우 작은 초기 연결 세기를 가질 때, hyperbolic tangent sigmoid 함수 에 대해 0.5 근처의 값으로 정규화된다. E_h 를 최소화시켜서 신경망을 더 좋은 robustness를 갖는 비선형 구간에서 동작하게 할 수 있다. γ 는 신경망의 학습 과정에서 은닉층 오차 E_h 의 출력 오차 E_o 에 대한 상대적 중요도이다. 이전의 논문[18]에서는 은닉층 벌칙을 상대적 중요도 γ_l 을 갖는 모든 은닉층에 대해 모두 더하였는데 γ_l 은 적절한 값 을 찾기가 어려웠다. 이제 식 3.2로부터 단 하나의 γ 만 찾으면 된다.

신경망의 학습에 흔히 사용되는 최대 오차 급경사 강하법 (steepest-descent error minimization) 알고리즘을 사용하여 신경망을 학습시킨다. 제일 마지막 층은

추가된 오차 항의 영향을 받지 않겠지만 다른 층에서는 전체 오차 함수의 각 weight 에 대한 편미분 항이 오차 함수에 추가된다. 그리고, 연결 강도(Interconnection weight)의 변경은 다음과 같다.

$$\Delta W^{(l)}_{j_l j_{l-1}} = -\eta_l \frac{\partial E}{\partial W^l_{j_l j_{l-1}}} = \eta_l h^{l-1}_{j j_{l-1}} \delta^l_{j_l} \tag{3.4}$$

여기서 $\delta^l_{j_1}$ 은 l번째 층에서의 시냅스 통과후 활성화 신호 $h^L_{j_l}$ 에 대한 전체 오차 함수의 민감도이며 다음과 같이 역전파(backpropagation)으로 계산할 수 있다.

$$\delta^l_{j_1} = -\frac{\partial E}{\partial \hat{h}^l_{j_l}} = \dot{f}\left(\hat{h}^l_{j_l}\right) \sum_{j_{l+1}=1}^{N_{l+1}} \delta^{l+1}_{j_1} W^{(l+1)}_{j_{l+1} j_l} + \gamma E_h \frac{\dot{f}\left(\hat{h}^l_{j_l}\right)}{\sum_{n=1}^{N_l} \dot{f}\left(\hat{h}^l_n\right)} \hat{h}^l_{j_l} \tag{3.5}$$

민감도 $\delta^L_{j_L}$ 는 일반적인 경우처럼 $\delta^L_{j_L} = \frac{1}{MN_L}(t_{j_L} - y_{j_L})\dot{f}(\hat{y}_{j_L})$ 로 출력층에서 정의된다. η_l 은 l번째 층의 학습 계수이고, 양극(bipolar) 시그모이드 함수 $f(x) = \frac{(1-e^{-x})}{(1+e^{-x})}$ 에 대해 $\dot{f} = \frac{(1-f^2)}{2}$ 와 $\ddot{f} = -f\dot{f}$ 가 성립한다. (윗첨자 s는 생략하였다.) 식 3.4와 식 3.5로부터 연결 강도 (Interconnection weight) 변경에 두 개의 성분, 즉 출력단 오차가 역전 파된 항과 입력값의 변화에 대한 출력값의 민감도를 줄이기 위한 은닉 뉴런의 벌칙항으로 구성되어 있음을 알 수 있다. 추가된 항은 $h^{l-1}_{j_{l-1}} h^l_{j_l}$ 에 비례하며, 개발한 알고리즘은 신경망의 학습 알고리즘으로 널리 사용되는 오차 역전파 알고리즘과 Hebbian 학습 규칙으로 이루어져 있다. 따라서, 시계열 예측 문제(Time series prediction problme)에서의 Iterative prediction 모드에서 예측 데이타의 차이에 의해 발생하는 예측 오차의 누적을 제안한 학습 알고리즘의 강건성(Robustness)에 의해 줄여줄 수 있을 것으로 기대된다.

제안한 방법의 타당성을 검토하기위해 시계열 예측 문제에서의 Bench mark 문제라고 할 수 있는 SantaFe Competition Data Set A를 이용하여 Single step prediction 모드와 Iterative prediction 모드의 경우 에 대해 시뮬레이션 한다[13]. 이 데이타는 NH3 레이저의 Chaotic intensity pulsations의 특성을 가지며, 이 중 1000개의 Sample을 신경망의 학습에 이용하고 이후의 100개의 Sample에 대해 예측 성능을 검증하는 문제이다. 사용한 신경망은 2층 구조로

25개의 입력과 40개의 온닉 뉴런, 그리고 1개의 출력으로 구성되어 있다.

그림 3.6온 실험에 사용한 Chaotic laser data의 모습이다. 그림 3.7는 보통의 오차 역전달 학습 알고리즘을 갖는 다층 구조 신경망 의 Single step prediction 의 결과이고, 그림 3.8온 제안한 방법에 의한 시뮬레이션 결과를 나타낸다. 여기서 점선은 예측해야 할 실제 값을 나타내고, Linespoints는 신경망 예측기의 결과를 나타낸다. 제안한 방법이 Single step prediction 인 경우 좀더 나은 예측 성능을 보여 줌을 알 수 있다. 그림 3.9는 보통의 역전달 학습 알고리즘을 갖는 신경망 모델의 Iterative prediction의 결과이고 그림 3.10는 제안한 방법의 Iterative prediction 모드의 예측 결과를 나타낸다. 역시 점선은 예측해야할 실제 값을 나타내고, Linespoints는 신경망 예측기의 결과를 나타낸다. Iterative prediction 모드에서 훨씬 더 정확한 예측 결과를 준다.

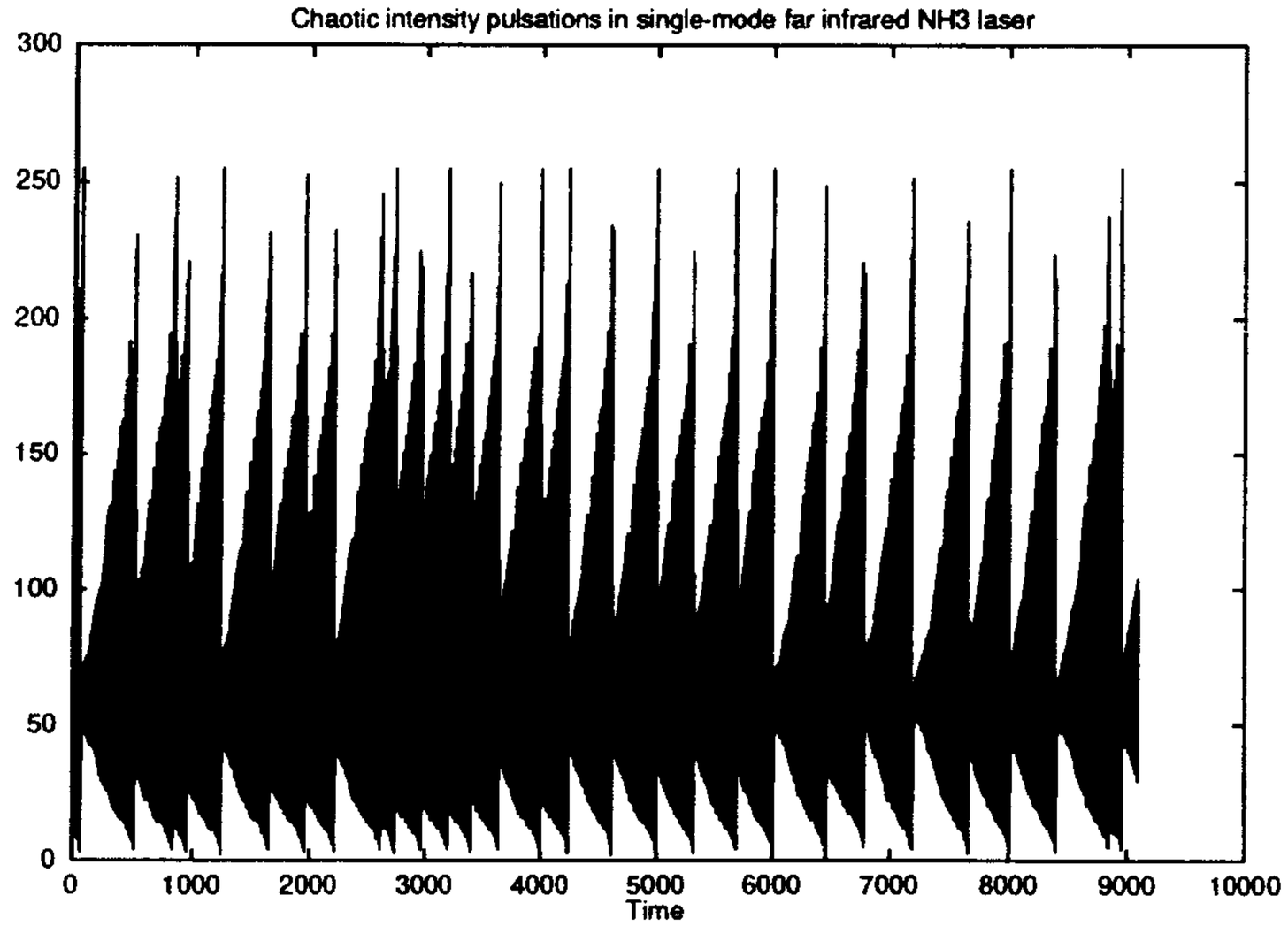


그림 3.6: Chaotic laser data의 시계열 모습

위의 여러 가지 방법들에 대한 NMSE(Normalized mean squared error) 는 다음 표 3.1에 주어진다.

실제로 시계열 예측 문제에서 Iterative prediction인 경우 하나의 신경망만으로 해결하는데에는 그 성능의 한계가 존재하며, 시계열 데이타의 특성에 따라 여러 개의 신경망을 이용하거나, 또는 별도의 다른 고려가 필요한 실정이다. SantaFe

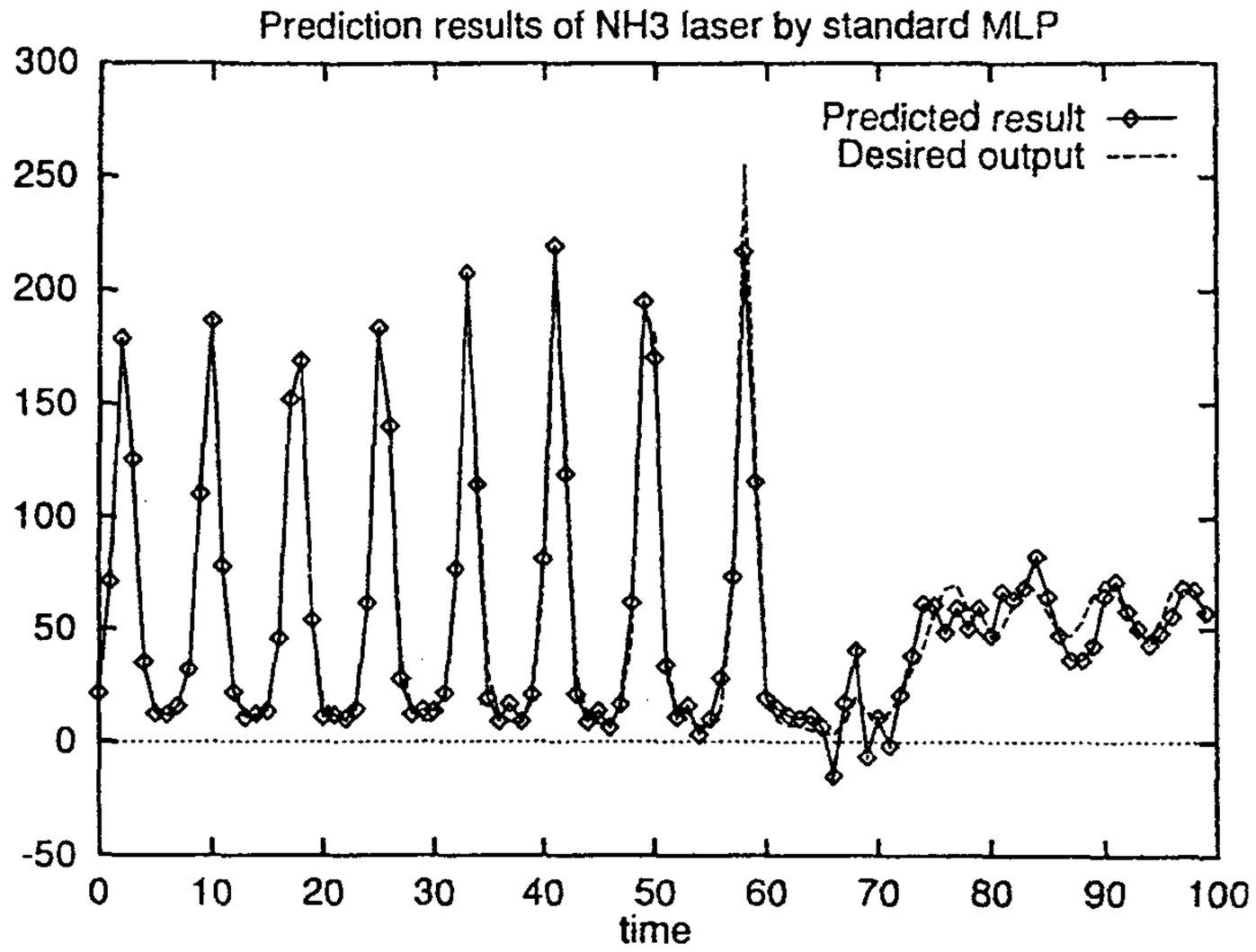


그림 3.7: 보통의 신경망에 의한 single step prediction 결과

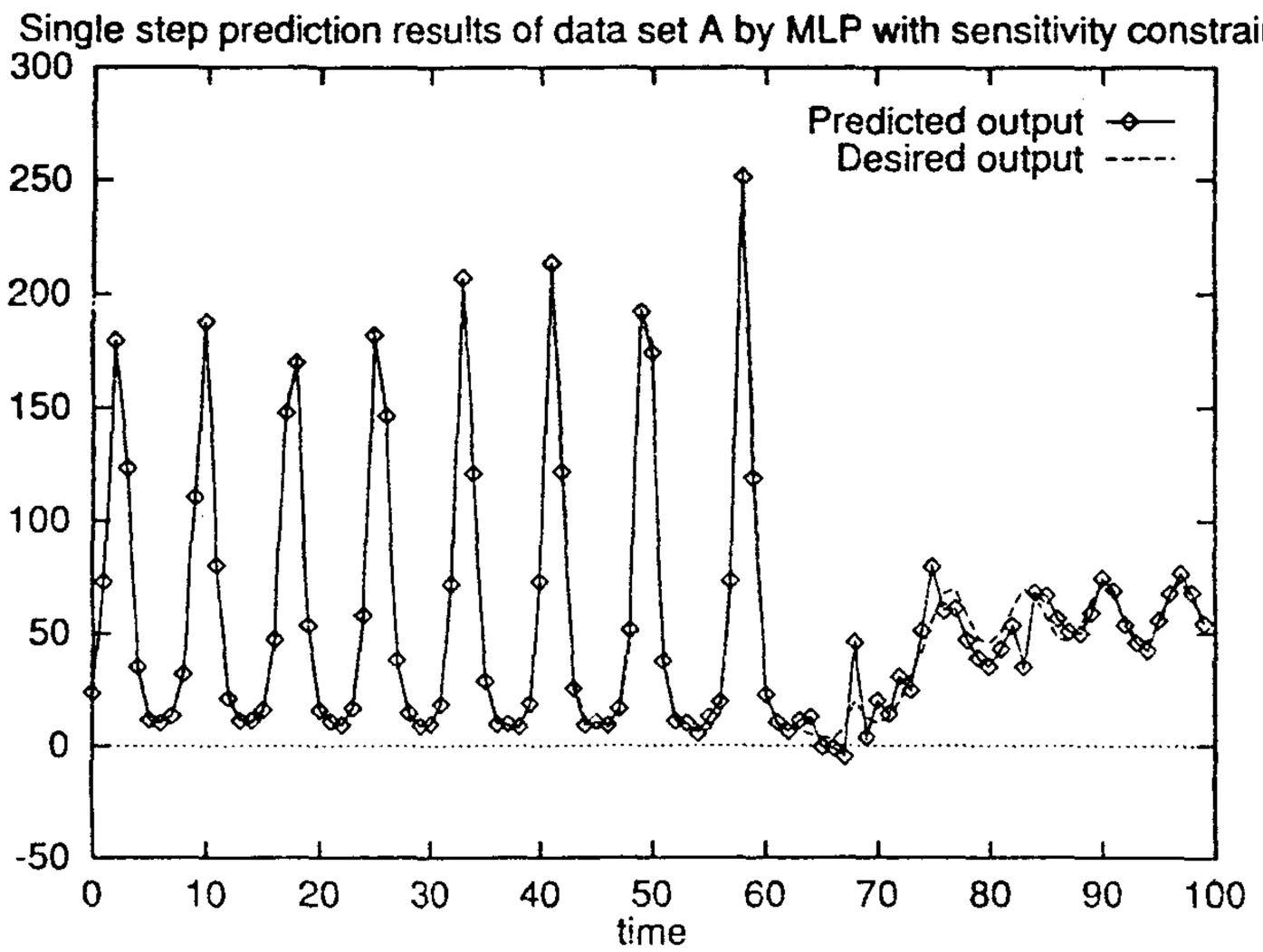


그림 3.8: 제안한 방법에 의한 Single step prediction 결과

예측모드	기존의 신경망	1차 미분제한을 갖는 신경망
Single step prediction	0.0365	0.0199
Iterative prediction	1.2599	0.6998

표 3.1: NMSE(Normalized mean squared error)

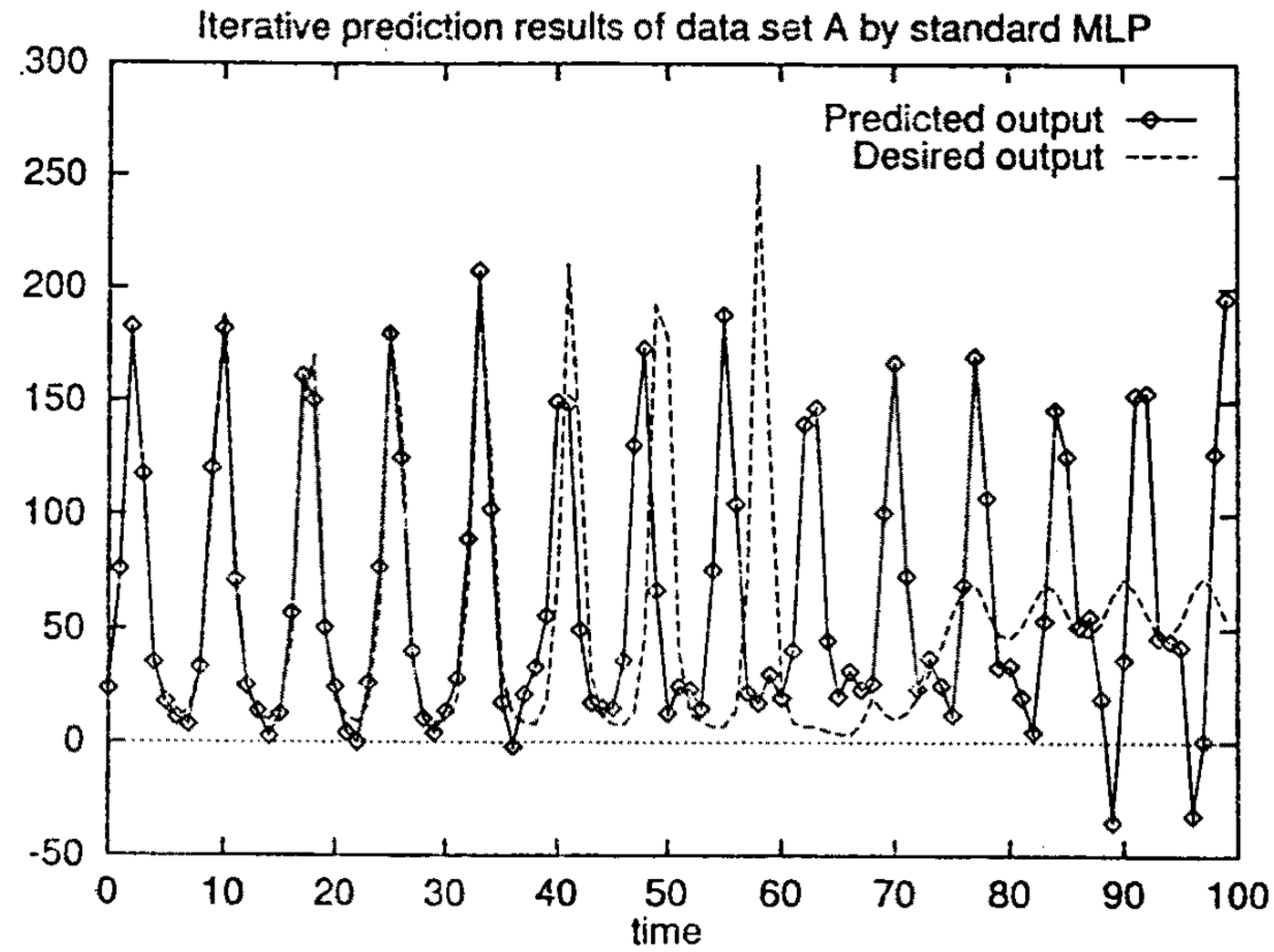


그림 3.9: 보통의 신경망에 의한 Iterative prediction 결과

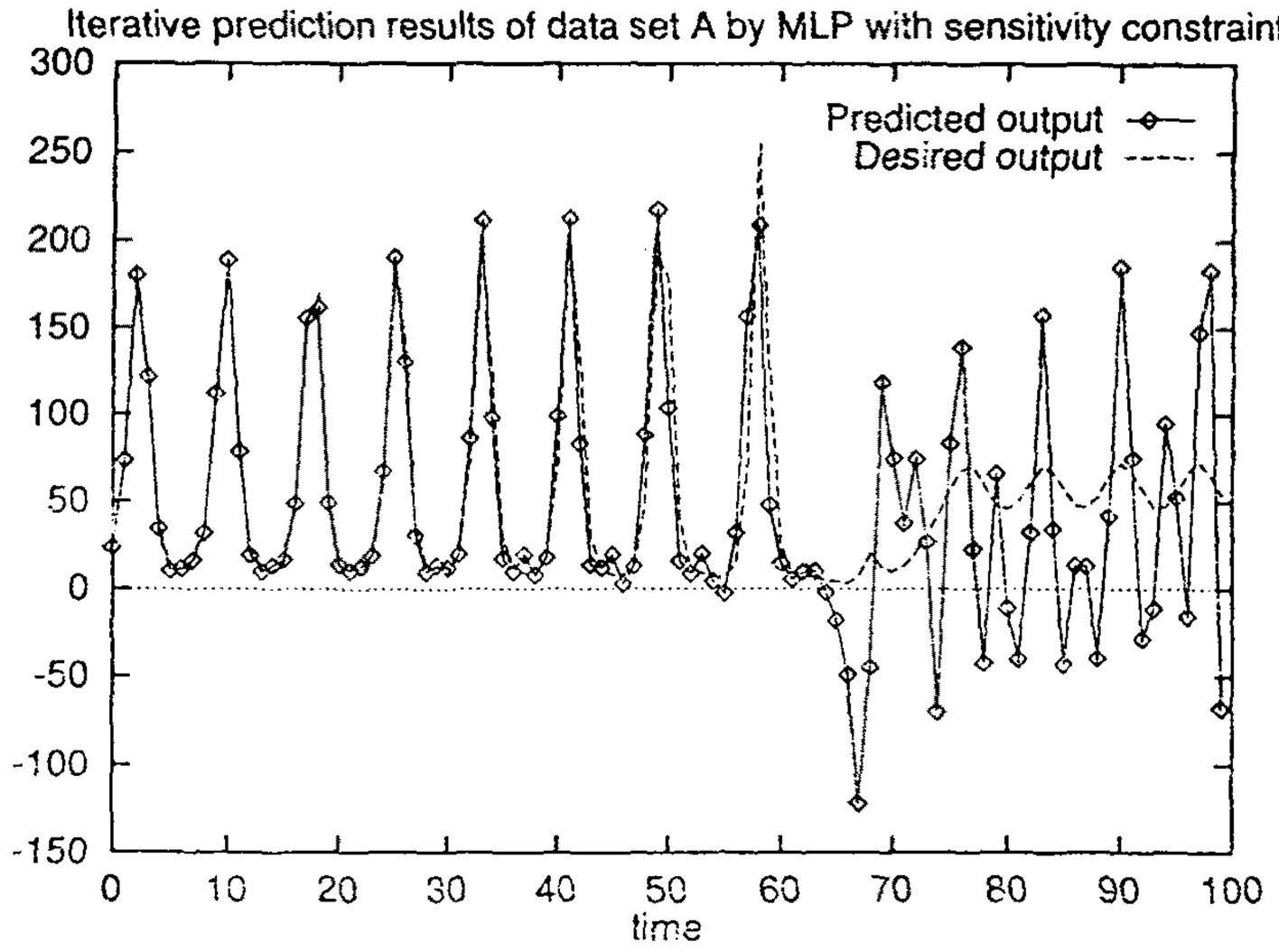


그림 3.10: 제안한 방법에 의한 Iterative prediction 결과

Competition Data Set A인 경우 시점 1000에서 시점 1100 사이의 데이타 중 Collapse가 일어나는 지점을 얼마나 잘 예측하는가가 중요하며, 이 부분의 예측 이후에는 다른 특별한 처리 방법이 필요하다. 예로, Weigend[26]의 논문에서는 Collapse가 일어난 후 학습 데이타에서의 Collapse발생 이후의 평균값을 이후의 데이타의 예측치로 이용했으며, Wan[26]은 이 부분의 발생이후 의 예측 데이타로 학습시의 같은 현상의 데이타의 예측치로 대신하는 방법들을 이용하고 있다. 이들 방법들 모두 인위적이며 체계화 되어 있지 못해 그 타당성에 의문이 남아 있으며, 향후 이 문제를 더욱 면밀히 고려해야 할 것으로 생각된다.

3.3 이차 미분 제한을 갖는 신경망의 개발

예측을 할 때, 일반적으로 출력값의 자세한 고주파 특성에는 관심이 없기 때문에 예측을 한 다음에 저주파 필터링(Low pass filtering)의 과정이 필요하다. 이와 같은 저주파 필터링(Low pass filtering)의 Post-processing의 과정을 신경망의 영역 내로 포함시키는 것이 바람직하며, 신경망 학습의 오차 함수를 적절히 정의하므로써 이러한 과정을 수행할 수 있다[17].

신호 $y(x)$의 주파수 제곱은 음의 이차 미분과 신호 자신과의 비가 된다. 즉, $-(d^2y/d^2x)/y$이다. 식 3.1에 chain rule을 적용하여 이차 미분을 다음과 같이 얻을 수 있다.

$$\frac{d^2y_i}{d^2x_k^s} = -\sum_j W_{ij}^{(2)} \dot{f}_j(\hat{h}_j)(W_{jk}^{(1)})^2 \tag{3.6}$$

여기서 논의를 간단하게 하기 위해 선형적인 출력을 갖고 하나의 은닉층으로 구성된 신경망을 가정하였다. 그리고 bipolar hyperbolic tangent sigmoid 함수에 대해 $\dot{f} = -ff$ 를 이용하였다. 여기서도 이차 미분에 대한 정확한 식을 사용하는 대신에, 식 3.6과 $y_i = \sum_j W_{ij}^{(2)} f_j(\hat{h}_j)$ 를 비교하고 더 간단한 형태를 채택하여 다음과 같은 저주파 필터 특성을 갖는 추가 오차 항을 정의하도록 하자.

$$E_h^s = \frac{1}{2N_h} \sum_j \dot{f}(\hat{h}_j^s) \sum_k (W_{jk}^{(1)})^2 = \frac{1}{N_h} \sum_j \dot{f}(\hat{h}_j^s) e_j \tag{3.7}$$

여기서 $e_j = \frac{1}{2}\sum_k (W_{jk}^{(1)})^2$는 첫번째 층의 synaptic weight energy로 정의 된다. $\dot{f}$와 e_j가 모두 양수임에 주목할 필요가 있다. 그리고, 출력 값의 차이에 의해서만 정의되는 오차 함수를 다음 식 3.8과 같이 새롭게 정의한다.

$$E = E_o + \gamma E_h = \frac{1}{2N_o}\sum_i^{N_o}(t_i - y_i)^2 + \gamma E_h \tag{3.8}$$

E_o 는 출력층의 오차를 의미하고, E_h 는 식 3.7에서 나타낸 부가적인 제한 함수로 신경망의 입출력 데이타의 이차 미분 제한에 의해 신경망이 저주파수 특성을 갖도록 하기 위한 온닉층 벌칙(Penalty)을 위한 함수이다. 편의를 위해 첨자 s는 생략하였다. 이제 식 3.8에 최대 오차 급경사 강하법(Steepest descent error minimization method)과 Chain rule을 적용하면, 다음 식 3.9와 같은 학습 알고리즘을 얻게 된다.

$$\begin{aligned} \Delta W_{jk}^{(1)} &= \eta_1 \dot{f}(\hat{h}_j)\left[x_k sum_i \delta_i W_{ij}^{(2)} + \gamma(x_k h_j e_j - W_{jk}^{(1)})\right] \\ W_{jk}^{1}(new) &= W_{jk}^{1}(old) + \Delta W_{jk}^{1} \end{aligned} \tag{3.9}$$

식 3.9에서 대괄호 안의 첫번째 항은 일반적인 역전파 오차항이며, 두번째항은 추가된 오차항으로 연결 강도 감소(Weight decay)와 Hebbian 학습 규칙으로 구성되어 있다.

식 3.8과 식 3.9는 신경망의 골격(Framework)내에 주파수 필터링의 수행을 포함시킬 수 있는 하나의 예를 보여주는 것이다. 보다 나은 주파수 응답을 얻기 위해 오차 함수를 적절히 설계해 줄 수 있을 것이다. 예를 들어, 추가 오차 항에 다음과 같은 비선형 지수 함수를 사용하여 훨씬 예리한 주파수 응답을 얻을 수 있다.

$$E_h^s = \exp\left[\frac{1}{2N_h}\sum_j \dot{f}(\hat{h}_j^s)\sum_k (W_{jk}^{(1)})^2 - \theta^2\right] = \exp\left(\frac{1}{N_h}\sum_j \dot{f}(\hat{h}_j^s)e_j - \theta^2\right) \tag{3.10}$$

여기서 바이어스 θ 는 cut-off 주파수를 조정하는 역할을 한다. 식 3.10에 근거

예측모드	$\gamma = 0.0$	$\gamma = 10^{-5}$	$\gamma = 10^{-1} \rightarrow 10^{-5}$	Weigend
Single step prediction	0.0365	0.0223	0.0172	0.0198
Iterative prediction	1.2599		0.8734	0.0960

표 3.2: 저주파수 특성을 갖는 신경망 모델의 예측 성능 비교

한 연결 강도(Interconnection weight) 변경은 식 3.9와 매우 유사하게 얻을 수 있다. 고주파 필터(high-pass filter)와 대역 필터 (band-pass filter)의 특성도 추가 오차 함수를 적절히 정의하여 얻을 수 있을 것이다.

제안한 방법의 타당성을 보이기 위해 SantaFe Competition Data Set A를 이용하여 예측 실험을 수행하였다. 사용한 신경망의 조건들은 이전절에서와 같으며, 역시 1000개의 Sample 데이타를 이용하여 신경망을 학습시키고, 이후 100개의 Sample 데이타들에 대해 예측 성능을 비교해 본다. Prediction은 전절에서와 마찬가지로 Single step prediction과 Iterative prediction의 두가지 모드로 수행하였으며, 각각의 NMSE(Normalized mean square error) 는 다음 표 3.2에 나타내어져 있으며, 각각의 경우에 대한 예측 실험의 결과들이 다음 그림들에 나타내어져 있다. 그림 3.11은 고정된 γ 를 갖는 경우의 Single step prediction의 결과이고, 그림 3.12은 신경망의 학습 과정중에 γ를 변화시켰을 때의 Single step prediction의 결과이다. 그림 3.13은 γ를 변화시켰을때 Iterative 모드로 예측 실험을 했을 때의 결과이다.

표 3.2와 위의 그림들에서 보는 바와 같이 Single step prediction인 경우는 저주파수 특성을 갖는 신경망 모델을 이용하는 경우가 가장 우수한 예측 결과를 주며, Iterative prediction인 경우는 입력 데이타의 변화에 대한 강건한 출력 특성을 갖는 신경망 모델이 좀 더 나은 예측 결과를 준다. 한편, 신경망의 학습 과정에서 최종 출력단의 오차에 대해 은닉층 오차에 대한 상대적인 비인 γ 를 고정시킨 경우보다 이 값을 학습 과정중에 변화시킨 것이 더 나은 예측 성능을 준다. 하지만, γ를 학습 과정 중에 적응적으로 변화시키는 체계적인 방법의 연구가 필요하다.

본 장의 3절과 4절에서 제안한 학습 알고리즘의 특성과 성능을 간단히 살펴보면 다음과 같다. 우선 제안한 방법들의 장점이라고 할 수 있는 것은 학습 알고리즘의 간단함과 그 설계의 간편함이라고 할 수 있다. 즉, 식 3.5와 식 3.9에서 두

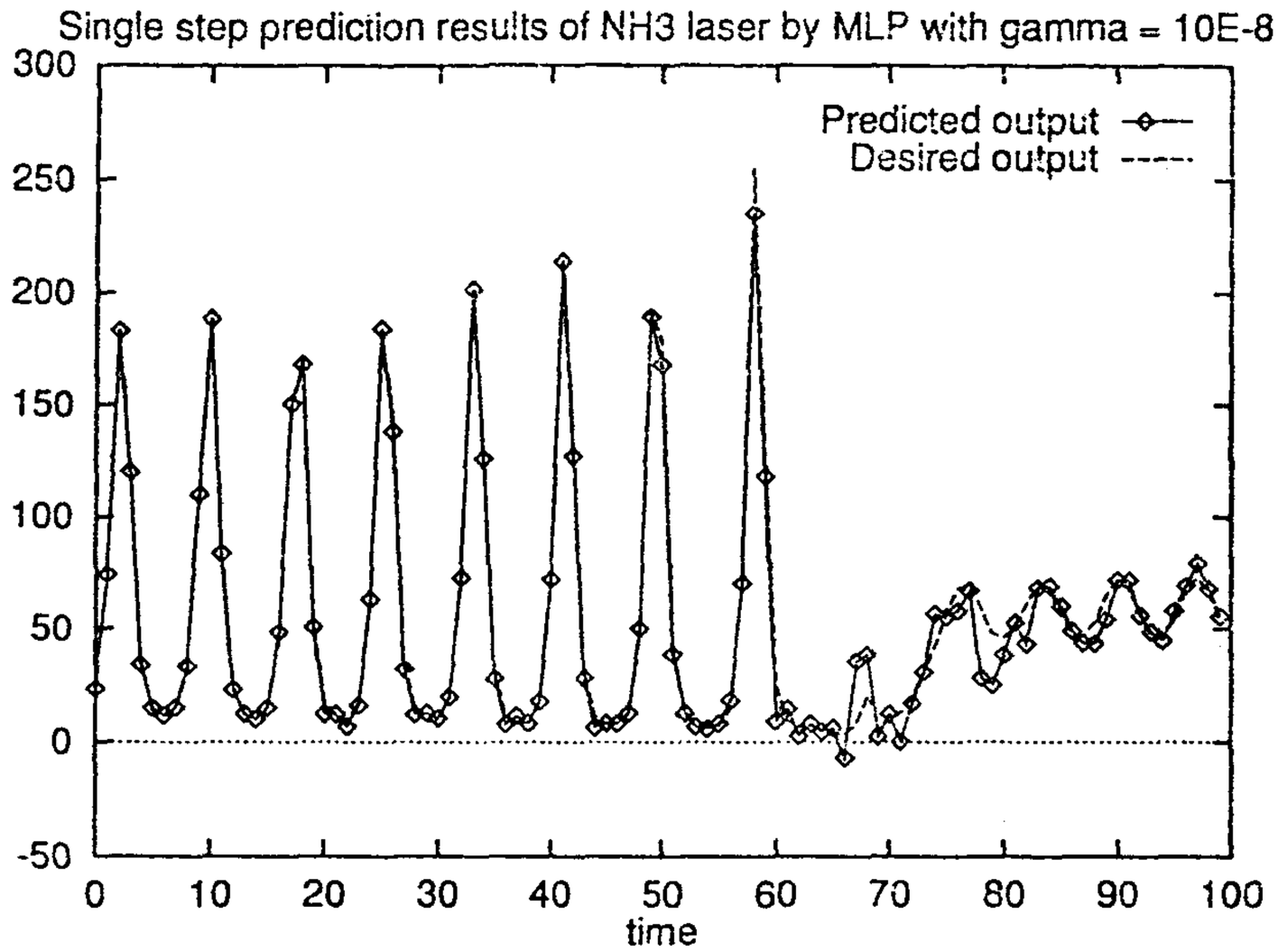


그림 3.11: $\gamma = 10^{-5}$ 인 경우의 single step prediction 결과

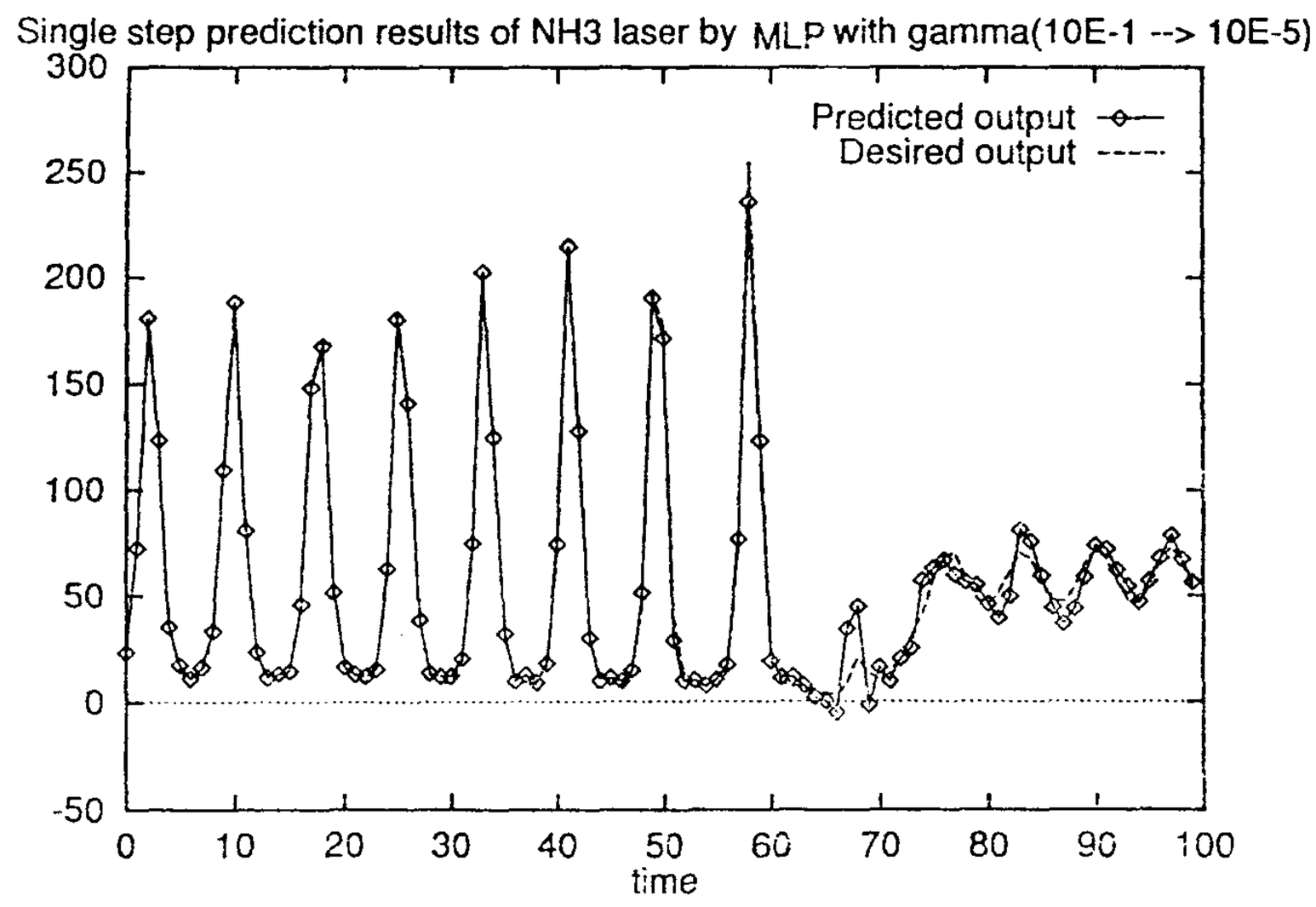


그림 3.12: $\gamma = 10^{-1} \rightarrow \gamma = 10^{-5}$ 일때의 single step prediction 결과

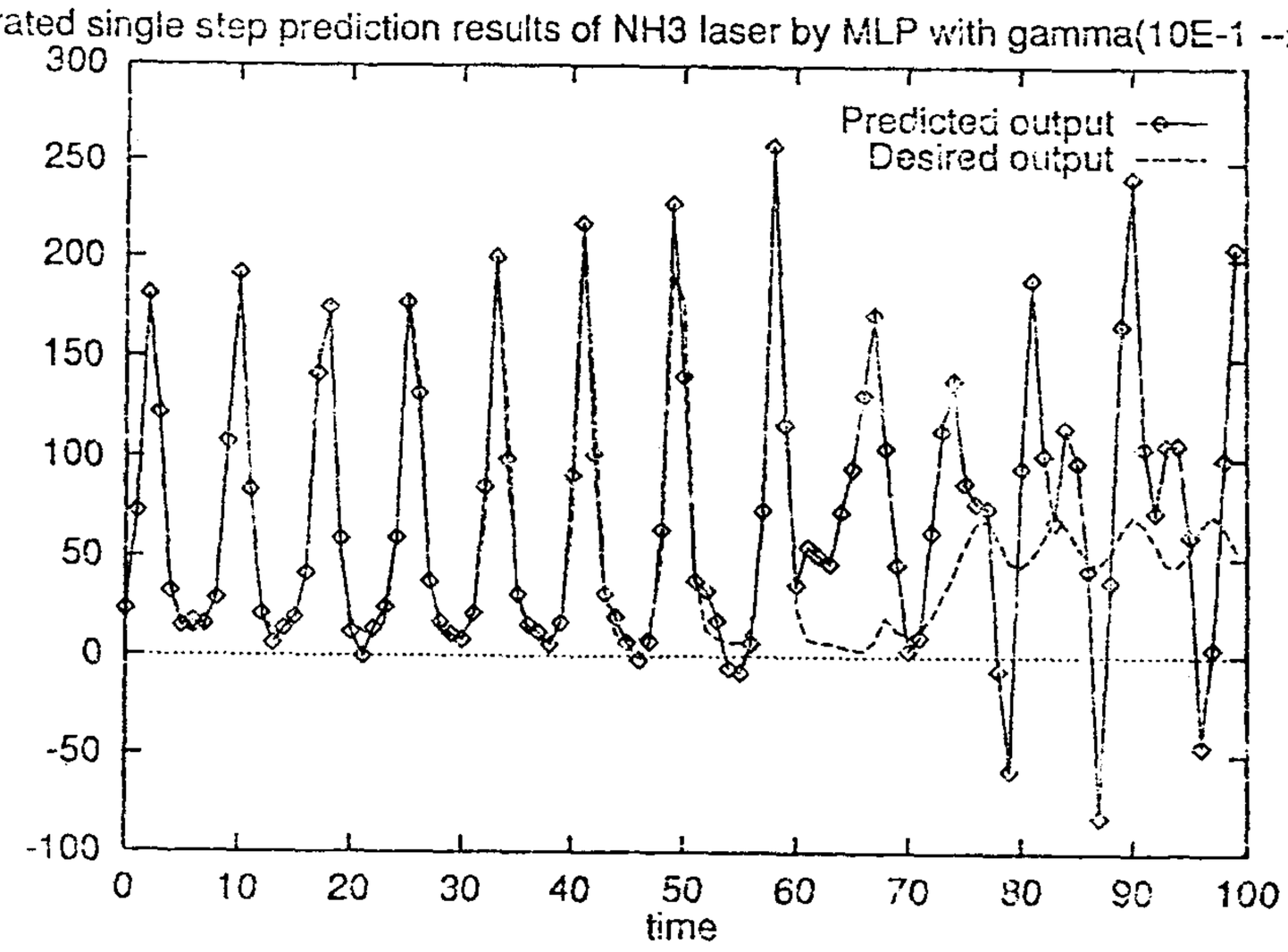


그림 3.13: $\gamma = 10^{-1} \rightarrow \gamma = 10^{-5}$ 일때의 iterative prediction 결과

번째 항은 일반적인 역전파 알고리즘을 수정한 것에 지나지 않는다. 따라서 추가적인 계산은 거의 무시할 정도이다. 위의 두가지 경우에서 Hebbian 학습과 관련된 항에는 $\dot{f}(\hat{h}_j)$가 곱해져서 훈련을 통해 0으로 수렴하고 시냅스 weight가 무한정 증가하거나 감소하는 것을 막아준다. 학습의 초기상태에서는 매우 작은 초기 랜덤 연결값(Weight value)을 이용하므로 초기 학습 과정에서는 $\hat{h}^l_{ji}$ 은 매우 작아서 Hebbian 항은 크게 영향을 끼치지 못한다. 첫번째 역전파 항에 의해 학습이 진행됨에 따라 은닉층의 활성화 신호가 커지고 Hebbian 항은 은닉층의 활성화 신호를 포화 구간(saturation region)에 위치하게 만든다. 또한 식 3.9에서처럼 weight decay와 경쟁한다.

하이브리드 알고리즘은 특정한 문제에 대해 충분한 복잡도를 갖는 다층 신경망을 위해 고안되었다. 하이브리드 알고리즘은 출력 오차뿐 아니라 은닉층의 오차까지 한꺼번에 최소화하려고 시도한다. 만일 이것이 불가능하다면, 출력 오차의 최소화가 더 높은 우선순위를 가지며 γ 의 값을 감소시켜서 은닉층의 요구조건을 어느 정도 완화시킨다. 실제로 처음에 γ 의 값은 크게 설정하고 학습도중에 네트워크의 복잡도가 충분히 크지 않은 것으로 밝혀지면 점점 줄여 나간다. 이러한 적응적(adaptive) 접근 방법을 가지고 하이브리드 학습 알고리즘은 하드웨어 리소

스(resource)와 기존의 네트워크 구조(architecture)를 최대한 이용하는 좋은 해답을 찾아낸다.

3.4 Temporal Backpropagaion

Wan은 신호처리의 이론인 FIR 선형 필터을 이용한 신경망 모델 및 학습 알고리즘을 보였다. 모델 구성은 적용 초기 MLP 모델 구조에 시간 정보를 embeding 시켜 모델을 확장시킨다. 그림 3.14에 있는 것처럼 기존의 backpropagation 모델은 한 층에 있는 뉴런들을 다음 층에 있는 뉴런의 입력으로 완전 연결시키는 다층 배열구조를 가지고 있다.

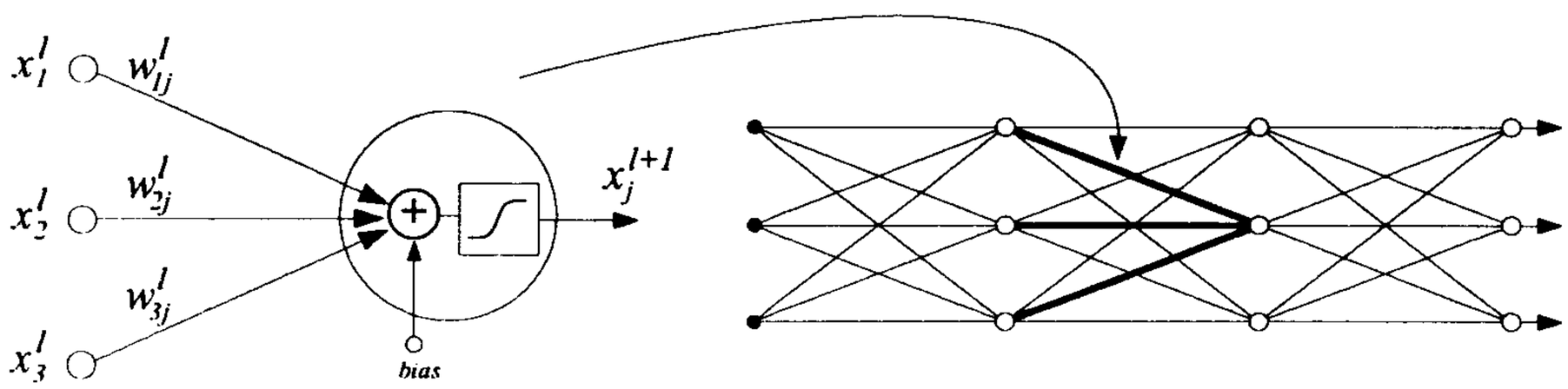


그림 3.14: Static neuron model

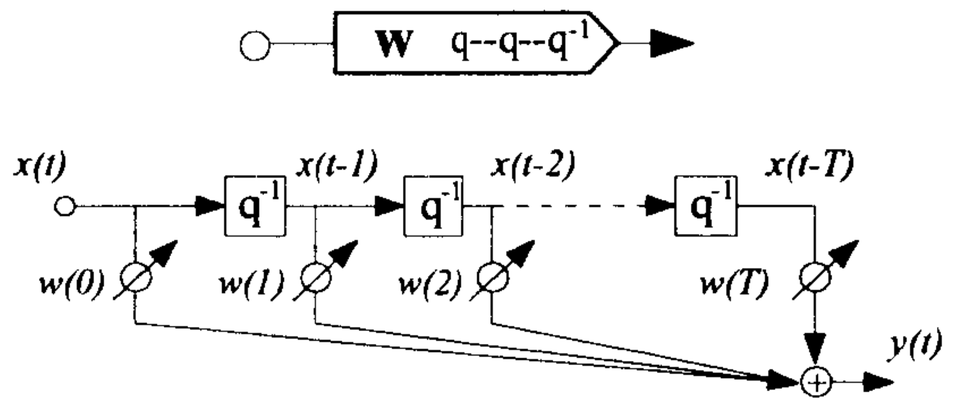


그림 3.15: FIR filter model

한 층에서의 뉴런 값은 연결 가중치과 곱해져서 다음층의 뉴런의 입력으로 들어가게 된다. 다음 층의 뉴런 값은 sigmoid 함수로 filtering 된다.

$$x_j^{l+1} = f(\sum_i w_{ij}^l x_j^l)$$

이러한 foreward 신경회로망은 내부적인 다이나믹스가 존재하지 않는 정적인 신경회로망이 된다. 그러나 이러한 정적인 연결구조에 FIR의 선형 필터를 쓰면 이 모델은 단순한 자기 회귀 모델과 비슷하다. 생물학적 과점에서 보면 FIR 연결구조는 마코프 모델의 신호 변환과 같다. 뉴런 i에서 뉴런 j로 가는 시냅틱 필터(FIR filter)는 벡터로 나타낼 수 있고, 이러한 시냅틱 필터로 다음 뉴런에서의 활성화 값은 다음식으로 나타낼수 있다.

$$\begin{aligned} W_{ij}^{l} &= [w_{ij}^{l}(0), w_{ji}^{l}(1), \cdots, w_{ji}^{l}(T^{l})] \\ x_{j}^{l+1}(k) &= f(\sum_{i} W_{ij}^{l} \cdot X_{i}^{l}(k)) \end{aligned}$$

FIR filter를 가진 모델의 학습방법은 기존의 backpropagation 학습알고리듬에 시간을 고려하여 학습시킨다. 주어진 입력 패턴 $x(k)$에 대해서 신경망의 출력값이 다음과 같다면 $(y(k) = N[W, x(k)]$, 여기서 W는 신경회로망 연결 가중치, N은 신경망), 각 패턴에 대한 error$(e(k))$와 모든 학습 패턴에 대한 error(C)는 다음과 같다.

$$\begin{aligned} e^{2}(k) &= ||d(k) - N[W, x(k)]||^{2} \\ C &= \sum_{k=1}^{K} e^{2}(k) \end{aligned}$$

Error 식 C에서 시간 k에서 연결 가중치에 대한 gradient $s_{j}^{l+1}(k)$를 구하면

$$\begin{aligned} \frac{\partial C}{\partial W_{ij}^{l}} &= \sum_{k} \frac{\partial C}{\partial s_{j}^{l+1}(k)} \cdot \frac{\partial s_{j}^{l+1}(k)}{\partial W_{ij}^{l}} \\ s_{j}^{l+1}(k) &= \sum_{i} W_{ij}^{l} \cdot X_{i}^{l}(k). \end{aligned}$$

출력층(L)에서 입력층(1)까지의 FIR 연결 가중치의 학습방법은 causality 조건에 따라서 다음과 같이 된다.

$$W_{ij}^{l}(k+1) = W_{ij}^{l}(k) - \eta \frac{\partial C}{\partial s_{j}^{l+1}(k)} \cdot \frac{\partial_{j}^{l+1}(k)}{\partial W_{ij}^{l}}$$

$$\delta_j^l(k) = \begin{cases} -2e_j(k)f'(s_j^L(k)) & (l = L) \\ f'(s_j^l(k)) \cdot \sum_{m=1}^{N_{l+1}} \overline{\delta}_m^{l+1}(k) \cdot W_{jm}^l & (1 \le l \le L-1) \end{cases}$$

$$\delta_j^{L-n}(k-nT) = \begin{cases} -2e_j(k)f'(s_j^L(k)) & (n = 0) \\ f'(s_j^{L-n}(k-nT) \cdot \sum_{m=1}^{N_{l+1}} \overline{\delta}^{L+1-n} m(k-nT) \cdot W_{jm}^{L-n} & \\ & (1 \le n \le L-1) \end{cases}$$

여기서 T는 필터 크기

3.5 Radial Basis Function Network

이 절에서는 신경망의 함수 근사 기능을 이용한 예측 방법 중 은닉층에 Radial Basis Function (RBF) 을 사용하고 입력 층과 출력층에 선형 activation 함수를 사용하는 3-층 feed-forward 구조의 Radial Basis Function Network 에 대해서 설명한다. 임의의 함수 근사에 있어 은닉층의 비선형 함수가 가져야할 조건으로는 Bounds 와 Locality 를 들 수 있다[13]. 분별 함수 $\phi(\mathbf{x})$ 가 다음과 같이 kernel 함수의 가중 합으로 표현될 수 있으며 아래의 bound 와 locality 의 성질을 만족할 때,

$$\phi(\mathbf{x}) = \sum_{i=1}^{M} c_i \psi(\mathbf{x}, \mathbf{p}_i)$$

(M 은 kernel 함수의 갯수이고 c_i 은 합의 가중치, $\mathbf{p}_i$ 은 i-th kernel 의 위치와 모양을 나타내는 모수 벡터)

Bounds :

$$L_\phi \le \psi(\mathbf{x}, \mathbf{p}_i) \le U_\phi$$

(L_ϕ 과 U_ϕ 은 $\psi(\mathbf{x}, \mathbf{p}_i)$ 의 상한과 하한을 각각 나타낸다.)

Locality :

$$\int_{-\infty}^{+\infty} (\frac{1}{a})^N \psi(\mathbf{x}, \mathbf{p}_a) = C$$

$$\lim_{a\to\infty}(\frac{1}{a})^N\psi(\mathbf{x},\mathbf{p}_a) = C\delta(\mathbf{x})$$

(N 는 $\mathbf{x}$ 의 차원 그리고 C 는 어느 양의 상수)

$\phi(\mathbf{x})$ 를 output 으로 구현하는 network은 compact 집합 상에서 정의되는 임의의 연속 함수를 임의의 정확도로 근사할 수 있는 universal function approximator 의 기능을 가짐을 보일 수 있다[13].

좋은 locality 와 수학적으로 좋은 성질들을 가지고 있는 Gaussian Potential Function (Gaussian PDF 의 normalize 안된 형태) 이 은닉층의 kernel 함수로 종종 사용된다.

$$\psi(\mathbf{x},\mathbf{p}_i) = \exp\{-d(\mathbf{x},\mathbf{p}_i)/2\}$$

$$d(\mathbf{x},\mathbf{p}_i) = d(\mathbf{x},\mathbf{m}_i,\mathbf{\Sigma}_i) = (\mathbf{x}-\mathbf{m}_i)'\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mathbf{m}_i)$$

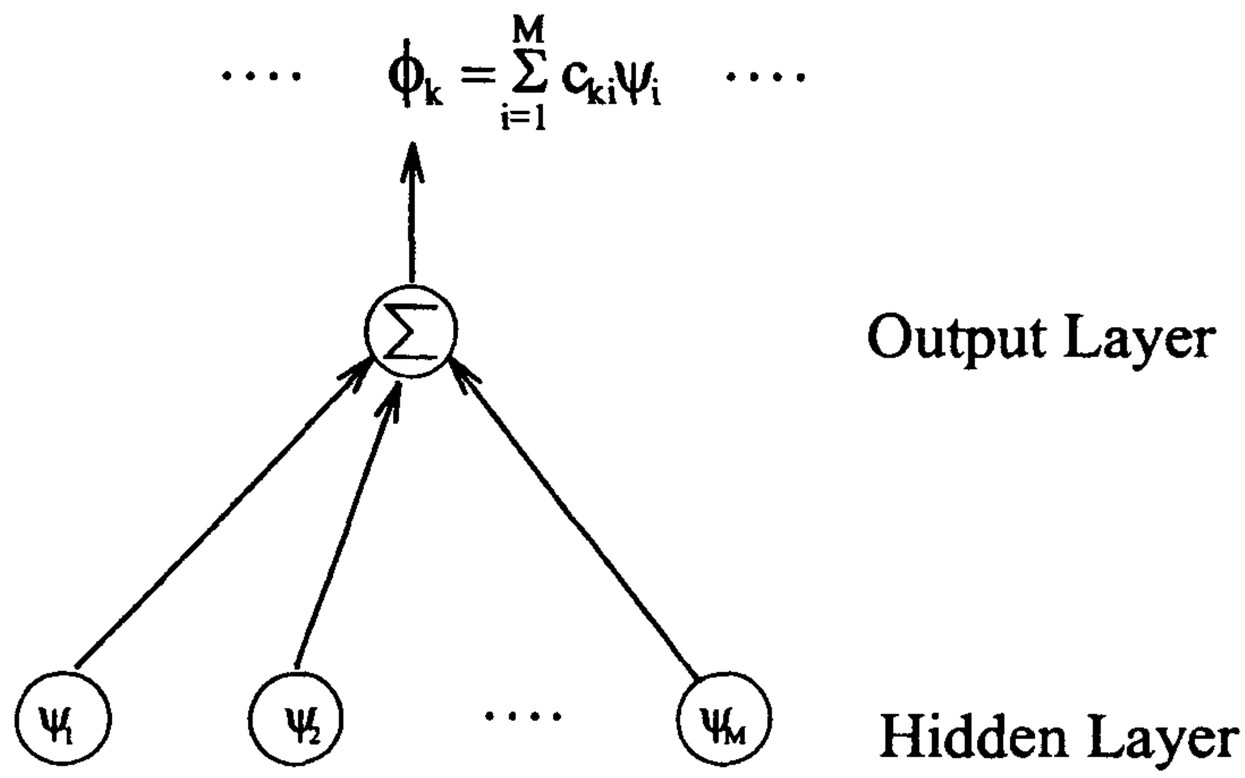
$\mathbf{x}$ 는 입력 패턴을, $\mathbf{m}_i$ 과 $\mathbf{\Sigma}_i$ 는 i-th kernel 함수의 평균 벡터와 공분산 행열을 나타낸다. 위의 함수를 은닉층에 사용한 Gaussian Potential Function Network (GPFN) 의 구조를 나타내면 그림 3.5 과 같다.

GPFN의 kernel 함수의 갯수 및 각 함수의 모수들과 출력층의 weight 들의 효율적인 학습법으로 hierachically self-organizing learning 방법이 제안되었다[16]. 각각의 kernel 함수는 입력 공간에서 정의되는 accomodation boundary 와 출력 공간 에서 정의되는 class representation을 갖게 되어 kernel 함수들의 self-recruitment 와 모수 추정의 두 단계를 거쳐 network 의 성능이 만족스런 수준에 이를 때까지 반복적으로 학습하는 것이다.

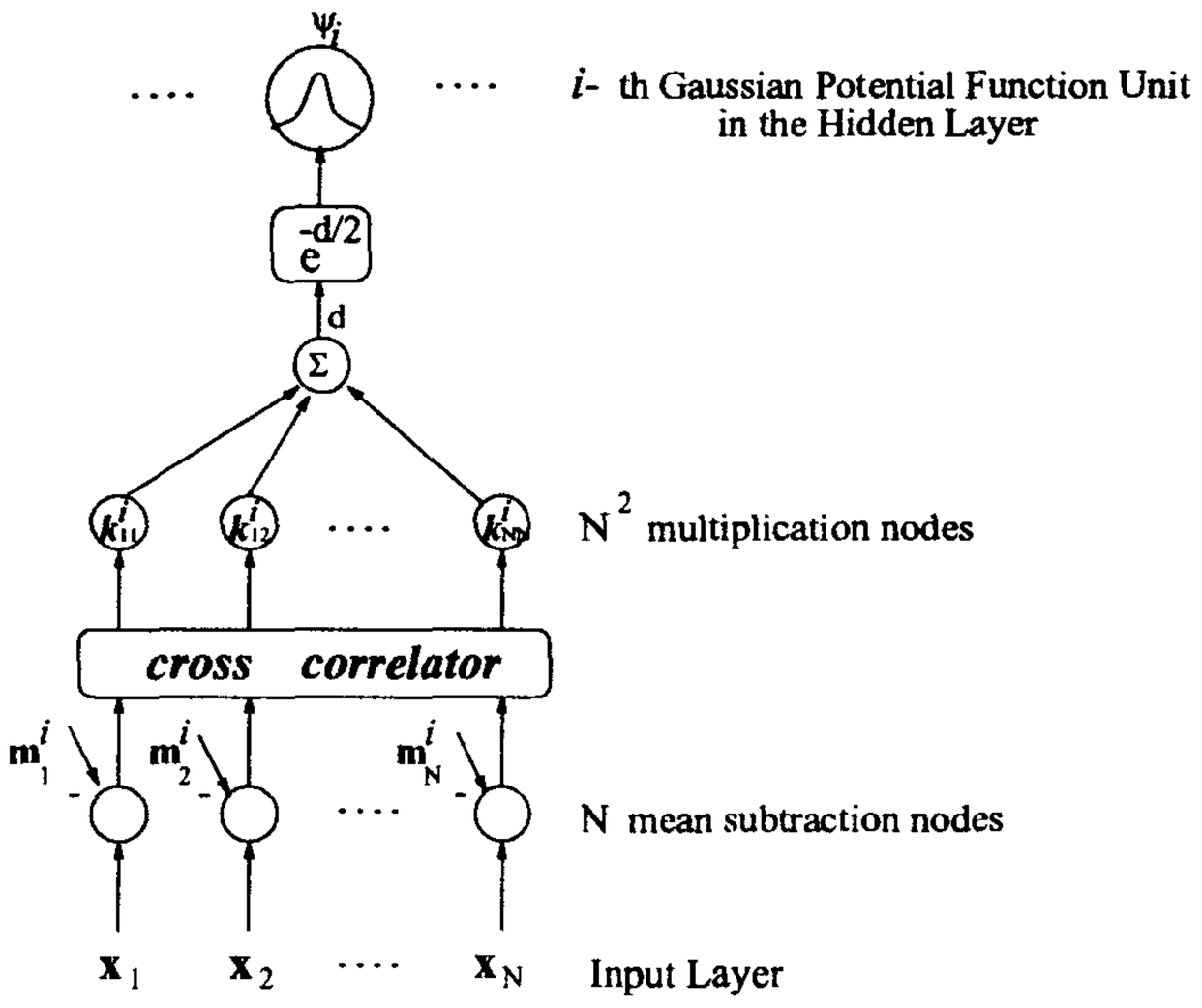
모수 재추정

Network 모수들을 Backpropagation 학습 알고리즘에 따라 재추정한다. j-th 출력 unit 과 은닉층의 i-th kernel 함수 사이의 weight의 수정항은

$$\eta\Delta w_{ji} = \eta(-\frac{\partial E_p}{\partial w_{ji}}) = \eta\cdot(t_j-\phi_j)\psi_i$$



(a) GPFN의 비선형 은닉층과 선형 출력층



(b) GPFN의 학습 pattern 입력과 은닉층 kernel 함수 unit의 계산

평균 벡터 $\mathbf{m}^i$의 j-th component의 수정항은

$$\eta\Delta m_j^i = \eta(-\frac{\partial E_p}{\partial m_j^i}) = \eta \cdot \sum_{l=1}^{N} k_{jl}^i (x_t - m_l^i)\psi_i \sum_{n=1}^{M}(t_n - \phi_n)w_{ni}$$

표준 편차 σ_j^i 의 수정항은

$$\eta\Delta\sigma_j^i = \eta(-\frac{\partial E_p}{\partial \sigma_j^i}) = \eta \cdot \sum_{l=1}^{N} k_{jl}^i \frac{(x_t - m_j^i)(x_t - m_l^i)}{\sigma_j^i}\psi_i \sum_{n=1}^{M}(t_n - \phi_n)w_{ni}$$

상관 계수 h_{jk}^i 의 수정항은

$$\eta\Delta h_{jk}^i = \eta(-\frac{\partial E_p}{\partial h_{jk}^i}) = -\eta \cdot \frac{1}{2}\frac{(x_j - m_j^i)(x_k - m_k^i)}{\sigma_j^i \sigma_k^i}\psi_i \sum_{n=1}^{M}(t_n - \phi_n)w_{ni}$$

이 된다. ($E_p = \frac{1}{2}\sum_{j=1}^{M}(t_{pj} - \phi_{pj})^2$. t_{pj}은 p-th 학습 패턴의 j-th desired output component. k_{jl}^i 는 i-th GPF unit의 역 공분산 행열의 j-th 행 l-th 열 요소.)

Hierarchically Self-Organizing 학습 알고리즘

1. 초기화

   (a) $i = 1$

   (b) $j = 0$

   (c) $p = 0$

   (i는 학습 반복 싸이클 수, j은 GPF unit의 갯수 그리고 p는 network에 입력된 학습 패턴의 갯수)

2. 학습 주기

   (a) 다음 학습 패턴의 입력 : $p = p + 1$

   (b) 만일 $\mid t_{pk} - \phi_{pk} \mid > \epsilon$

i. p-th 학습 패턴의 관련 class 와 같은 class를 가지는 GPF unit이 있고 p-th 학습 패턴이 그 GPU unit 의 hypersphere 안에 떨어질 경우 ⇒ 모수 재추정의 적용

ii. 그러한 GPF unit이 없는 경우 새 GPF unit 의 생성 :

$j = j + 1$ and $p = 0$

iii. 모수 학습 saturation measure 초기화

iv. 새 GPF unit 의 모수들 초기화

v. go to 2a

(c) 만일 $\mid t_{pk} - \phi_{pk} \mid < \epsilon$ 이면 모수 재추정의 적용

(d) 모수 학습 saturation measure 의 계산

(e) 만일 saturation 에 해당하면 각각의 GPF unit의 효과 반경을 줄인다 :

$$r_k^{new} = \begin{cases} r_k^{old} \cdot c & k = 1, \ldots, j \quad \text{if} \quad r_k > r_L \\ r_k^{old} & k = 1, \ldots, j \quad \text{otherwise} \end{cases}$$

(r_L 은 반경의 하한, c는 반경 수축율.)

3. If $p < N$ go to 2a

4. i=i+1

5. 만일 network performance 가 만족스러우면 멈춤 그렇지 않으면 go to 2

모수 학습의 saturation measure 로서는 j-th 출력 unit 에 대한 모수 saturation vector $\mathbf{s}_j$ 가 제안됨.

$$\mathbf{s}_j(p) = \alpha \frac{\partial E_p}{\partial \mathbf{n}_j} + (1 - \alpha)\mathbf{s}_j(p - 1)$$

($\mathbf{s}_j$는 j-th 출력 unit과 연관된 모든 모수 집합의 벡터, $0 < \alpha < 1$ 는 $\frac{\partial E_p}{\partial \mathbf{n}_j}$ 에 대한 decaying factor 그리고 p 는 학습 패턴의 index.) 이 $\mathbf{s}_j$ 의 목표는 현재 갯수의 GPF unit들로써 E_p의 극소화에 network 모수들이 충분히 수정되는가를

monitor 하는 것이다. Saturation 기준 $\rho_j(p)$은

$$\rho_j(p) = \begin{cases} \rho_j(p-1) + \beta \frac{\sqrt{d_j}}{\|\mathbf{s}_j(p)\|} & \text{if } p > p_0 \\ 0 \quad \text{otherwise} \end{cases}$$

j-th 출력 unit에 대한 $\|\mathbf{s}_j(p)\|^{-1}$ 의 integration 으로 정의하여 network 의 수행이 saturated 되었는가의 판단은 ρ_j와 $\|\mathbf{s}_j\|/\sqrt{d_j}$ 를 비교해서 한다. $\|\mathbf{s}_j\|$ 가 큰 값에 수렴하면 ρ_j는 서서히 증가하여 모수 saturation 을 나타낸다. 이 scheme 을 통해 각각의 GPF unit 의 유효반경을 줄이는 적당한 때를 효과적으로 정할 수 있다.

3.6 MLP Abnormality

MLP 학습진행을 방해하는 다음 요인들은 신경망 topology, 학습파라미터, 초기 연결 가중치, 학습패턴, 학습 알고리즘 등에서 발생한다. 이러한 abnormality는 초기 몇번의 학습을 통하여 알게되며 적절한 신경망 topology 설정, 학습파라미터 조절, 학습패턴 분석 및 전처리 등을 고려하여 재학습(retraining)시켜 극복하게된다. Abnormality 현상에는 신경망 모델의 일반화에 영향을 미치는 memorization을 발생시키는 overfitting 및 underfitting, premature saturation, overshooting, stalling condition 등이 있다.

Overfitting(또는 Overtraining) 잡음으로 인한 학습 데이타의 질때문에 일어나는 현상으로 학습 결과는 만족할수 있으나 test 패턴에 대하여 error 값이 높게 나타나는 현상을 말한다. 일반화(generalization)에 영향을 미치는 비 현상으로 새로운 잡음이 적은 학습데이타로 재학습시키거나 학습패턴을 전처리을 한후 재학습 등이 요구된다.

Underfitting 학습시 같은 부류에 속하는 학습패턴의 빈도수가 높은 패턴들은 fitting이 이루어지나 빈도수가 적은 부류에 속한 패턴들은 학습이 잘 이루어지지 않는 현상을 말한다. 학습패턴 선택시 각 부류에 속하는 패턴들이 고르게 포함될수 있도록 하여 학습시킨다.

Overshooting 학습 error의 수렴이 선형이 되지 않고 어느 정도 수렴후에 더 이상 수렴이 발생하지 않고 해 근처에서 진동하는 현상을 일컫는다.

Premature Saturation 일반적으로 MLP의 학습단계는 수렴단계, 경쟁단계, 우세단계로 구분된다. 학습 초기 파라미터 연결가중치, bias, 학습 및 momentum 율의 선택이 적절하지 않을때 학습 error가 커서 학습이 진행되지 않으므로 수렴단계에서 오랜 시간을 소비하는 현상을 premature saturation 이라 한다.

Stalling Condition 학습시 연결가중치가 무한히 커져 학습이 진행되지 않는 현상으로 학습패턴들이 각기 다른 독특한 dynamics를 갖는 경우에 발생한다. 문제의 dynamics을 내포하는 학습패턴을 다시 찾아 학습시키거나, 정규화를 통한 학습패턴의 범위를 조절함으로서 재학습을 한다.

기술된 비 현상은 학습 진행시 발생하며 문제의 abnormality를 찾을수 있는 방법은 없으나 error 그래프나 cross validation을 통하여 알수 있다. 이러한 문제을 예방하는 방법은 다음과 같다.

학습패턴 추가 및 전처리 빈도수가 적게 선택된 패턴을 추가하거나 만약 패턴을 추가 할수 없는 겨우 pseudo 패턴을 만들어 추가. 또한 문제 dynamics를 손상하지 않는 전처리 방법을 사용

Early stopping 학습시 학습 error의 수렴이 갑자기 높아지는 시점에서 학습을 끝내는 전략사용

Weight pruning 학습시 연결강도의 변화가 거의 일어나지 않는 연결강도는 제거시켜 일반화 성능을 저해하는 요인을 제거

여 백

4 장

신경망, 퍼지논리, 유전 알고리즘의 통합형 모델 개발

4.1 신경망의 비선형 입출력 함수의 확장

컴퓨터 비젼, 신호 처리, 음성/패턴 인식, 로보트 제어와 같은 많은 분야들에 신경망 모델들이 활발히 이용되고 있지만, 크게는 신경망의 공학적 응용 영역을 다음의 3가지로 분류할 수 있다. 즉, 주어진 패턴들을 원하는 클래스로 분류하는 구별(Classification) 기능과, 주어진 입출력 데이타를 이용하여 함수의 특성을 배우는 근사화 (Approximation) 기능, 그리고 여러가지의 제한 조건을 동시에 만족하며 최적해를 찾는 최적화(Optimization) 기능으로 나누어 생각할 수 있다. 본 과제에서의 신경망 모델은 입력 공간과 출력 공간사이의 사상 (Mapping)을 위한 학습의 도구로써 고려된다. 결국, Compact 집합 $U \subset R^p$ 에 속하는 어떤 입력과 또다른 Compact 집합 $V \subset R^m$ 에 속하는 출력에 대해, 주어진 입력값에 대해 원하는 출력값을 주면서, 새로운 입력값에 대해서도 일반화(Generalization)할 수 있는 설계 변수(Design parameter)들을 결정하는 문제로 귀결된다. 이 문제를 해결하기 위한 기본 골격(Framework)은 주어진 입출력 데이타로부터 연속이며 Multivariate 함수를 적절히 보간하거나 근사화하기위해 어떤 Basis 함수를 이용할 것인가와 또 설계 변수들을 어떻게 결정하는가의 근사화 이론 (Approximation theory)에 그 학문적 기초를 둔다. 즉, 근사화해야 할 어떤 함수 $f : R^p \rightarrow R^m$ 가 있을때, N차원의 변수 벡터 $\theta = [\theta_1, \theta_2, \cdots, \theta_N]^T$ 를 가지는 근사화 함수

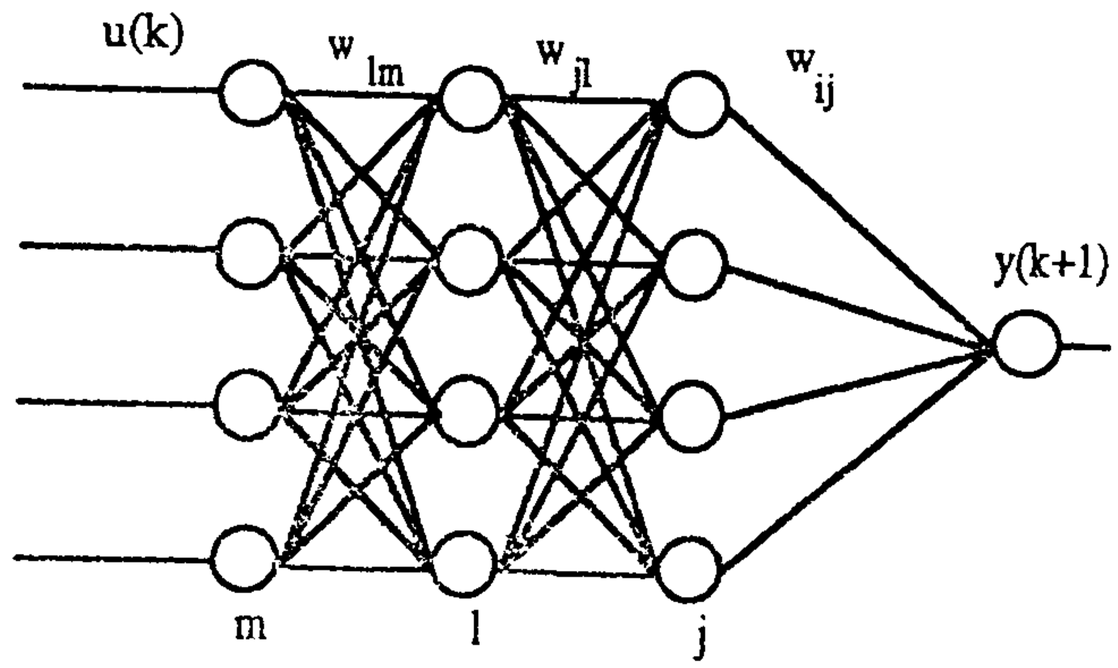


그림 4.1: 3층 구조 신경망 모델

$F(\theta) : R^p \rightarrow R^m$ 를 이용하여 주어진 입출력 데이타를 갖는 함수 $f[u^i] = v^i$ 를 가장 잘 표현하는 변수 벡터 θ^* 과 함수 $F(\cdot)$ 를 어떻게 결정하는가의 문제이다. 많은 수학자들이 이 문제를 해결하기 위해 노력해 왔으며 Taylor series, Fourier series, 또는 Polynomial series와 같은 많은 방법들이 연구되어 왔다. 다층 구조를 갖는 신경망의 연구 역시, 근사화 이론의 범주안에서는 이들 근사화 이론의 다른 방법의 새로운 시도로 간주될 수 있다.

입력 u와 출략 y를 가지는 n층의 신경망 모델은 다음 식 4.1로 표현될 수 있다.

$$f[W_n f[W_{n-1} \cdots f[W_1 u + b_1] + \cdots + b_{n-1}] + b_n] = y \qquad (4.1)$$

여기서 W_i는 i번째 층의 연결 강도값이고, 벡터 b_i 는 i번째 층의 임계값을 나타내고 함수 $f(\cdot)$ 는 시그모이드(Sigmoid)형태의 함수나 가우시안 (Gaussian) 형태의 함수가 많이 사용되고 있다. 그림 4.1은 3층 구조의 신경망 모델을 나타낸다.

신경망의 은닉층 뉴런의 입출력 전달 함수로 그림 4.2에서 실선으로 표시한 시그모이드가 가장 많이 이용되고 있으나, 오차 역전달 학습 알고리즘에 의한 신경망의 학습 과정이 지역 최소화(Local minima) 문제를 가지는 외에도 신경망의 연결 강도 값들이 다음 층의 뉴런 값들에 완전히 연결되어 있어 추가 학습이 필요한 경우 모든 학습 데이타를 이용하여 다시 학습해야하는 불편함이 존재한다. 한편, 그림 4.2에

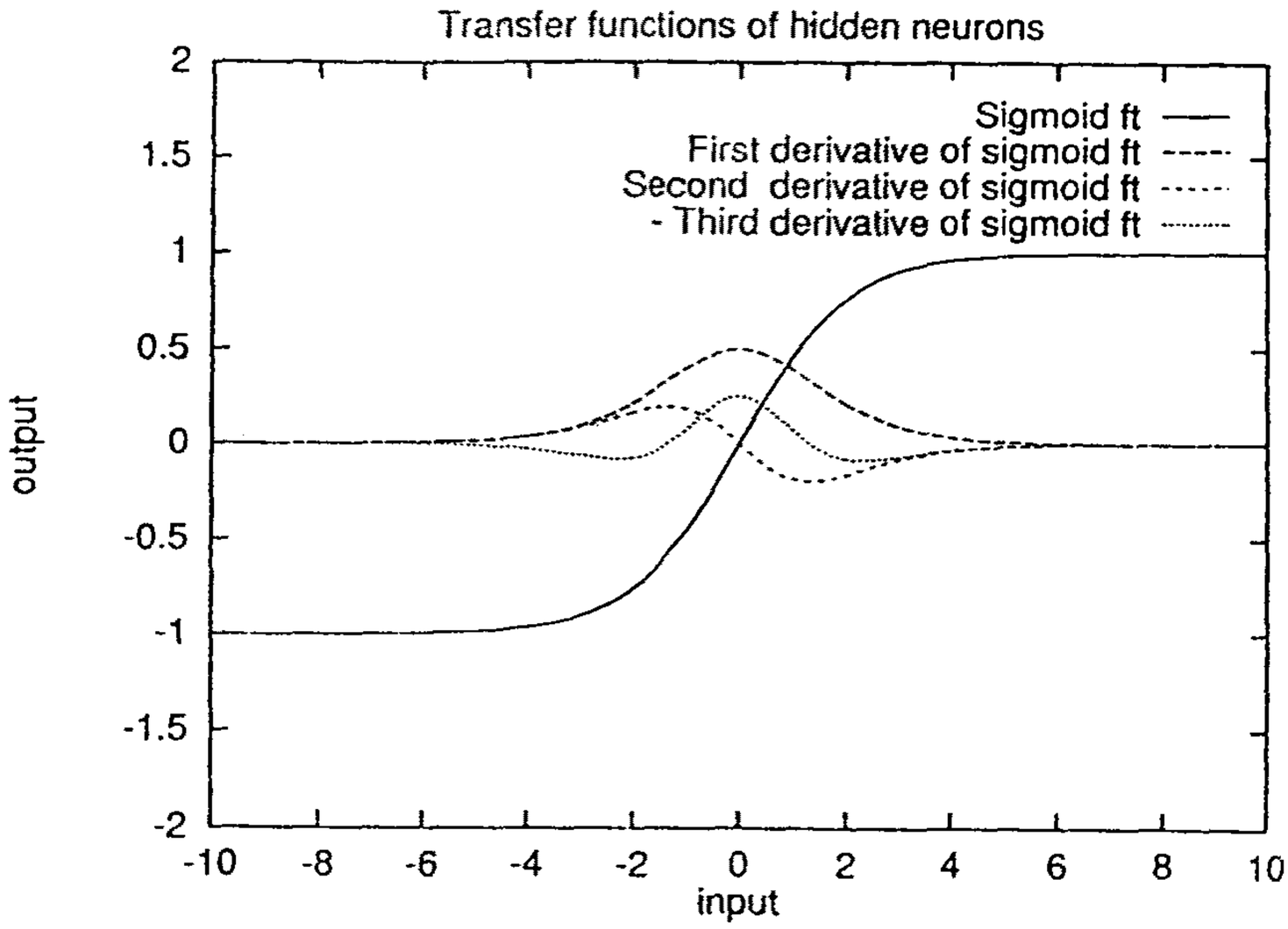


그림 4.2: 은닉층 뉴런의 전달 함수들

서 보인 시그모이드 함수 $f(\cdot)$ 를 일차 미분한 함수 형태인 $f'(\cdot) = (1 - f^2(\cdot))/2$ 는 그림 4.2에서 굵은 반선(Bold dashed lines)에서 보인 바와 같이 Gaussian 모양의 전달 특성을 가지게 되고, 이를 이용함으로써 신경망 학습 과정상의 지역 최소화(Local minima) 문제를 어느 정도 해결할 수 있을 것으로 기대할 수 있다. 즉, 학습 과정 동안에 지역 최소점(Local minima point)에 걸린 학습 패턴들을 가우시안 (Gaussian) 모양의 전달 특성을 갖는 뉴런들에 수록함으로써 지역 최소화(Local minima)문제를 해결할 수 있을 것으로 기대된다. 또한, 시그모이드 함수를 세번 미분한 값에 부호를 바꾼 함수는 그림 4.2에서 점선으로 보인 것처럼 Garbor 함수 형태를 가지며, 이와 같은 함수를 신경망의 은닉층 뉴런의 전달 함수로 이용함으로써 혼돈(Chaotic) 특성을 갖는 신경망을 설계할 수 있으며, 이를 이용함으로써 혼돈 (Chaotic) 특성을 갖는 time series를 예측하는데에 효과적인 모델로써 이용될 가능성이 있다. 본 과제에서는 이들 시그모이드 함수와 이 함수의 1차, 2차 혹은 그 이상의 미분형태를 갖는 함수들을 임의의 함수를 근사화하기 위한 신경망의 은닉층 뉴런의 전달 함수로 이용하여 근사화를 위한 Basis 함수로 사용하는 새로운 형태의 신경망 모델을 개발한다.

개발한 알고리즘은 임의의 다층 구조를 갖는 신경망 모델로 확장하여 사용할 수 있지만, 설명의 편의를 위해 그림 4.1에서 보인 2 층 구조 신경망을 이용하여 설명하고자 한다. 신경망의 출력층의 뉴런 수를 N_o라 하고, 은닉층 뉴런의 수를 N_h, 그리고 입력층의 뉴런 수를 N_i라고 하자. 또한 t_i와 y_i 는 각각 i번째 출력 뉴런의 목표값(Target value)과 실제값(Activations)이며, h_j 와 $\hat{h}_j$ 는 각각 j 번째 은닉층 뉴런의 실제값(Activations)과 비선형 함수를 거치기 전의 값이라고 하고, x_k 는 입력층의 k 번째 뉴런의 출력이라고 할때, 제안하는 신경망 모델은 다음 식들과 같이 표현될 수 있다.

$$\begin{aligned} h_j &= \sum_{m=1}^{M} \alpha_{mj} f^{(m-1)}(\hat{h}_j) \\ y_i &= \sum_{j}^{Nh} w_{ij}^{(2)} h_j \\ \hat{h}_j &= \sum_{k=1}^{N_i} w_{jk}^{(1)} x_k \end{aligned} \tag{4.2}$$

여기서 $w_{jk}^{(1)}$ 와 $w_{ij}^{(2)}$ 는 각각 첫번째와 두번째 층의 연결 강도 (Interconnection weight)를 나타내고, α_{mj} 은 은닉층의 j 번째 뉴런의 시그모이드 함수를 $m-1$ 차 미분한 함수의 계수이다. 또한, $f^{(t)}(\cdot)$는 함수 $f(\cdot)$ 를 t번 미분한 함수를 나타낸다. 만약 Bipolar값을 갖는 시그모이드 함수를 이용하며 M이 4일때,

$$\begin{aligned} f^{(0)}(x) &= \frac{2}{1+\exp(-x)} - 1 \\ f^{(1)}(x) &= \frac{1-f^2(x)}{2} \\ f^{(2)}(x) &= -\frac{f(x)(1-f^2(x))}{2} \\ f^{(3)}(x) &= -\frac{(1-f^2(x))(1-3f^2(x))}{4} \\ f^{(4)}(x) &= \frac{f(x)(1-f^2(x))(2-3f^2(x))}{2} \end{aligned} \tag{4.3}$$

로 주어진다. 식 4.3에서 보는 바와 같이 시그모이드 함수 $f(\cdot)$를 m차 미분한 함수는 $f(\cdot)$의 $m+1$차 다항식으로 표현된다는 것을 알 수 있다. 따라서, 어떤 함수를 근사화하는데 사용되는 신경망의 Basis 함수를 $f(\cdot)$의 1차항뿐만 아니라 고차항을 이용하는 것으로 간주할 수가 있다. 따라서, 주어진 함수를 근사화하는데 필요한 은닉층 뉴런의 수가 단순히 시그모이드 함수만을 이용하는 경우보다 더 작아도 될 것으로 기대되며, 또한 시그모이드 함수의 미분에서부터 생기는 함수 들의 특성이 그림 4.2에서 보인 것처럼 가우시안(Gaussian) 함수나 Garbor함수의 형태를 가지므로 혼돈(Chaotic) 특성을 갖는 Time series prediction과 같은 문제를 해결하는데 효과적일 수 있는 가능성이 있다.

이제, 신경망의 연결 강도(Interconnection weight)값과 식 4.3에서 보인 미분 함수들의 계수 α_{mj} 를 결정하는 방법을 설명하고자 한다. 이들 변수들을 적응적인 방법으로 찾기 위해 다음 식 4.4에서 나타낸 오차 함수에 최대 오차 급경사 강하법(Steepest decent method)을 적용하면 다음과 같은 학습 알고리즘을 얻을 수 있다. 우선, 연결 강도(Interconnetion weight)값의 변화분은 다음 식 4.5와 같다.

$$E = \frac{1}{2}\sum_{i=1}^{N_o}(t_i - y_i)^2 \tag{4.4}$$

$$\Delta w_{ij}^{(2)} = -\delta_i h_j \tag{4.5}$$

$$\Delta w_{jk}^{(1)} = -\delta_i w_{ij}^{(2)} \sum_{m=1}^{M} \alpha_m f^{(m)}(\hat{h}_j)$$

따라서, 오차 보정을 한 연결 강도(Interconnection weight)는

$$w_{jk}^{(1)}(new) = w_{jk}^{(1)}(old) - \eta \Delta w_{jk}^{(1)} \tag{4.6}$$

$$w_{ij}^{(2)}(new) = w_{ij}^{(2)}(old) - \eta \Delta w_{ij}^{(2)}$$

가 된다. 여기서 η는 학습율이다. 또한 시그모이드의 미분 함수들의 계수 α_{mj} 역시 최종 출력단 오차가 작아질 수 있도록 다음 식 4.7과 같이 학습 시킬수가 있

Knowledge \ Framework	Symbolic	Numerical
Structured	전문가 시스템	퍼지-논리
Unstructured	유전알고리즘	신경망

표 4.1: 지식의 분류 및 표현 방법

다.

$$\alpha_{mj}(new) = \alpha_{mj}(old) + \eta_\alpha \delta_i w_{ij}^{(2)} f^{(m-1)}(\hat{h}_j) \tag{4.7}$$

여기서 η_α는 시그모이드 함수를 미분한 함수들의 계수 α_{mj}를 학습시키 기 위한 학습율이다.

한편, 제안한 신경망을 이용하는데 있어 고려해야 할 한가지는 제안한 방법이 많은 지역최소점(Local minima)를 가질 수 있으므로 향후 이 문제를 개선해야 할 필요가 있다.

4.2 뉴로-퍼지망

실제 어떤 문제를 풀고자 할때 이용 가능한 정보는 다음 표 4.1에 나타낸 바와 같이 크게 구조적인 정보와 비구조적인 정보로 대별된다고 할 수 있다[15].

표 4.1에서 보는 바와 같이 신경망 모델은 비구조적인 정보를 수치적인 표현 방법으로 처리하는 정보 처리 메카니즘이고, 퍼지 논리는 구조적인 정보를 수치적 표현 방법으로 처리하는 정보 처리 메카니즘이다. 따라서, 신경망만으로는 주어진 문제를 해결하고자 할때 얻을 수 있는 구조적인 지식을 충분히 이용할 수 없다. 한편 퍼지 논리는 비구조적인 지식을 다루는데 그 한계가 존재하며, 구조적인 정보를 이용하는데에도 잘 표현된 퍼지 규칙 및 퍼지 설계 변수들이 필요하다. 따라서, 구조적인 정보뿐만 아니라 비구조적인 정보를 효과적으로 다루기 위해서 기존의 퍼지 논리를 신경망의 골격(Framework)안으로 도입하여 이들 정보들을 충분히 이용할 수 있는 새로운 모델을 개발하는 것이 필요하다.

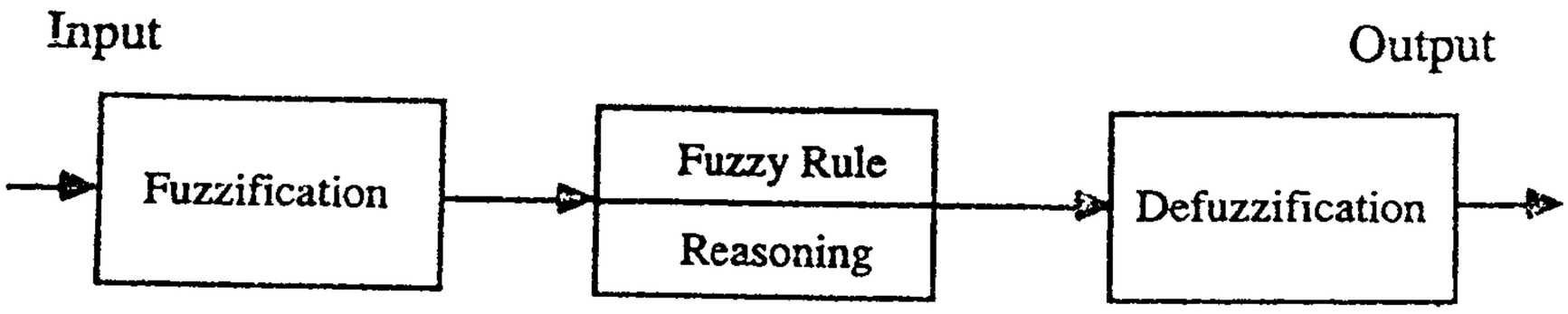


그림 4.3: 일반적인 퍼지 논리의 구조

그림 4.3은 퍼지 논리를 이용하는 일반적인 구조이다[29]. 퍼지화 과정을 통해서 입력 공간을 적절히 퍼지 분할하여 주어진 데이타에 대해 하나의 퍼지 집합을 얻는다. 이를 위해 전문가로부터 얻은 퍼지 규칙을 이용하여 퍼지 추론 과정을 거쳐 나온 결과를 비퍼지화 (Defuzzification) 과정을 통해 최종 결과를 얻는다. 이 과정에서 미리 퍼지 규칙을 잘 설정해야 할 뿐만 아니라, 퍼지화와 비퍼지화 과정에서의 여러가지 설계 변수들을 잘 설정해야만 원하는 결과를 얻을 수 있다. 하지만, 이들 값들을 주어진 문제에 맞게 설계하기란 아주 어려운 일이다. 따라서, 신경망의 적응 학습 기능을 이용하여 이들 변수들을 최적화하고자 하는 연구들이 많이 있어 왔다[10][28]. 하지만, 이들 방법들은 단순히 찾고자하는 퍼지 논리의 설계 변수들을 신경망의 연결 강도(Interconnection weight)로 표현하는 연구가 대부분인데, 만약 초기에 설정한 변수들의 값이 원하는 값에 가깝지 않거나 자유도(Degrees of freedom)의 수가 충분하지 않으면 학습에 의한 성능의 큰 향상을 기대하기가 어렵다[18]. 따라서, 본 과제에서 제안하는 뉴로-퍼지 모델은 퍼지화와 비퍼지화 과정을 그대로 두고, 퍼지 규칙과 추론과정을 신경망으로 대체한 구조로 개발한다. 그림 4.4는 제안한 뉴로-퍼지 구조 이용하여 시스템 특성을 동정화(Identification)하는 방법을 나타낸다.

주어진 시스템을 기준 입출력 집합들 사이의 상관 관계, 즉 퍼지 규칙으로 표현하기 위하여 시스템의 입출력값을 각각의 기준 퍼지 집합들에 대하여 나타낼 필요가 있다. 이는 실제 시스템의 입력과 동적 특성을 학습하기 위해 필요한 시스템의 상태값을 퍼지 싱글톤(Fuzzy singleton)으로 하여 이 값과 전체 집합(Universe of discourse)상에 미리 정의된 몇 개의 기준 퍼지 집합과의 유사도를 다음 식 4.8과

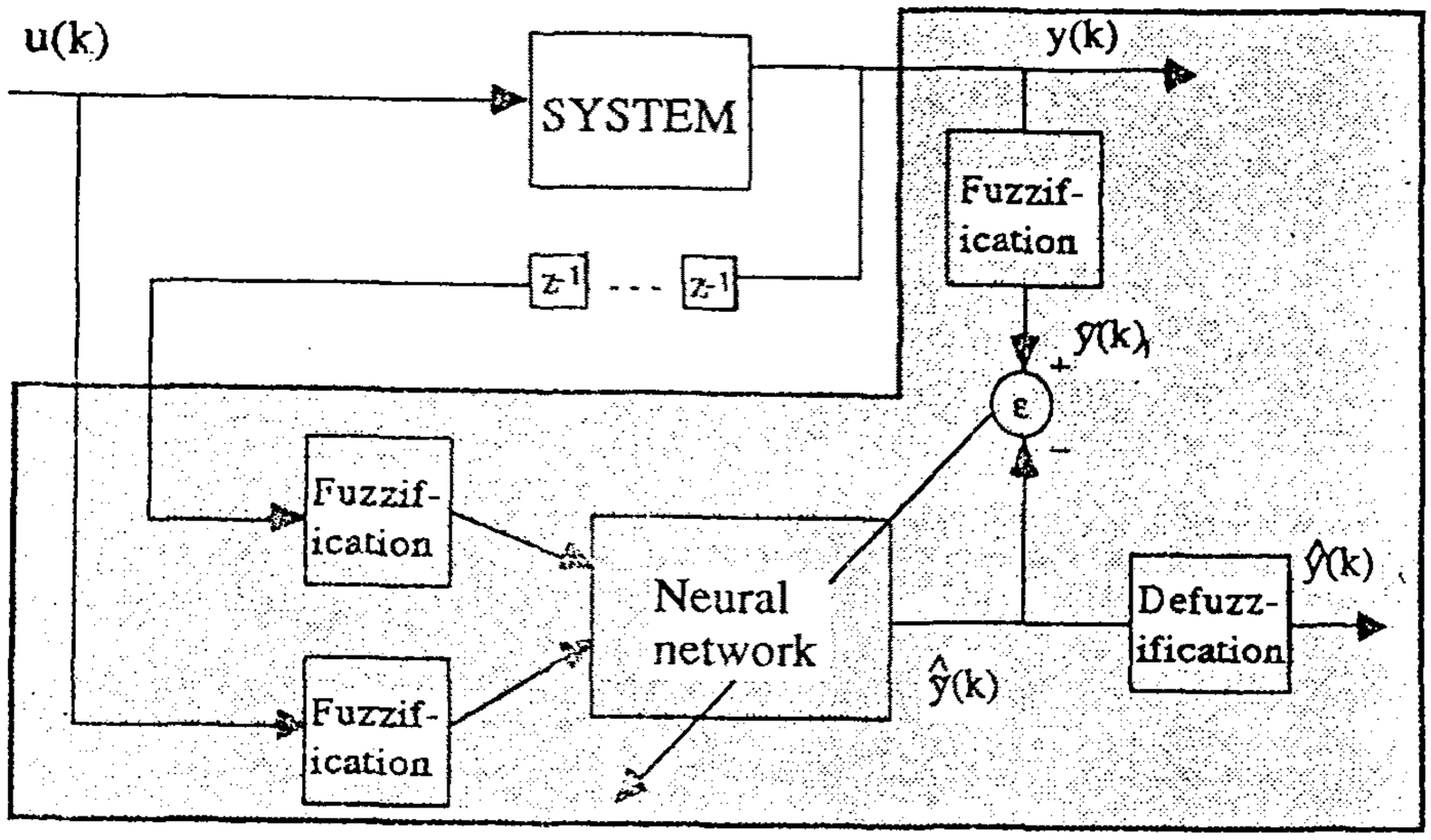


그림 4.4: 제안한 뉴로-퍼지망

같이 계산함으로써 하나의 Fuzzy 집합을 얻을 수 있다[18].

$$\begin{aligned} \tilde{X}_k &= fuzzification(x_k) = \{x_k^1/CX^1, x_k^2/CX^2, \cdots, x_k^M/CX^M\} \\ x_k^i &= possibility\left(x_k \mid \tilde{RX}^i\right) = \max\left(\tilde{RX}^i \cdot \delta(x - x_k)\right) \end{aligned} \tag{4.8}$$

식 4.8에서 $\tilde{RX}^i$는 i번째 기준 퍼지 숫자(Fuzzy number)를 위한 소속 함수(Membership function)를 나타내고, $x_k^1, x_k^2, \cdots, x_k^M$는 실제 시스템의 k번째 입출력 값에 대해 기준점 $CX^1, CX^2, \cdots, CX^M$에 대한 기준 퍼 지 집합에서의 소속도를 나타낸다. $\delta(\cdot)$는 퍼지 싱글톤 함수를 나타낸다. 설명한 퍼지화 과정은 그림 4.5와 같다.

이렇게 해서 얻은 퍼지 집합을 퍼지 규칙을 찾기 위한 신경망 모델의 입력으로 이용한다. 실제 시스템의 출력값을 역시 퍼지화하여 신경망 모델의 출력값과 비교하고, 오차 함수 $E_k = \sum_i (\tilde{Y}_k - \hat{y}_k^i)^2/2$를 최소화하도록 최대 오차 급경사 강하법(Steepest descent method)에 의해 신경망의 연결 강도(Interconnection

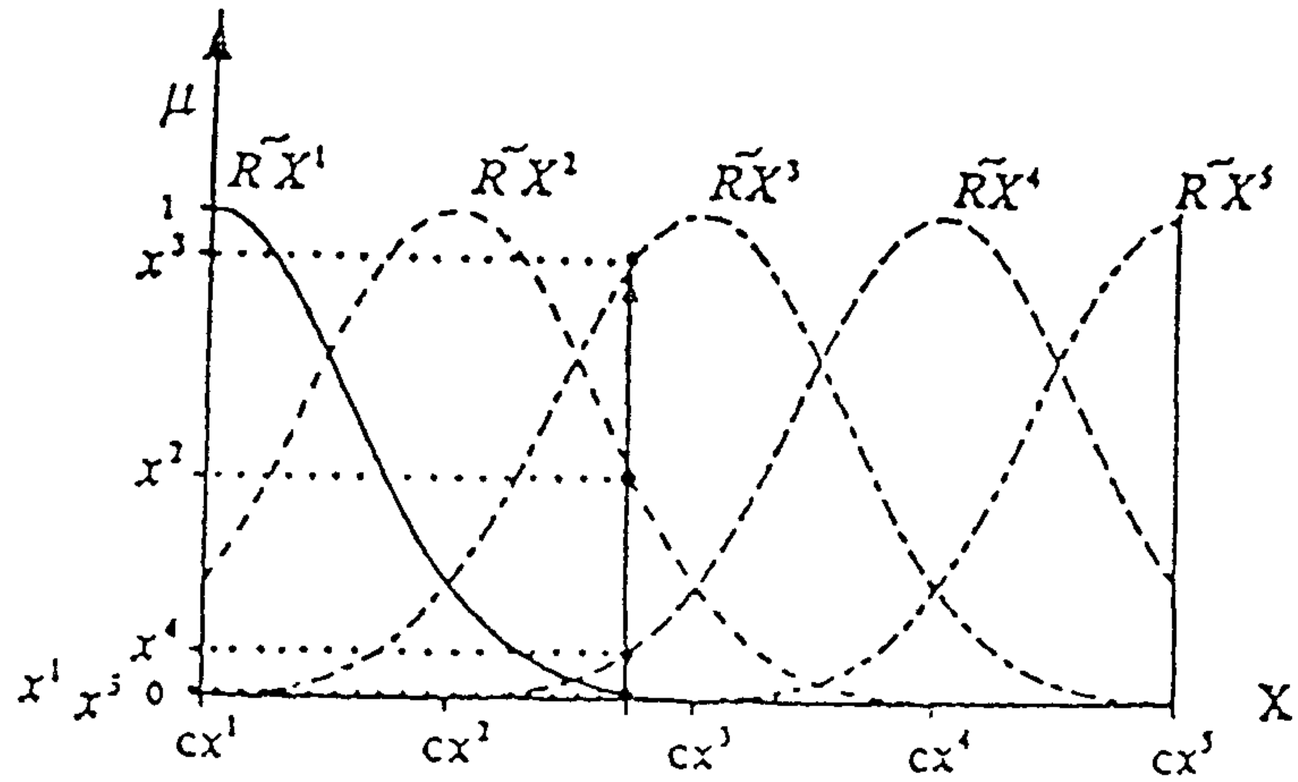


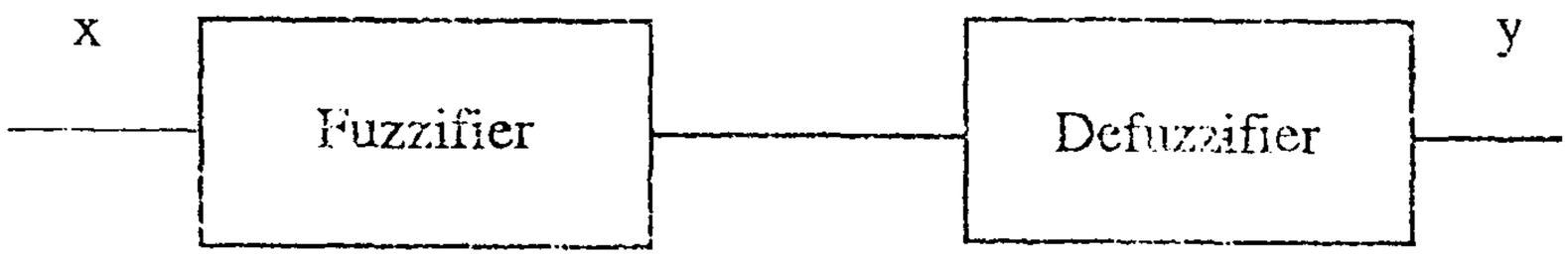
그림 4.5: 기준 퍼지 집합에 대한 데이타의 퍼지화 과정

weight)를 바꾼다. 여기서 $\tilde{Y}_k = \left\{y_k^1/CY^1, y_k^2/CY^2, \cdots, y_k^M/CY^M\right\}$이고, $y_k^1, y_k^2, \cdots, y_k^M$는 역시 몇개 의 기준점 $CY^1, CY^2, \cdots, CY^M$ 에 대한 소속도를 나타낸다. 실제 시스템의 출력에 대응하는 뉴로-퍼지 동정화 모델의 출력을 얻기 위해 필요한 비퍼지화 방법으로 다음 식 4.9의 면적 중심 방법을 이용한다[29].

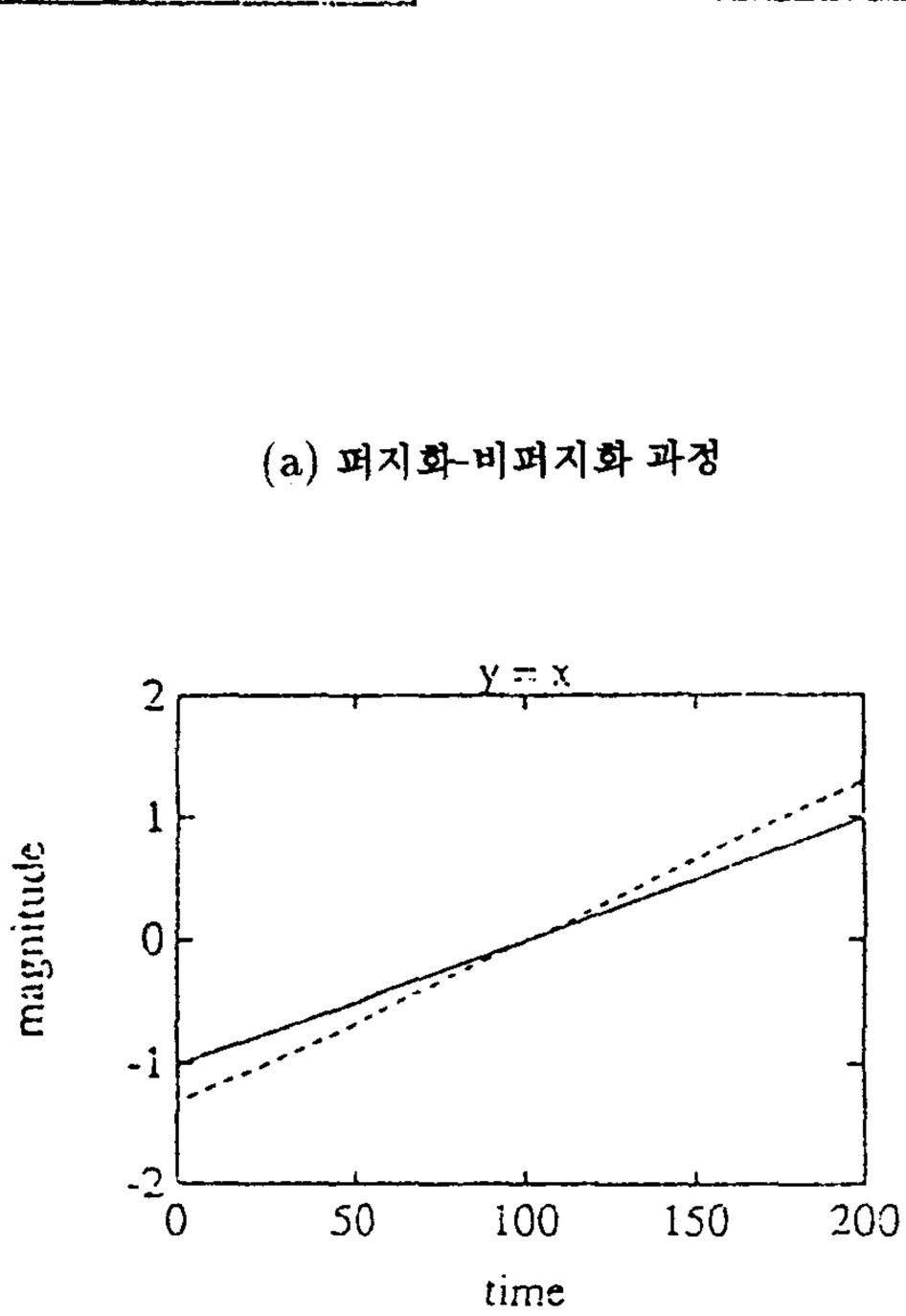
$$\hat{y}_k = defuzzification(\hat{\tilde{Y}}_k) = \frac{\sum CY^p \hat{y}_k^p}{\sum \hat{y}_k^p} \tag{4.9}$$

하지만, 이와 같은 구조는 신경망의 학습 오차가 충분히 줄어들었음에도 불과하고 여전히 동정화(Identification) 오차를 유발할 수 있는데 이는 주로 퍼지화-비퍼지화 과정에 의해 발생하는 오차 때문이라고 할 수가 있다. 즉, 그림 4.6 (a)와 (b)에서 보는 바와 같이 단순히 퍼지화와 비퍼지화 과정만에 의해서는 오차가 남게 되는데 이를 보상하기 위해서 다음 그림 4.7 (a)에서처럼 신경망의 학습을 위한 오차값 을 비퍼지화 블럭밖에서 잡음으로써 그림 4.6 (b)의 퍼지화-비퍼지화의 오차를 줄일 수 있음을 그림 4.7의 (b)에서 알 수가 있다.

한편, 퍼지 논리를 이용함에 있어 기준 퍼지 숫자를 이용한 입출력 데이타의 퍼지화 과정의 소속 함수(Membership function)의 중심점 및 모양을 잘 결정하는 것이 매우 중요하다. 이를 위해 그림 3.7의 뉴로-퍼지 구조를 퍼지 규칙을 찾기 위한 신경망 블럭과 비퍼지화 블럭의 중심값의 적응적 학습에 부가적으로 입력측 데

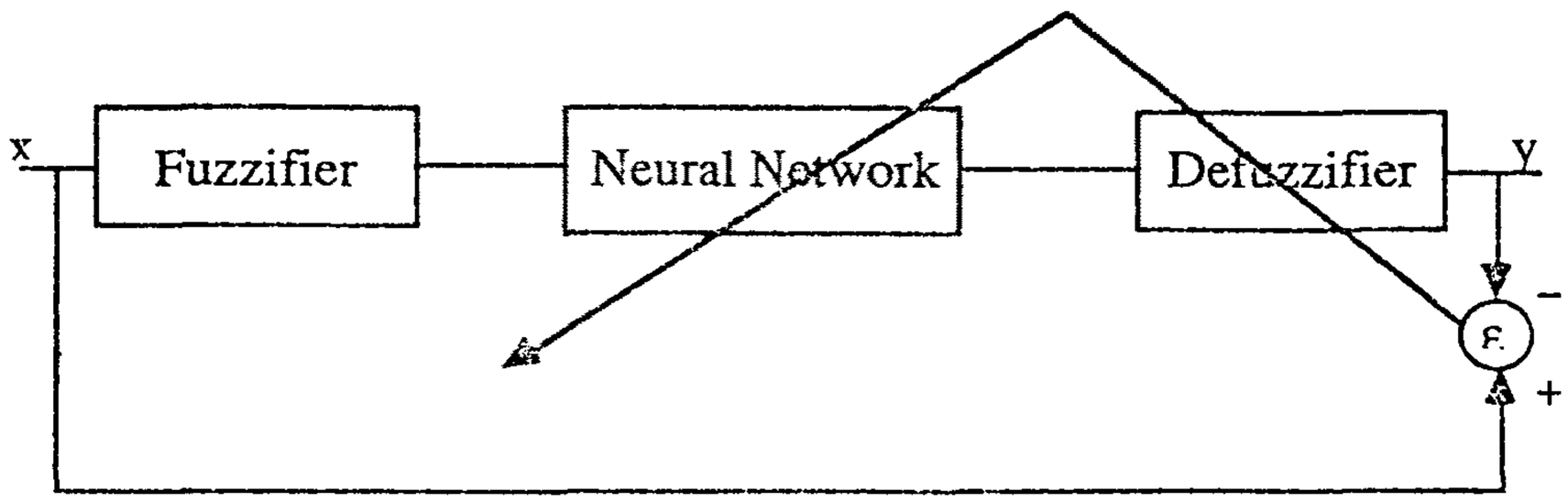


(a) 퍼지화-비퍼지화 과정



(b) 퍼지화-비퍼지화 오차

그림 4.6: 퍼지화-비퍼지화 과정의 오차



(a) 퍼지화-신경망-비퍼지화 과정

(b) 보상된 퍼지화-비퍼지화의 오차

그림 4.7: 신경망을 이용한 퍼지화-비퍼지화 과정의 오차

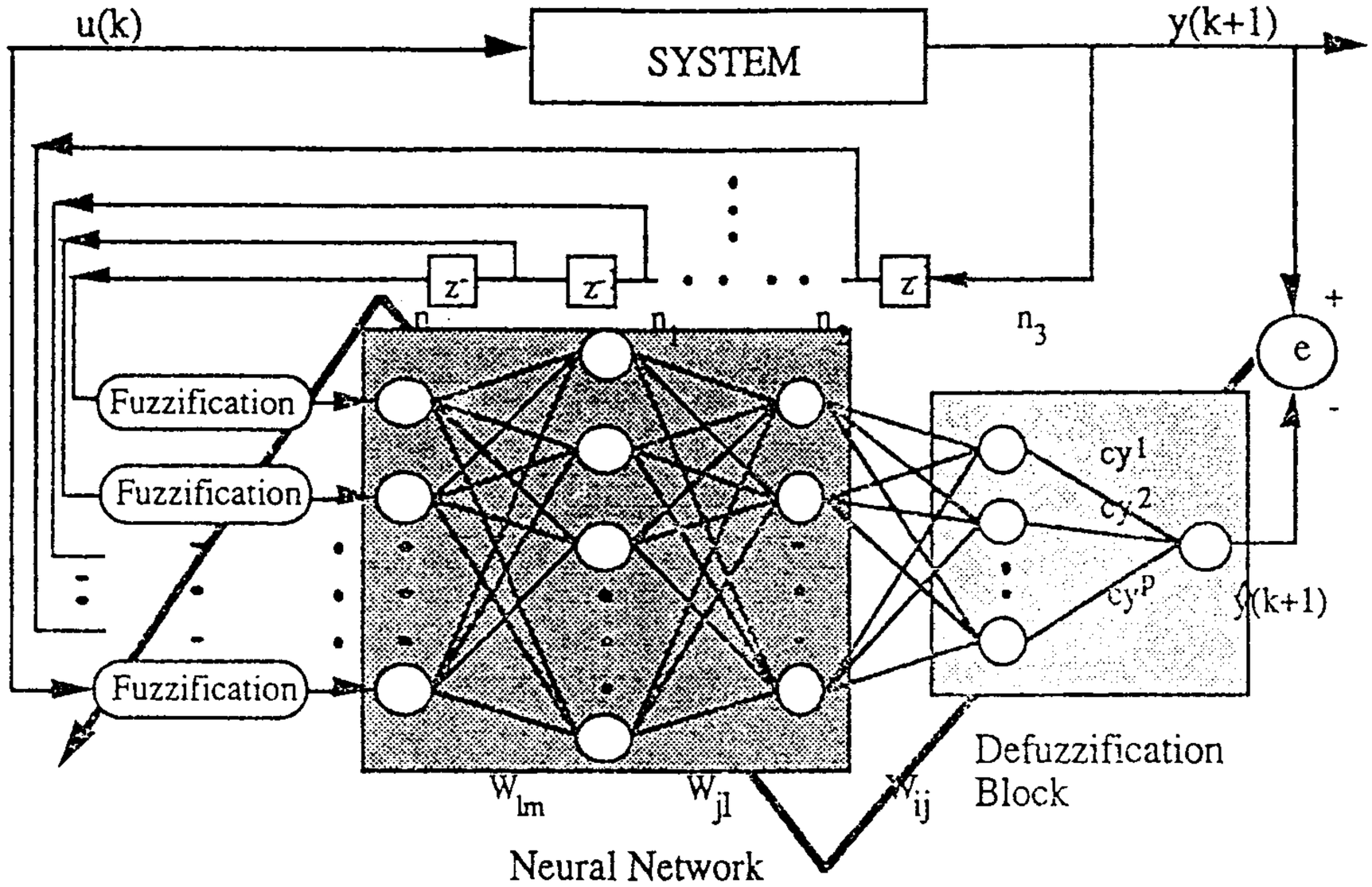


그림 4.8: 수정된 뉴로-퍼지망을 이용한 시스템 동정화

이타를 퍼지 소속 함수까지도 적응적으로 학습시킬 수가 있다. 수정된 뉴로-퍼지 구조를 이용한 임의의 동적 시스템을 동정화(Identification)하는 구조는 다음 그림 4.8과 같다.

수정된 뉴로-퍼지 동정화 구조의 출력은 식 4.9에서 보인 것처럼 비퍼지화의 하나인 무게 중심법(Center of gravity)에 의해 구해지는데 여기서 CY^p는 그림 4.5에서 출력 퍼지 숫자의 중심점을 나타내고, 또한 동정화 학습 과정동안 그림 4.8의 비퍼지화 블럭안의 고정된 연결 강도(Interconnection weight)로 작용한다. 한편, 신경망 블럭의 출력은 퍼지 숫자에서의 가상적인 소속 함수값(Virtual membership function value)들의 집합이 된다. 그러나, 이 가상적인 퍼지 소속 함수값은 오차를 줄이기 위한 적응 학습 알고리즘에 의해 퍼지 소속 함수 (Fuzzy membership function)가 만족해야 하는 여러가지 성질 중 Convexity를 만족하지 못하는 경우가 생길 수 있다. 이를 고려하기 위해 학습을 위한 오차 함수를 다음 식 4.10과 같이 기존의 최종 출력단 오차에 Convexity를 위한 부가적인 오차

함수를 추가하여 이용한다[18].

$$E(k+1) = \frac{1}{2}\left[(y(k+1) - \hat{y}(k+1))^2 + \sum_{p}^{n_3} \delta^p(k)^2\right] \tag{4.10}$$

그리고,

$$\hat{y}(k+1) = \frac{\sum_p^{n_3} CY^p \hat{y}^p(k)}{\sum_p^{n_3} \hat{y}^p(k)} = \frac{\sum_p^{n_3} CY^p f_1(\sum_j^{n_2} W_{pj} h_j^2)}{\sum_p^{n_3} f_1(\sum_j^{n_2} W_{pj} h_j^2)} \tag{4.11}$$

여기서 n_i는 i 번째 층의 뉴런 수를 의미하고, $\delta_p(k)$ 는 출력 퍼지 소속값의 Convexity를 위한 측정(Measure)값으로 다음 식 4.12로 주어진다.

$$\delta_p(k) = \begin{cases} \max\{\hat{y}^p(k) - \hat{y}^{p+1}(k), 0\} & whenp < P_{\max} \\ \max\{\hat{y}^p(k) - \hat{y}^{p-1}(k), 0\} & whenp > P_{max} \end{cases} \tag{4.12}$$

식 4.12에서 $P_{\max}$는 신경망 블럭의 출력단값들 중 최고값을 갖는 p번째 뉴런을 나타낸다. 식 4.10에서 주어진 오차 함수를 이용하여 퍼지 규칙을 찾기 위해 신경망 블럭의 연결 강도(Interconnection weight) 값들을 오차 역전달 학습 알고리즘으로 학습시킬 수가 있다. 그림 4.8의 부가적인 설계 변수들인 입출력 소속 함수(Membership function)들의 모양과 중심점의 학습 알고리즘은 다음과 같다.

$$CY^p(k+1) = CY^p(k) + \alpha\{y(k+1) - \hat{y}(k+1)\}\frac{\hat{y}^p(k)}{\sum_p^{n_3} \hat{y}^p(k)} \tag{4.13}$$

그리고,

$$m_i(k+1) = m_i(k) + \alpha_m \epsilon_m(k+1) x_i \tag{4.14}$$

여기서

$$\epsilon_m(k+1) = \epsilon(k+1) \sum_j^{n_2} W_{pj} f' \left(\sum_l^{n_1} W_{jl} h_l^1 \right) \sum_l^{n_1} W_{jl} f' \left(\sum_m^{n} W_{lm} x_m \right) \frac{x_i - m_i}{\sigma_i^2(k)} \tag{4.15}$$

이고,

$$\sigma_i^2(k+1) = \sigma_i^2(k) + \alpha_\sigma \epsilon_\sigma(k+1) x_i \tag{4.16}$$

여기서,

$$\epsilon_\sigma(k+1) = \epsilon(k+1) \sum_j^{n_2} W_{pj} f' \left(\sum_l^{n_1} W_{jl} h_l^1 \right) \sum_l^{n_1} W_{jl} f' \left(\sum_m^{n} W_{lm} x_m \right) \frac{(x_i - m_i)^2}{\sigma_i^3(k)} \tag{4.17}$$

이다. 또한,

$$\epsilon(k+1) = (y(k+1) - \hat{y}(k+1)) \frac{CY^p \sum_p^{n_3} \hat{y}^p(k) - \sum_p^{n_3} CY^p \hat{y}^p(k)}{\sum_p^{n_3} \hat{y}^p(k))^2} f_1' \left(\sum_p^{n_2} W_{pj} h_j^2 \right) + \sum_p^{n_3} \delta^p(k) f_1' \left(\sum_p^{n_3} W_{pj} h_j^2 \right) \tag{4.18}$$

이고, 여기서

$$\begin{aligned} f_1(r) &= \frac{1}{1 + \exp(-\alpha r)} \\ f(r) &= \frac{2}{1 + \exp(-\alpha r)} - 1 \end{aligned} \tag{4.19}$$

이다. 시그모이드 함수 $f(\cdot)$는 은닉층 뉴런을 위한 Bipolar 함수이고, $f_1(\cdot)$는 가상적인 퍼지 소속 출력값을 얻기 위한 신경망 블럭의 출력단을 위한 Unipolar 함수이다. 또한 $f'(\cdot)$와 $f_1'(\cdot)$는 각각의 1 차 미분한 함수를 나타내고, m_i와 $\sigma_i(k)$는 각각 시점 k 에서의 i 번째 입력측 퍼지 소속 함수의 중심점과 분산을 나타낸다. 제안한 알고리즘 으로 신경망 블럭의 연결 강도(Interconnection)값과 퍼지의 설

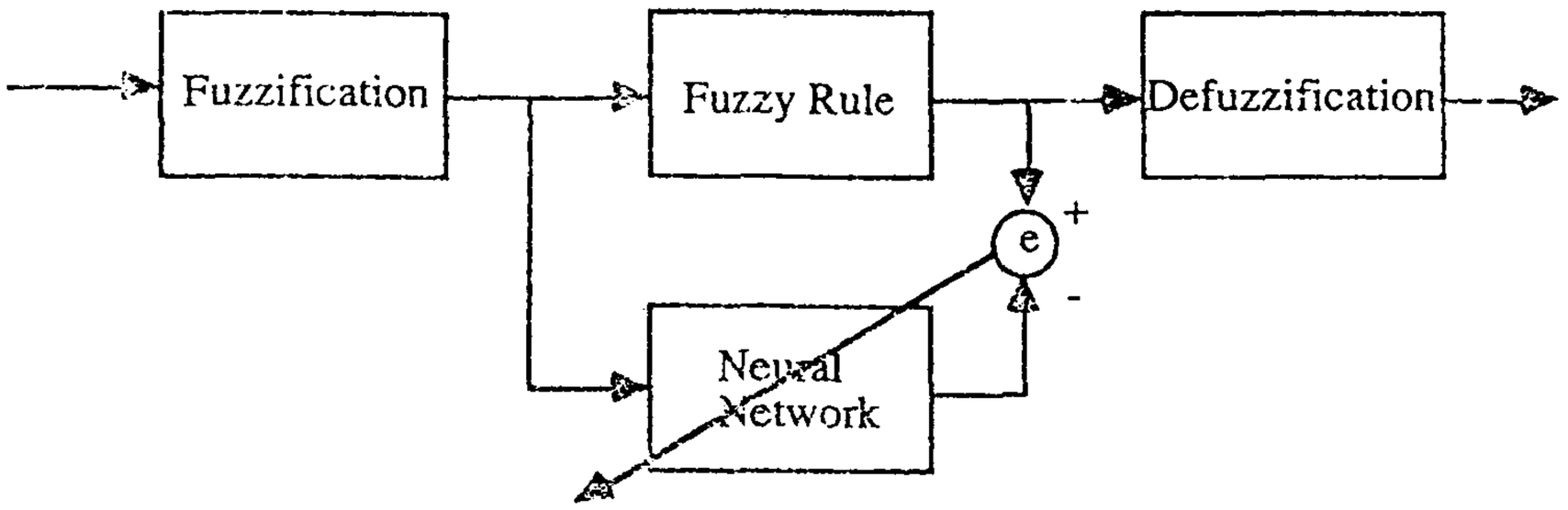


그림 4.9: 초기 퍼지 규칙을 배우는 뉴로-퍼지망

계 변수 들을 학습시킴으로써 학습 속도가 상당히 향상되며, 더 정확한 동정화의 결과를 줄 뿐만 아니라, 최적의 소속 함수(Membership function) 모양과 중심점을 찾을 수가 있다[18].

또한, 제안한 뉴로퍼지 구조를 이용하여 주어진 시스템에 대한 퍼지 규칙을 미리 알고 있는 경우에 다음 그림 4.9와 같이 퍼지 규칙을 배우고, 이후에 실제 시스템에 인가할때 발생되는 오차를 그림 4.10과 같이 보정할 수도 있다.

제안한 뉴로-퍼지 구조의 성능을 검증하기 위해 다음과 같은 비 선형 동적 시스템을 고려한다[18].

모델 1 :

$$y_p(k+1) = \frac{y_p(k)}{1+y_p^2(k)} + u(k) \tag{4.20}$$

모델 2 :

$$y_p(k+1) = \frac{y_p(k)}{1+y_p^2(k)} + u^3(k) \tag{4.21}$$

위의 모델들은 입력과 시스템 상태 사이의 비선형성에 근거하여 선택하였다. 모델 1은 시스템 출력의 지연값에만 비선형 특성이 있고, 모델 2는 입력과 출력에 비선형 특성이 있지만 선형 결합되어 있는 경우이다.

동정화를 위한 신경망 블럭의 은닉층의 수는 2개를 사용하였고, 학습을 위해서 [-1.5,1.5] 사이의 랜덤 입력 4,000를 사용하였다. 퍼지화를 위한 입력측 기준 퍼

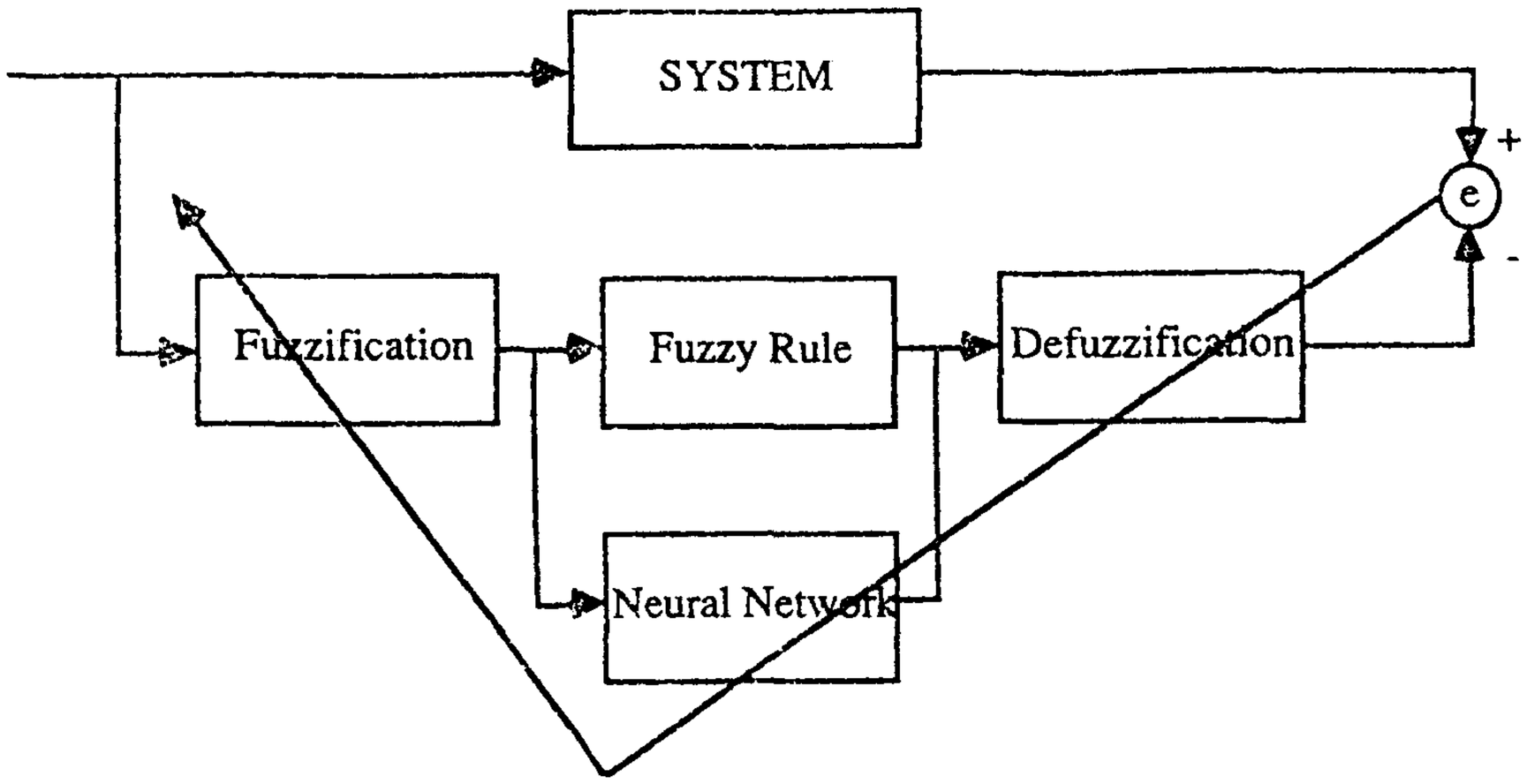
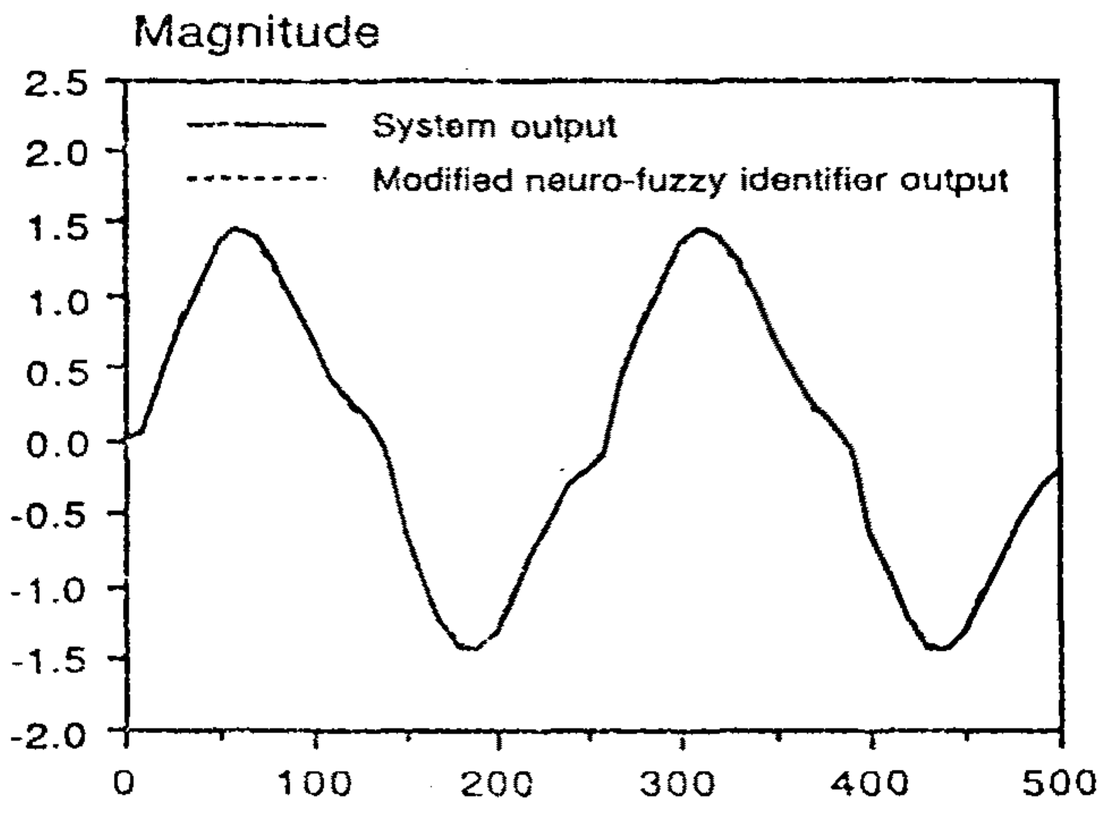


그림 4.10: 수정된 뉴로-퍼지망을 이용한 초기 퍼지 규칙의 오차 보정

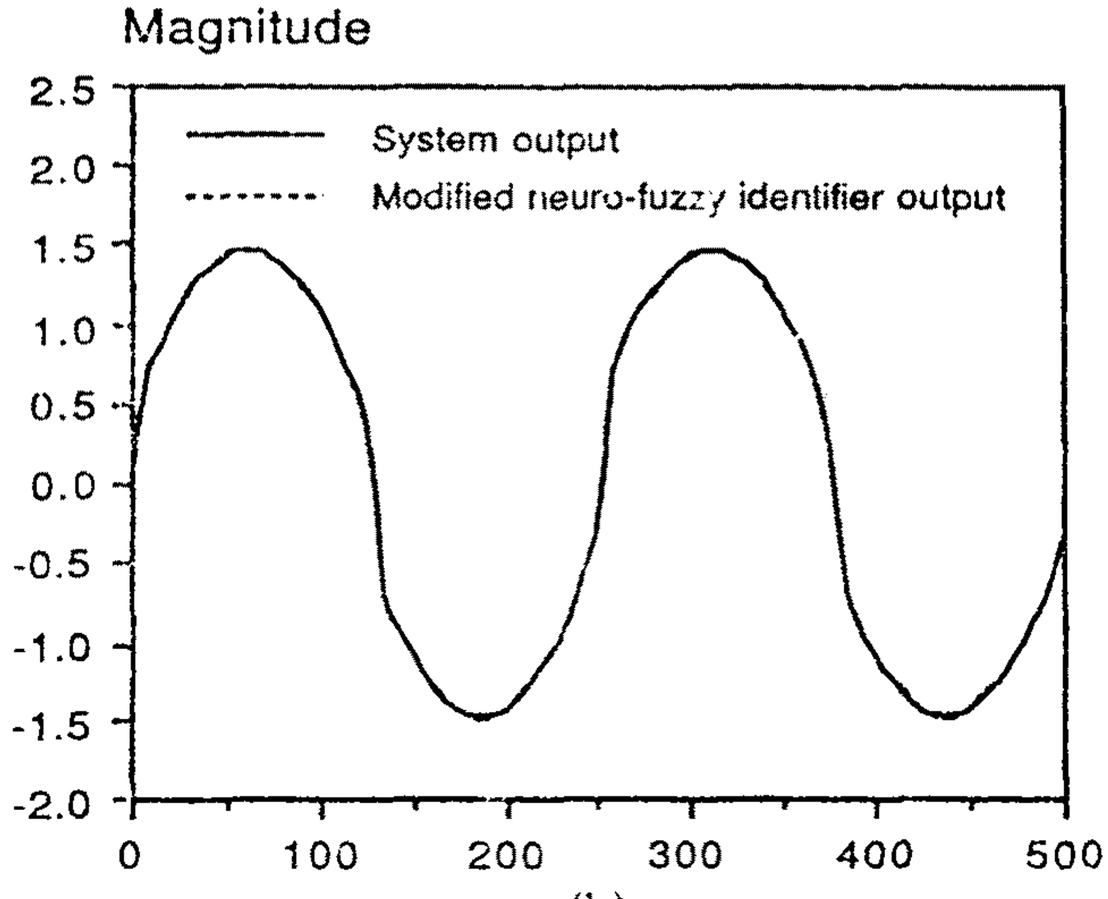
지 집합의 초기 평균값으로는 -1.5, -0.75, 0, 0.75, 1.5의 값을 이용하였고, 분산이 0.2인 가우시안(Gaussian)함수를 사용하며, 출력측의 가상 퍼지 소속 함수의 초기 평균값은 -2, -1, 0, 1, 2를 이용하였다. 그림 4.11은 학습이 끝난후 테스트 입력으로 $\sin(2pik/250)$ 을 이용했을때의 동정화 결과이다. 실선은 시스템의 실제 출력을 나타내고, 점선은 뉴로-퍼지 동정화 모델의 출력을 나타낸다.

4.3 유전 알고리즘의 신경망에의 도입

다층구조 신경망 모델의 학습을 위해, 많이 이용되는 오차 역전파 학습 알고리즘은 지역 최소점의 문제점이 존재하며, 또한 주어진 문제를 해결하는데 적합한 은닉층 뉴런의 수를 정하기가 어렵다. 또한 시그모이드(Sigmoid) 함수의 미분에 의해 얻어지는 함수들을 신경망 모델의 뉴런 전달 함수로 이용할때 기존의 최대 급경사 방법(Steepest descent method)에 기반한 오차 역전달 방법은 지역 최소점의 문제가 더욱 심각하다. 이러한 문제점들을 해결하기 위한 방법으로 통계적 방법의 일종인 유전 알고리즘을 이용할 수가 있다[9].



(a)



(b)

그림 4.11: 수정된 뉴로-퍼지 모델을 이용한 비선형 동적 시스템의 동정

본 연구 과제에서는 신경망 학습에 유전 알고리즘을 이용할 수 있는 가능성을 검토하기 위해 신경망의 연결 강도값들로 양자화된 값만을 가진다고 가정하고, 이런 조건하에 영문자 알파벳을 유전 알고리즘으로 학습하는 방법들을 살펴 본다. 유전 알고리즘을 이용하여 N개의 입력 뉴런과 L개의 은닉층 뉴런, 그리고 M개의 출력 뉴런으로 구성된 신경망을 이용하여 학습하는데에는 N*M*L개의 가변 요소들을 최적화해야 한다. 해결해야 할 입출력 뉴런들의 수가 늘어남에 따라 최적화해야 하는 변수들의 수는 급격히 늘어남으로 기존의 유전알고리즘을 이용하여 그 해를 구한다는 것은 매우 큰 어려움이 있다. 이런점을 보완하기 위해 기존의 신경망이 연결 강도들을 학습시키는 방법을 이용하는 것에 비해 은닉층 뉴런의 최적값을 유전 알고리즘으로 학습시키고, 이후에 연결 강도들을 최적화된 은닉 뉴런들의 패턴과 학습해야 할 입출력 뉴런들의 패턴들의 Hebbian 학습에 의해 구할 수가 있다. 이렇게 하면 앞에서 설명한 같은 구조의 신경망에 P개의 패턴을 저장시킬 경우 (N+M+L)*P 개의 가변 요소들만 최적화 시키면 되므로 유전 알고리즘을 이용하여 보다 쉽게 최적화시킬 수가 있다. 한편, 신경망의 하드 웨어 구현시에 아날로그 값을 이용하는 신경망을 만드는 것보다 이진 값만을 갖는 디지탈 신경망을 구현하기가 훨씬 쉽고 많은 장점을 가진다고 할 수가 있다. 이를 위해서는 신경망 학습이 끝난 상태에서 모든 결과들이 이진수값을 이용할 수 있어야하는데 기존의 오차 역전달 학습 알고리즘으로는 이를 구할 수가 없다. 따라서, 본 과제는 신경망의 하드웨어 구현에 적합하며 기존의 유전 알고리즘을 신경망 학습에 이용할 수 있도록 하는 방법을 연구한다. 또한, 여러가지 유전 알고리즘의 변수들을 변화시키므로써 그에 따른 신경망의 학습속도에 대한 성능을 비교 검토해 본다.

우선 유전 알고리즘의 정보 처리 과정을 간단히 살펴 보면 다음과 같다.

1) 유전정보를 가진 많은 chromosome을 초기화한다. 이때는 random 함수를 사용하여 초기화한다.

2) 이들 생성된 chromosome으로부터 children을 생성하기 위한 parents 유전자를 선택한다. 이때 목적 함수에 적합한 유전자가 선택되 확률이 높도록한다.

3) parents의 유전정보를 서로 crossover를 한다.

4) 약간의 돌연변이(mutation)을 생기도록 한다.

5) 2)-4) 과정을 반복한다.

위와 같은 유전알고리즘을 이용한 하드 웨어 구현에 적합한 신경망 모델을 학습시키는 알고리즘은 다음과 같다.

1) 입력과 출력의 값을 결정한다.

2) 은닉층의 값을 초기화한다.

3) 유전 알고리즘을 통해 은닉층의 값을 결정한다.

4) 입력에 의한 출력값을 계산하다.

5) 미리 정의된 오차를 계산한다. 오차는 출력값과 실제값과의 틀린 비트수를 오차로 정의한다.

6) 유전알고리즘에 의해 생성된 은닉층값을 평가한다.

7) 3)-7)까지의 학습을 반복한다.

위에서 설명한 방법을 이용하여 영문자 알파벳 패턴들에 대해 유전 알고리즘을 이용한 학습 성능에 대한 시뮬레이션 결과는 다음 그림 4.12와 4.13에 나타내어져 있다.

그림 4.12는 Population을 40개로 하고, Fitness 함수를 선형 함수로서 Population의 우선 순위에 의해 정해지고, 돌연 변이율을 0.1%로 고정시킨 경우의 시뮬레이션 결과이다. "o"이 있는 실선은 패턴 6개를 학습시켰을때 학습 횟수에 따른 오차값을 의미하고, 실선은 패턴 4개를 학습시킨 경우이다. 그림 4.13은 Population을 40개로 하고, Fitness 함수를 선형 함수로서 Population의 우선 순위에 의해 정해지고, 지역 최소점에서의 학습향상을 위해 돌연 변이율을 약 0.1%에서 선형적으로 2% 까지 증가 시킨경우의 시뮬레이션 결과이다. "o"이 있는 실선은 패턴 6 개를 학습시켰을때 학습 횟수에 따른 오차값을 의미하고, 실선은 패턴 4

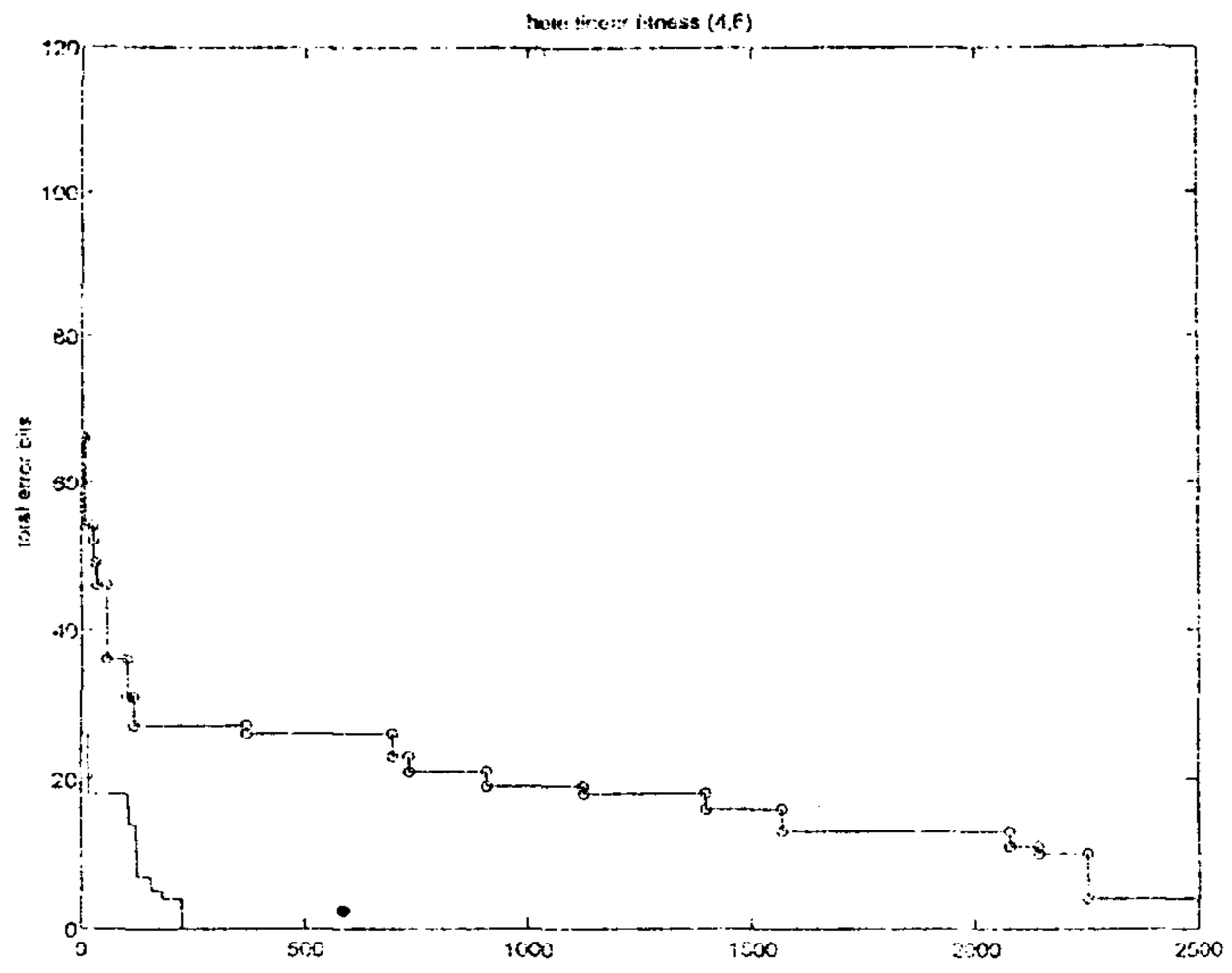


그림 4.12: 영문자 알파벳의 학습 성능(선형 fitness 함수, 0.1% 돌연변이율을 이용한 경우)

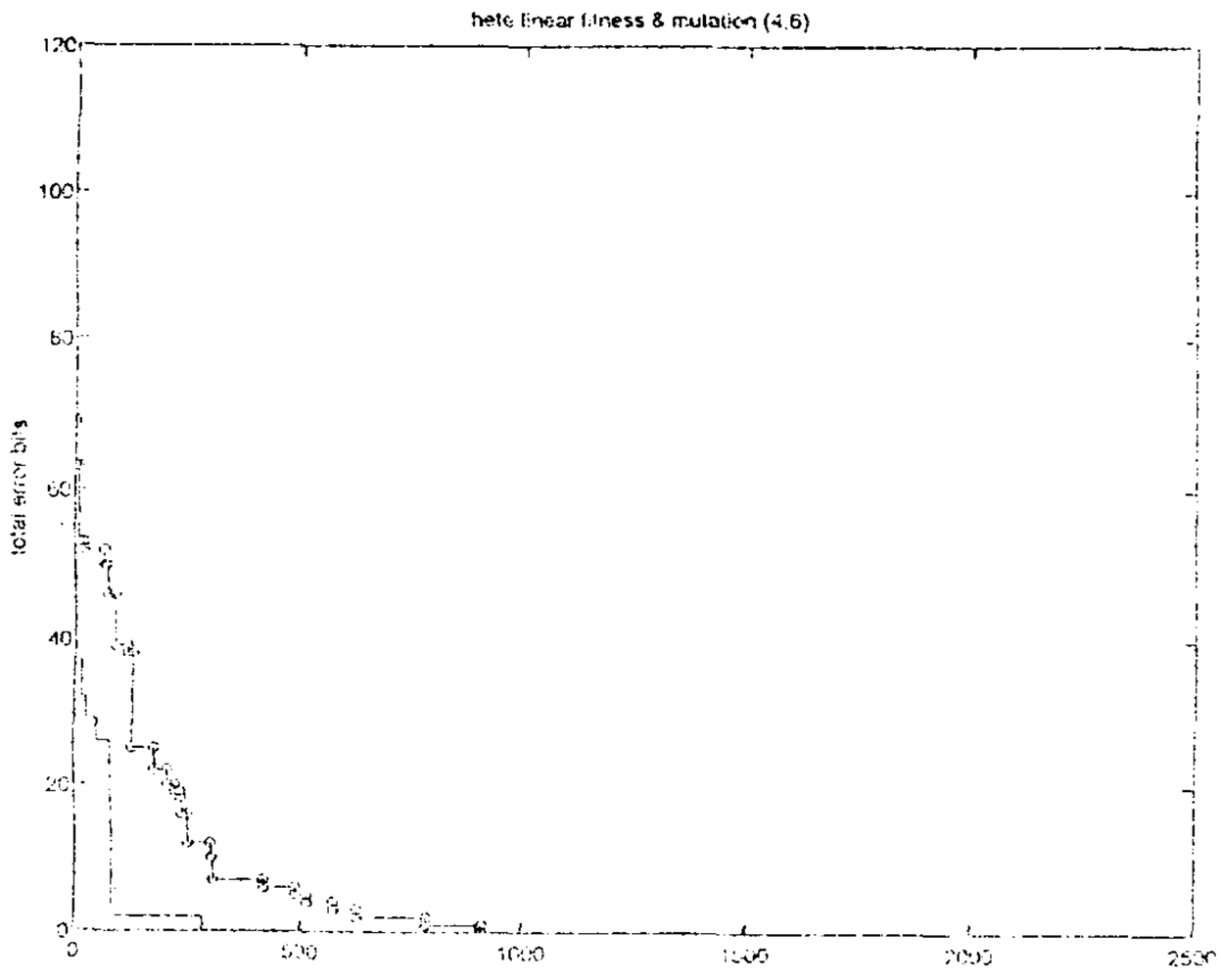


그림 4.13: 영문자 알파벳의 학습 성능(선형 fitness 함수, 선형적으로 증가하는 돌연변이율을 이용한 경우)

개를 학습시킨 경우이다. Fitness 함수를 선형 함수로 하고 돌연 변이율을 선형적으로 증가 시킨 경우 Fitness 함수를 지수 함수로 하고, 돌연변이율을 일정하게 한 경우보다 지역 최소점에서 잘 빠져 나온다는 것을 알 수가 있다. 왜냐하면, Fitness 함수를 지수 함수로 하고 돌연 변이율을 일정하게 한 경우 처음 학습과정에서 좋은 형질의 유전자가 많이 선택되어 학습이 빨리되나 이는 이상 학습이 되었을 경우, 같은 유전자들이 선택될 경우가 많아져 지역 최소점에 빠지게 되어 더 이상 학습이 되지 않을 수 있다. 그림에서 보듯이 학습 패턴이 6개인 경우, 돌연 변이율을 학습이 되어감에 따라 증가시킴으로써 돌연변이에 의한 학습을 시킴으로써 지역 최소점에서 더 잘 빠져 나와 학습이 이루어지는 것을 알 수 있다.

여 백

5 장

회전기기 고장진단 및 수명 예측

5.1 개요

회전기계의 진단은 전문가의 경험에 바탕을 둔 의견에 전적으로 의존하는 경우가 대부분이었으며, 실제 사람이 진단도구의 도움으로 진단하였을 경우에도 비교적 신뢰도 있는 진단이 가능하다. 그러나 회전기의 진단의 경우 과학적인 분석을 통하여 정확한 진단이 가능할 뿐만 아니라 소수의 전문가가 플랜트 전체의 시스템을 관리한다는 것은 한계가 있으므로 진단시스템에 의한 진단이 필요한 것이다.

이장에서는 시스템의 진단을 위한 진동에 관한 기초이론과 이상진동 현상에 관하여 알아보고 회전기의 이상진동을 측정하기 위한 자료값을 측정하는 진동, 가속도 등의 센서에 관하여 살펴본다. 그리고 신경망을 이용하여 시스템의 이상현상을 진단하는 방법에 관하여 알아본다.

5.2절에서는 진동에 대한 기본적인 이론들을 살펴본다. 5.3절에서는 진동을 측정하는 센서들에 관하여 살펴보고, 5.4절에서는 이상진동에 관하여 알아보고, 5.5절에서는 신경망을 이용하여 이상진단에 적용하는 방법에 관하여 알아보기로 한다.

5.2 진동이론

회전기의 진단 시스템을 위한 자료값들은 이미 기술한 바와 같이 회전기에 부착되어 있는 진동, 속도, 가속도 등을 측정하는 센서의 출력값에 의존한다. 센서로부터의 입력값을 필터링시켜 그 파형에 따라 시스템의 상태를 진단하고 앞으로의 상태

를 미리 예측하는 것이다. 따라서 시스템 진단및 예측의 핵심적인 역할을 하는 진동에 관한 이론들을 잘 살펴볼 필요가 있다.

단순 진동자

진동자의 운동이 단일축 방향에서만 이루어지는 단일 자유도 선형 단순 진동자는 그림5.1과 같이 일반화해서 나타낼 수 있다.

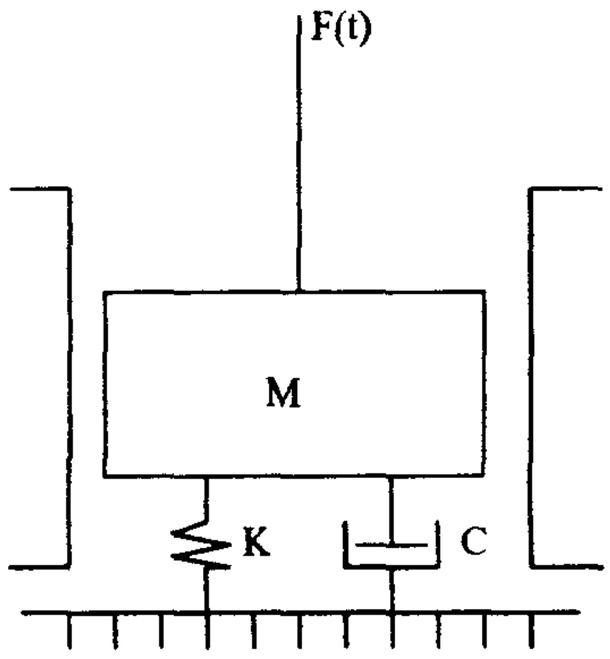


그림 5.1: 단순 진동자 모델

이 그림에서 K는 스프링 상수, C는 점성 마찰계수(viscous friction coefficient), M은 진동자의 질량이고, x는 평행위치로 부터의 변위이다. 이러한 단순 진동자에 외력 $F(t)$가 가해질 때 진동자의 운동은 다음식에 의해서 설명된다.

$$M\frac{d^2x}{dt^2} + C\frac{dx}{dt} + Kx = F(t) \tag{5.1}$$

외력 $F(t)$가 0이 아닌 경우는 강제진동으로서 이 경우는 뒤에서 별도로 다루고 먼저 $F(t)$가 0인 자유진동의 경우에 대해서 기술하고자 한다.

자유진동

자유진동은 식5.1에서 $F(t) = 0$ 인 경우로서 초기치에 대한 응답을 말한다. 이 때 식은 다음과 같이 된다.

$$M\frac{d^2x}{dt^2} + C\frac{dx}{dt} + Kx = 0 \tag{5.2}$$

식5.2를 Laplace변환을 거치면

$$Ms^2X(s) - Msx(0) - Mx'(0) + CsX(s) - Cx(0) + KX(s) = 0$$

가 된다. 따라서

$$X(s) = \frac{As + B}{s^2 + \frac{C}{N}s + \frac{K}{M}} \tag{5.3}$$

$$= \frac{As + B}{s^2 + 2\zeta w_n s + w_n^2} \tag{5.4}$$

여기서

$$A = x(0) \tag{5.5}$$

$$B = \frac{C}{M}x(0) + x'(0) \tag{5.6}$$

$$w_n = \sqrt{\frac{K}{M}} \tag{5.7}$$

$$\zeta = \frac{1}{2w_n}\frac{C}{M} = \frac{C}{2\sqrt{MK}} \tag{5.8}$$

이다. 이때 ζ는 댐핑비(damping ratio), w_n는 자연 주파수(natural frequency)라고 부른다. 분모가 0이 되는 점, 즉

$$s^2 + 2\zeta w_n s + w_n^2 = 0 \tag{5.9}$$

의 해를 극(pole)이라고 한다. 식5.9의 해, 즉 극은 다음과 같이 주어지며

$$s_1,\ s_2 = -\zeta w_n s \pm w_n\sqrt{\zeta^2 - 1}$$

ζ에 따라서 실수가 될 수도 있고, 공액복소수가 될 수도 있다.

강제진동

앞에서 기술한 자유진동은 시스템을 평형 위치(equilibrium point)로부터 벗어나

게 했다가 놓아둘 때 발생한다. 그러나 많은 실제의 진동 문제에서는 시스템에 외력이 가해지며 그때 시스템의 응답(response)을 결정하는 것이 시스템을 이해하는데 중요한 첫 단계가 된다. 이와 같이 외력이 가해진 경우를 강제 진동이라고 부르며, 이 경우에 외력의 주파수가 시스템의 고유 진동수와 일치하여 공진이 일어나서 진동의 진폭이 크게 증가될 수 있다. 강제진동 문제의 실질적인 중요성에 비추어 선형운동을 하는 단순진동 시스템을 대상으로 하여서 그의 특성을 자세히 살펴보기로 한다.

강제진동 시스템의 운동 특성

그림2-1과 같은 시스템에 외력 $F(t)$가 가해질 때 진동자의 운동은 다음식에 의해서 설명된다.

1. 댐핑이 0인 경우에 $(Q = \infty)$ 외력의 주파수 f가 공진 주파수 f_0보다 작을 때는 시스템의 응답과 외력은 동일 위상이고 공진주파수보다 클때는 $180°$위상차가 생긴다. 이 두 경우 사이의 변화는 공진 주파수에서 불연속적으로 일어난다.

2. 댐핑이 있을 때 $(Q \neq \infty)$ 는 없을 때와는 달리 공진 주파수 부근에서의 위상지연은 불연속적이지 않고 점진적으로 일어난다. Q가 작아 질수록 즉 댐핑이 커질수록 공진 주파수 부근에서의 기울기는 작아진다.

3. 댐핑의 크기에 관계없이 공진주파수에서의 시스템 응답은 가해진 힘에 대해서 $90°$의 위상지연을 갖는다.

5.3 진동센서

각종 진동문제를 해결하고 운전상태의 감시 및 점검등을 위해서 현장에서 많이 쓰이게 되는 진동측정용 센서는, 역학적인 움직임을 정량적으로 나타내기 위하여 그 단위상의 크기에 비례하는 전기적인 신호를 발생시키는 것이다. 이를 위하여는 역

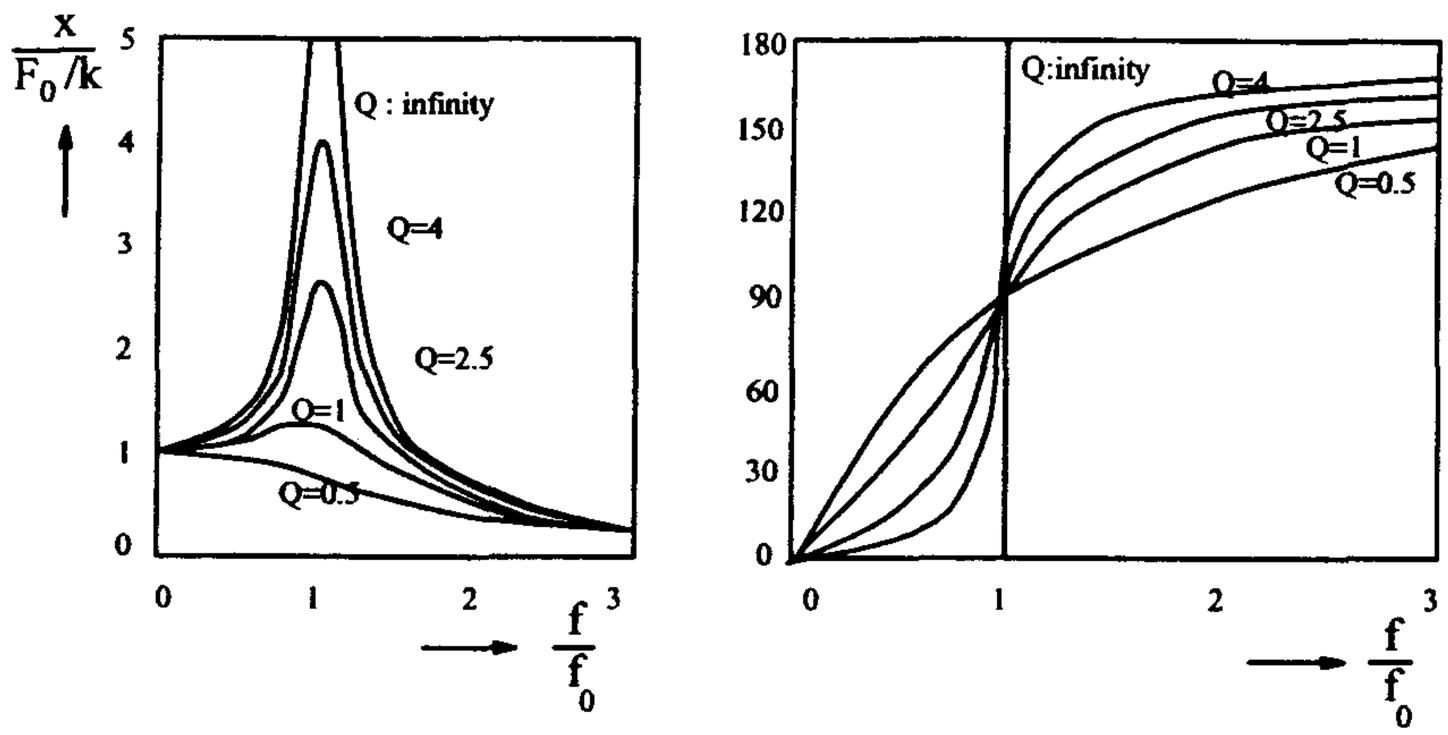


그림 5.2: Q값에 따른 강제진동의 진폭과 위상지연

학적인 진동과 전기적인 출력 사이에 정비례성이 뛰어나고 그 동작범위가 사용용도에 적합해야할 것이다.

이장에서는 진동 측정용 센서의 종류와 각 센서의 특성에 관하여 간단히 살펴보기로 한다.

진동 센서의 종류

진동센서는 크게 접촉형과 비접촉형으로 나눌 수 있고, 측정 파라미터에 의해서도 구분이 될 수 있는데 표5.3에 정리되어 있다.

부착방법	형식	측정 파라미터	주파수범위
접촉형	압전형	가속도	1 ~ 20kHz
	동전형	속도	10 ~ 1kHz
	스트레인게이지형	변위, 가속도	DC(0) ~ 500Hz
비접촉형	와전류형	변위	DC(0) ~ 5kHz
	용량형	변위	DC(0) ~ 100kHz
	전자광학형	변위	DC(0) ~ 100kHz

표 5.1: 진동센서의 종류별 특성

이가운데 회전기계의 진단에는 접촉형으로는 압전형, 비접촉형으로는 와전류형이 많이 이용되고 있다.

압전형 가속도 센서

압전형 물질(piezoelectric material)은 스트레스를 받으면 그 크기에 비례하는 전하량을 발생시키는 성질이 있다. 스트레스란 곧 가속도에 비례하는 양이므로 전하량을 측정하면 진동 가속도를 알 수 있는 것이다. 이는 온도나 습도, 외부 유도의 영향을 받을 우려가 있는 반면 소형, 경량이며 측정범위가 넓은 장점을 갖고 있어 보편적으로 가장 널리 쓰인다.

인덕턴스형 센서

인덕턴스형 센서는 어떤 폐회로내에서 자속이 변하면 단위시간당 자속변화율에 비례하는 기전력이 생기는 원리를 이용한 것이다. 따라서 자계와 코일을 관성으로 분리하면 자속의 변화율 즉 진동의 속도에 비례하는 전압출력을 얻게되는 것이다. 전자형 센서도 있는데 이는 코일내의 철심에 진동을 주어 임피던스의 변화로 변화시키는 방식이다. 이들은 출력이 크고 외부유도의 영향이 적은 장점이 있는 반면 형태가 크고 측정범위가 제한된다.

용량형 센서

축전기는 두 전극 사이의 거리에 따라 그 용량이 변하게 되므로 한 전극을 진동시키면 그 용량의 변화를 통하여 진동의 크기를 측정할 수 있게 된다. 이러한 원리를 사용하므로 피측정체를 하나의 전극으로 이용하면 센서를 피측정체에 접촉시키지 않은 상태로 측정이 가능하다.

5.4 이상진동 현상

회전기계에서는 여러가지 요인에 의해서 진동이 발생하게 되는데 이러한 진동은 크게 외력의 작용 유무에 따라서 강제진동과 자려진동으로 분류할 수 있다. 이들의 특징을 요약한 것이 표 5.4이다.

	강제진동	자려진동
외력	있음	없음
발생진동	회전주파수의 2배, 3배, ···, 1/2배, 1/3배의 성분 발생	기본 주파수만 발생
공진곡선의 재현성	좋음	좋지 못함
진동 예	rotor unbalance 축의 미스얼라인먼트 비선형 진동	베어링의 오일휩 공작기계 떨림 진동 축의 건성마찰

표 5.2: 강제진동과 자려진동의 비교

강제진동

진동계에 외부로부터 주기적으로 변동하는 강제외력이 작용하는 경우 그 강제외력에 의하여 이상진동을 발생시키는 것을 강제 진동이라 말하고 대표적인 이상 현상의 예로는 rotor의 편심, 축의 미스얼라인먼트, 비선형 진동(분수조화 진동), 주기적으로 변화하는 강제외력에 의한 진동(베어링, 기어등의 이상)등이 있다. 이에 관한 이론은 제5.2장에 기술되어 있다. 회전계에서 회전수가 상승함에 따라 회전 주파수(회전주파수의 2배, 3배, ···, 1/2배, 1/3배의 성분도 포함)도 증가한다. 그리하여 위험속도(축의 고유진동수)부근에 각각의 회전주파수의 성분이 있게되면 진동, 진폭은 급격히 증가한다. 이 상태를 공진이라 한다. 강제진동의 주요 특징은 첫째, 진동 및 공진 곡선의 재현성(repeatability)이 좋고 둘째, 진동의 지속성이 양호하고 셋째, 진동을 멈추면 곧 정상진동으로 돌아가 과도적 상태가 짧으며, 마지막으로 진동을 멈추기위해 비교적 큰 힘이 필요하다는 점이다.

자려진동

자려진동은 강제 외력이나 축의 회전주파수(강제외력의 주기)에 관계없이 축 자신의 위험속도에 의해서 세차운동을 일으키는 진동이다. 대표적인 이상현상의 예로는 미끄럼 베어링의 오일휩(oil whip), 공작기계의 떨림진동, 축의 건성마찰등이 자려진동이다. 이것은 고속회전기계 특유의 불안정현상이고 현상에 따라 발생주파수나 진동회전의 방향등이 달라지므로 진단에는 현상에 대한 지식을 요한다.

자려진동의 특징은 첫째, 재현성이 나빠 그때마다 이상한 데이타가 되기도 하고, 같은 조건하에서 발생하기도 하고, 발생하지 않을 수도 있다. 둘째, 정상상태가 되

면 천천히 진폭이 증감한다. 셋째, 약간의 힘을 가하는 것으로 진동이 멈춘다. 끝으로 일정한 축 회전수이상에 다다르면 거의 진폭변화가 없는 진동이 계속된다.

강제진동과 자려진동의 특징비교

회전수에 대한 상대 진폭 그래프를 그려 비교해보면 강제진동의 경우와 자려진동의 경우 그 모양이 크게 다르다는 것을 알 수가 있다. 강제진동의 경우 기계가 회전하게 되면 회전주파수에 해당하는 기본진동(fundamental component) 뿐만 아니라 회전 주파수의 2배 되는 2차진동, 1/2배되는 1/2차 진동도 따라서 발생한다. 따라서, 기본진동, 2차진동, 1/2차진동 3개중 어느 하나라도 기계의 공진주파수에 가깝게 되면 진폭이 커진다. 그러나, 기본진동의 주파수가 공진주파수에 근접할 때 진폭이 가장 커진다. 이것이 그림5.3에 나타나 있다.

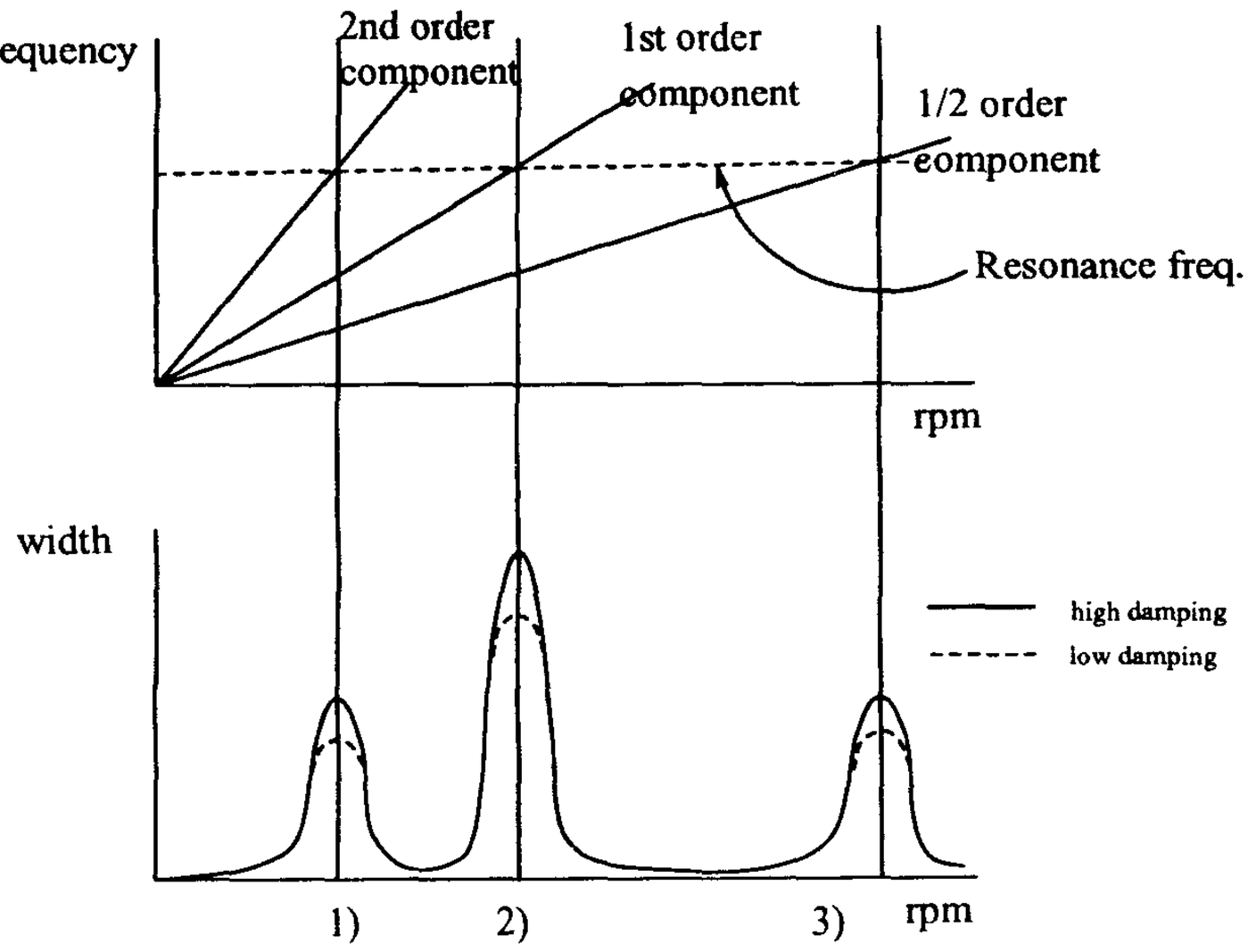


그림 5.3: 진폭-회전수 그래프(강제진동의 경우)

자려진동의 경우(그림5.4)는 축의 회전주파수와 무관하게 어느 회전주파수 이상이면 진폭이 커지게 되는데 이때 커진 상태를 그대로 유지하다가 일정한 축회전수에 다다르면, 거의 진폭의 변화가 없는 진동이 계속된다.

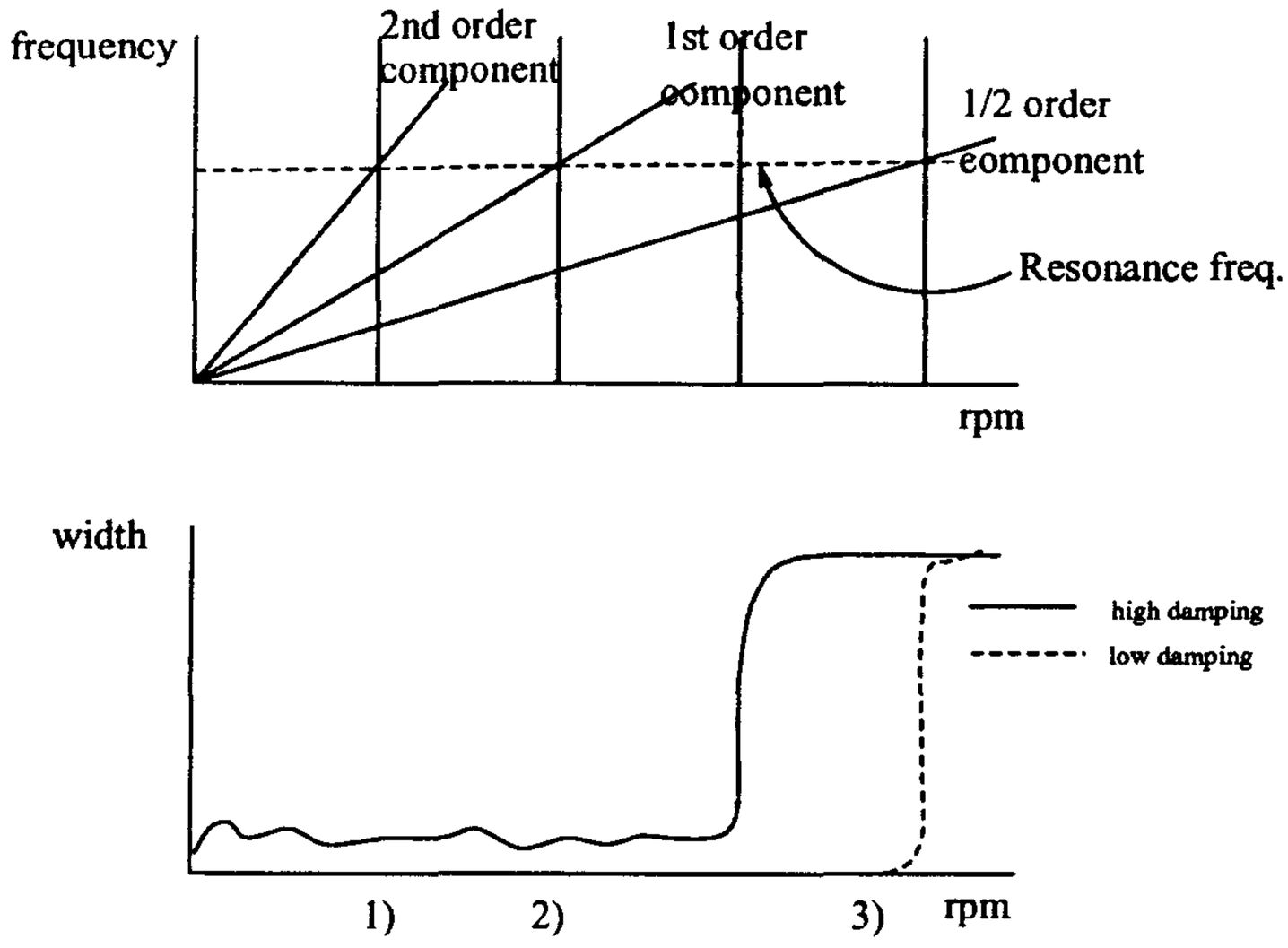


그림 5.4: 진폭-회전수 그래프(자려진동의 경우)

진동의 측정

기계 진동이란 기계의 운동 또는 변위를 표시하는 양의 크기가 어떤 평균치 혹은 기준치보다 크거나 작은 상태를 교대로 반복하는 시간적 변화를 말한다. 그 단위 변화(1왕복)에 필요한 시간을 주기(T)라고 하고, 1초동안 왕복한 횟수를 주파수(진동수, f)라 한다. 진동의 크기를 표시하는 것으로 평균치(average), 실효치(rms), 피크치(peak)등이 있으며, $x(t) = D\sin wt$일때 다음의 관계식이 성립한다.

$$average\ x_{av} = \frac{1}{T}\int_0^T |x(t)|dt = \frac{2}{\pi}D = 0.637D$$

$$rms\ x_{rms} = \sqrt{\frac{1}{T}\int_0^T |x(t)|^2 dt} = \frac{D}{\sqrt{2}} = 0.707D$$

$$peak\ x_{peak} = \frac{\pi}{2}x_{av} = \sqrt{2}x_{rms}$$

진동의 측정은 데이타가 대상 시스템의 진동 특성을 올바르게 나타낼 수 있도록

측정의 모든 단계에서 면밀한 검토와 주의가 요구된다. 이를 위해서는 측정장치의 올바른 선택이 중요하며 또한 측정하고자하는 양의 물리적 의미에 대한 이해가 필요하며 측정위치의 선정과 측정절차에 대한 체계적인 이해가 선행되어야 한다.

측정 설비

기계진동을 측정하여 대상 시스템의 이상유무를 비교적 정확하게 감지할 수 있는 대표적인 설비로 펌프(pump), 컴프레서(compressor), 터빈(turbine) 및 롤러 등의 회전 설비가 있다. 또한 회전기계가 아닌 왕복기계등에도 진동을 측정함으로써 설비의 이상을 발견할 수 있는 경우도 있다.

측정 위치

진동을 측정하기 위한 유효한 측정위치는 설비에 따라 각각 다르나 가능한한 진동 발생원에 근접된 위치에서 측정하는 것이 효과적이다. 회전기계에 있어서는 축수상의 직각방향에 진동이 발생되므로 측정하고자 하는 설비의 특성 및 진동의 종류에 따라 수직방향(V), 수평방향(H), 축방향(A)으로 측정하는 것이 보편적이다.

측정 시스템

진동센서중에서 널리 쓰이는 가속도계의 특성은 전하감도와 전압감도를 나누어 진다. 따라서 가속도계의 선택에 따라 pre-amp의 종류가 결정되어진다. 압전센서를 사용한 진동측정시스템은 기본적으로 그림5.5와 같은 구조를 가지고 있다.

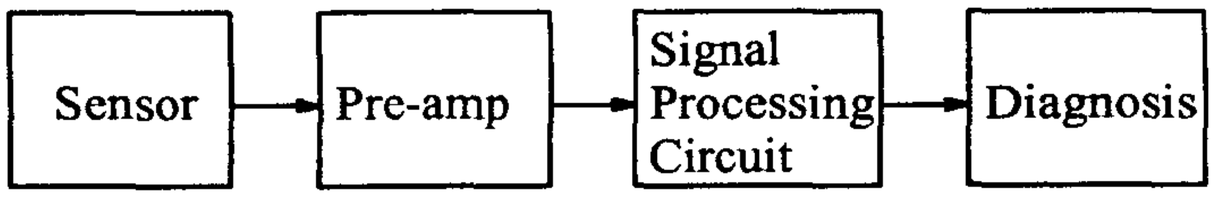


그림 5.5: 진동측정시스템

주파수 영역에 따른 진동

회전기계에서 발생되는 진동의 주파수 영역을 분류하면 다음과 같이 크게 3가지 영

역으로 구분할 수 있다.

- 저주파수 영역 : 회전주파수의 5배 정도
- 중간주파수 영역 : 회전주파수의 5배 ~ 1KHz정도
- 고주파수 영역 : 1KHz이상

회전기계의 이상형태와 그 이상형태에 기인하여 발생하는 진동의 대표적인 예를 표5.3에 나타내었다.

주파수 영역	이상형태	발생주파수	진동방향
저주파수 영역	- 회전체의 편심	f_0	radial
	- 축의 미스얼라인먼트	$2f_0, 3f_0$	axial
	- 축의 휨	$2f_0, 3f_0$	axial
	- 볼트 풀림	$1/3f_0, 1/2f_0$	radial
	- 오일 휩(whip)	$(0.4 \sim 0.45)f_0$	radial
중간 주파수 영역	- 기어 이상		thrust
	o 전체적마모, 편접촉, 치형오차	~	radial
	o 피치 오차, 편심		
고주파수 영역	- 롤링 베어링의 이상		thrust
	o 내륜 결함		radial
	o 외륜 결함	1KHz	
	o 전동체 결함		
	- 미끄럼 베어링		

표 5.3: 이상종류와 발생 주파수

진동의 측정 파라미터

회전기계의 이상진단에 있어서 진동을 측정할 수 있는 파라미터로 변위, 속도, 가속도등을 이용하는데, 결함의 종류에 따라 어떤 파라미터를 이용할 것인가가 매우 중요하다. 1KHz 이하의 저주파 대역에서는 속도를 사용하는 것이 유리하고, 1KHz 이상의 고주파 대역에서는 가속도를 사용하는 것이 좋다. 또한 미끄럼 베어링으로 지지되는 고속회전기계나 공작기계등의 진단에 있어서는 변위의 사용이 필요 불가결하다.

측정 파라미터	이상의 종류	이상예
변위	변위량 또는 움직임의 크기 자체가 문제가 되는 이상	공작기계의 위축현상, 회전축의 편차회전
속도	진동에너지나 스트레스(stress)가 문제가 되는 이상	회전기계의 진동
가속도	충격 등과 같이 크기가 문제가 되는 이상	베어링의 결함 진동, 기어의 결함 진동

표 5.4: 이상의 종류별 측정 파라미터

회전기계의 이상현상

기계진동에는 저주파에서 고주파까지 여러가지 진동이 존재한다. 따라서 회전기계 설비에서 발생하는 진동도 마찬가지로 광범위한 주파수 성분을 갖고 있다. 그런데 종래에는 회전기계에서 발생하는 진동을 관리하는 경우 회전주파수의 수배정도까지의 진동밖에 관리하지 않았다. 그러나 회전기계의 이상현상중에는 러너(runner)의 접촉등에 의한 충격진동등도 있으며 이와같은 현상은 종래의 저주파 진동으로는 관측되기 어렵다. 따라서 회전기계의 진단에 있어서 더욱더 많은 열화정보를 얻어내기 위해서는 넓은 주파수 대역에서 진동을 관측할 필요가 있다.

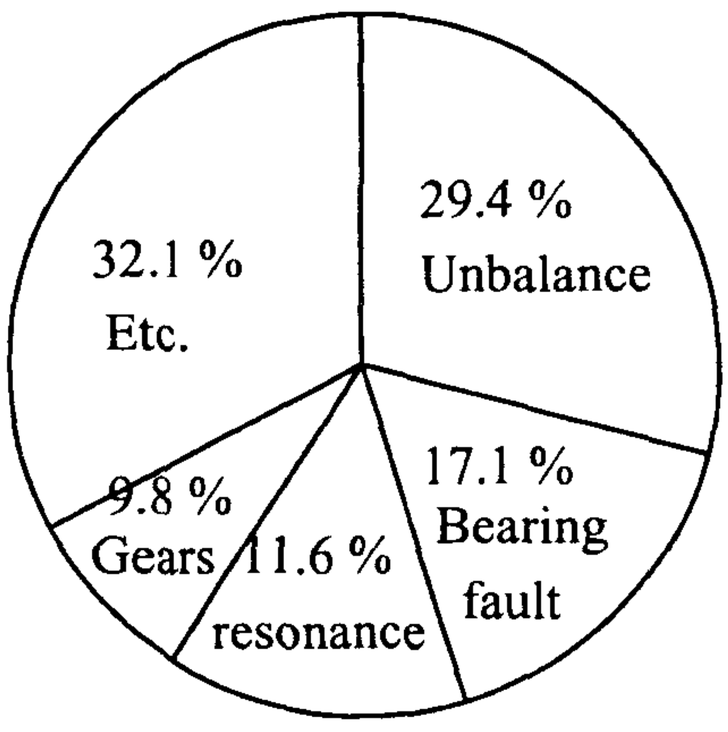


그림 5.6: 진동의 주요 발생원인과 비율

그림 5.6는 각종 이상 현상이 현장에서 어느 정도의 비율로 발생하고 있는가를 보여준다. 이 것은 업종, 기업간에 따라 상당한 차이가 있다고 생각되지만 이 그림은 철강업에 있어서의 비율을 나타내는 것이다. 이 그림에서 알 수 있는 바와 같이 회전기계에 발생하고 있는 이상 현상의 상당부분이 편심(unbalance), 미스얼라인먼트(misalignment), 볼트풀림(bolt loosening)등의 저주파 이상 진동이지만 고

주파이상도 상당량이 된다.

5.5 신경망을 이용한 진단과 예측

대상 시스템의 동작의 정확한 예측을 위해서는 시스템 상태의 올바른 진단이 선행되어야 한다.

회전기의 진단에는 기존에 이용되어 오던 통계적인 방법을 이용한 모델이 있고, 최근에 여러분야에서 성공적으로 적용되어 각광을 받고 있는 신경망을 이용한 방법 등이 있을 것이다. 두가지 방법이 서로 배타적인 것이 아니므로 두가지 방법을 동시에 적용하여 성능향상도 가능하다. 이에 관해서는 [1]에 회전기의 이상진단에 관하여 진단하는 시스템을 성공적으로 연구하고 상품화한 보고가 있다.

신경망을 이용하여 진단시스템을 구현하는 과정을 간단히 설명하자면 다음과 같다. 회전기계에 이상이 있으면 이상진동이 발생한다. 이것을 진동센서를 통해 신호를 받아 스펙트럼을 구해보면 이상의 종류에 따라 특정한 주파수 성분이 다른 것들에 비해 커짐을 알 수 있다. 이 스펙트럼은 각 이상 원인에 따른 특징을 보유하고 있으므로 이것으로부터 신경망의 입력으로 쓰일 특징벡터(feature vector)를 추출해 낸다. 이 특징벡터와 적당히 정해준 출력벡터를 벡터쌍으로 잡아 backpropagation 신경망(BPN)으로 학습시킨다. 그리고 미지의 이상으로 인한 진동스펙트럼으로부터 만든 특징벡터를 신경망의 입력에 가하면 그 특징벡터가 속해 있을 class에 해당하는 값이 신경망으로부터 출력이 되므로 이것으로써 이상원인을 분석할 수 있다. 이 방법에 의하면 사람이 스펙트럼을 보아서 식별하기 어려운 경우도 신경망 자체의 수렴성에 의해 분별을 하므로 보다 과학적인 이상진단이 가능하다.

앞에서(표5.3) 살펴본 바와 같이 파워 스펙트럼의 주파수대에 따라서 회전기계의 이상을 크게 몇가지로 나눌 수 있으며 또 각각이 그 나름의 특징을 가지고 있음을 알 수 있다. 이를 바탕으로 사람의 감각으로는 판단하기 어려운 이상현상도 정확한 데이타를 바탕으로 학습한 신경망이 구별할 수 있으며 보다나은 진단이 가능한 것이다.

따라서 위와같은 각각의 이상현상에 대하여 이를 진단하는 신경망을 이용한 접

근이 가능하고 이를 바탕으로 시스템의 동작을 예측하는 시스템을 구현한다면 보다 나은 예측 시스템을 구현할 수 있을 것이다.

6 장

신경망 모델 예측 시뮬레이터

신경망 예측 모델 시뮬레이터는 신경망 모델 script 화일을 가지고 모델을 정의하며 학습패턴 전처리, 학습상태 및 예측 결과에 대한 그래픽 사용자 인터페이스를 제공한다. 본 신경망 예측 모델 시뮬레이터는 유닉스 운영체제(Unix Operating System)에서 X-윈도우를 기반으로 하며 OSF Motif 환경에서 개발되고 있다.

6.1 시뮬레이터 구성

신경망 예측 모델 시뮬레이터 구성은 모델 설정(setup), 학습 알고리즘 선택, 학습패턴 전처리 및 예측 실행 등으로 구성된다. 그림 6.1은 시뮬레이터 화면을 보여준다. 시뮬레이터 화면은 사용자 명령어 메뉴 바, error 그래프, 학습패턴 error 바, 예측결과 그래프 및 동작상태 설명 바등으로 구성 되어 있다.

사용자 메뉴

본 신경망 예측 모델 시뮬레이터에서 사용되는 메뉴는 pull-down 및 blind 메뉴를 지원한다. Pull-down 메뉴는 그림 6.1에서 보는 것과 같이 화면의 상단에 위치한 것으로 시뮬레이터의 모든 기능을 제공한다. 이를 구성하기 위해서는 다음과 같은 자료구조를 구성시킨다.

```
typedef struct top_menu_struct {
char         *label;   /* name of the button */
char          mnemonic;/* mnemonic; NULL if none */
```

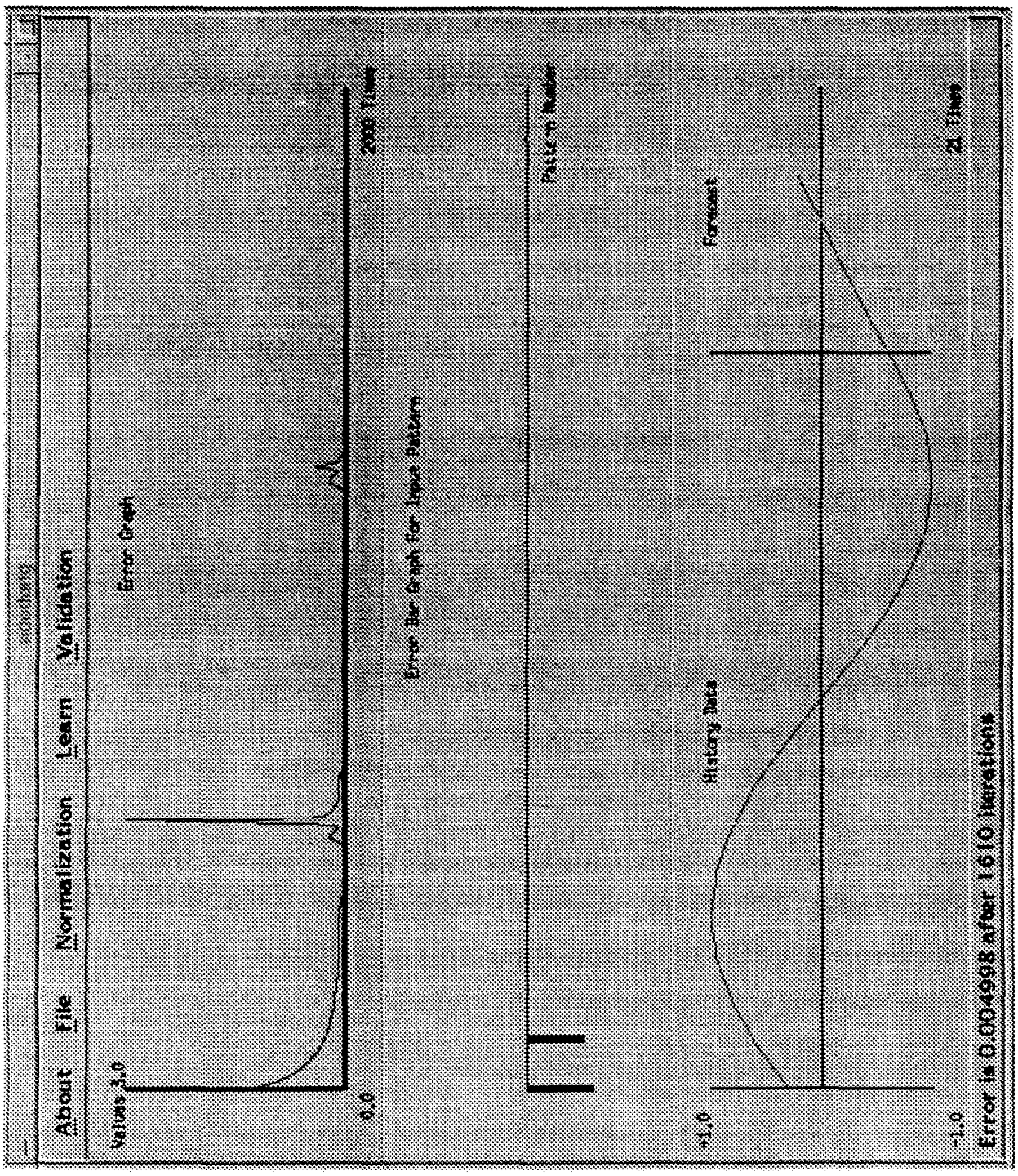


그림 6.1: 시뮬레이터 화면

```
menu_struct *subitems;/* pullright menu items, if not NULL */
} top_menu_struct;
```

여기서 label은 버턴의 이름, mnemonic는 meta key로서 빠른 접근을 위한 것으로 보통 "Alt+해당 key"로서 이용한다. subitems 는 아래에 설명될 부메뉴와 연결을 위한 것으로 menu_struct로 구성된다. 이들 상위 메뉴의 실제 예는 다음과 같다.

```
/* Top-Menu */
static top_menu_struct TopMenu[] = {
  {" About ",   'A', Message    },
  {" File  ",   'F', FileMenu   },
  {" Normalization ",  'N', NormalMenu },
  {" Learn ",   'L', LearnMenu  },
  {" Validation ",  'V', ValMenu    },
  NULL
};
```

위와 같은 모든 메뉴는 부메뉴(cascade submenu)를 가질수 있으며 다음과 같은 자료구조로 구성된다.

```
typedef struct menu_struct {
char        *label;        /* name of the button */
WidgetClass *class;        /* pushbutton, label, separator..*/
char         mnemonic;     /* mnemonic; NULL if none */
char        *accelerator;  /* accelerator; NULL if none */
char        *accel_text;   /* to be converted to compound string */
void       (*callback)();  /* routine to call; NULL if none */
XtPointer    callback_data;/* client_data for the callback */
int          enbl;         /* Initial Flag (Enable/Disable) */
struct menu_struct
    *subitems;     /* pullright menu items, if not NULL */
Widget      *this;         /* Widget Pointer to save */
} menu_struct;
```

class는 WidgetClass를 나타내는 버턴, 레이블 혹은 분리자 등이 될수 있다. callback() 및 callback_data는 버턴이 눌러졌을때 불려지는 callback 함수 및 파라미터를 나타낸다. enabl flag는 메뉴가 생성될때 버턴의 활성화 여부를 결정하며, this필드에는 자신의 포인터 값을 부여한다. 그림 6.1에서 File 주 메뉴에 대한 부 메뉴 예는 다음과 같다. 메뉴항에서 "..."은 실행시 또 다른 부 메뉴나 혹은 다이알로그 박스가 제공된다.

```
/* Sub-Menu for FileMenu */
static menu_struct FileMenu[] = {
{" Open & Set Neural Network...", &xmPushButtonGadgetClass,
 'O', "Meta<Key>O", "Meta+O", FileOpenCB, (XtPointer)OPEN_NN,
  True, NULL, NULL},
{"separator", &xmSeparatorWidgetClass, NULL, NULL, NULL,
  NULL, NULL, True, NULL, NULL},
{" Open Topology...", &xmPushButtonGadgetClass, 'T', "Meta<Key>O",
 "Meta+T", FileOpenCB, (XtPointer)TOP_FN, True, NULL, NULL},
{"separator", &xmSeparatorWidgetClass, NULL, NULL, NULL,
  NULL, NULL, True, NULL, NULL},
{" Save All...", &xmPushButtonGadgetClass, 'A', "Meta<Key>A",
 "Meta+A", FileOpenCB, (XtPointer)SAVE_ALL, True, NULL, NULL},
{" Save Weight...", &xmPushButtonGadgetClass, 'S', "Meta<Key>S",
 "Meta+S", FileOpenCB, (XtPointer)SAVE_WEI, True, NULL, NULL},
{" Save Result of Test...", &xmPushButtonGadgetClass, 'R',
 "Meta<Key>R", "Meta+R", FileOpenCB, (XtPointer)SAVE_RES,
  True, NULL, NULL},
{"separator", &xmSeparatorWidgetClass, NULL, NULL, NULL,
  NULL, NULL, True, NULL, NULL},
{" Quit ", &xmPushButtonGadgetClass, 'Q', "Meta<Key>Q",
 "Meta+Q", ExitCB, NULL, True, NULL, NULL},
   NULL
};
```

Blind 메뉴

Blind 메뉴는 많이 사용되는 메뉴의 다이알로그를 pop-up 형식으로 필요시 화면에 두고 사용하는 메뉴이다. 이는 다이알로그를 close 하지 않고 설정된 함수를 수행함으로서 가능하다. 예를 들면 현재 학습 상태를 보여주는 학습 파라미터 정보 다이알로그 box가 있다.

6.2 신경망 예측 모델 기술

예측 시뮬레이터를 사용하기 위하여 신경망 모델을 정의한 모델 사본(model script)을 정의해야 한다. 모델 사본 화일은 약 15개의 신경망 topology 구성 및 학습에 필요한 파라미터들로 이루어진다. 문자 # 으로 시작하는 라인은 설명문으로 인식되며, 각 키워드 정의를 설명하는데 도움이 된다. 또한 blank 라인은 무시된다. 다음은 sin 파 function approximation의 경우 모델 사본 화일의 예이다.

```
#
# determine the topology of a neural network
#     Model Script : sin.nn
#
layers= 3
neurons= 1 5(3) 1(4)
patttern_fn= sin.pat
weight_fn= sin.wei
response_fn= sin.res
test_fn= sin.tst

#
# set parameters about learning
#
no_of_patterns= 21
```

```
no_of_forecasts= 5
learning_method= 5
error_function= 3
initial_weight_range= -1.0 1.0
learning_rate= 0.01
momentum_rate= 0.00
target_error= 0.02
total_iterations= 350
#
# determine parameters about graphs
#
graph_scale= 4.5
```

모델 사본 화일의 확장명은 ".nn"이다. 위 에에서 보듯이 모델 사본 화일에는 모델 topology, 학습 파라미터, error 그래프에 관계되는 파라미터들이 정의된다.

6.3 학습데이타 전처리

학습데이타 준비는 주어진 데이타로부터 패턴들간에 내제된 dynamics를 잘 반영하는 학습패턴을 선택하는 것으로부터 시작된다. 학습데이타 전처리는 신경망 예측 모델 설정을 위한 데이타 encoding 과정이다. 준비된 학습 데이타들은 전처리를 통하여 학습에 적절한 패턴들로 변형된다. 신경망 예측 모델 시뮬레이터의 학습에 이용되는 데이타는 $[0, 1]$ 혹은 $[-1, 1]$ 내의 값으로 scaling 하여 사용한다. 학습데이타 처리는 응용시스템에 적합하게 이용되나 대체적으로 continuous feature 와 nominal feature 방법이 이용된다. Nominal feature 처리방법은 각 학습 feature을 0, 1의 이진 값으로 표현하여 학습데이타를 생성시키는데 이용되는 방법으로 기상예측을 자료의 풍속을 4방향으로 나타날때 풍향을 부여하는데 이용될 수 있다. 즉 서쪽에서 바람이 있으면 풍향 벡터 $(0, 1, 0, 0)$으로 나타내어 학

습에 이용한다. 학습데이타 정규화는 scaling 식을 가지고 한다. 정규화 전략은 학습패턴이 $n-$component 벡터로 구성되어 있는 때를 일반적인 경우로 보고 학습패턴 행열의 열에 대한 최대값(max) 과 최소값(min), 평균(mean,μ)를 계산하여 scaling 식을 만든다. 다음은 학습데이타 x_i이 대한 정규화된 데이타 $x_i^{'}$의 scaling 방법들 이다.

정규화 방법 1

$$x_i^{'} = \frac{x_i}{max}$$

정규화 방법 2

$$x_i^{'} = \frac{x_i}{max + \alpha\ max},\ 0 \leq \alpha \leq 1$$

정규화 방법 3

$$x_i^{'} = \frac{x_i - min}{max - min}$$

정규화 방법 4

$$x_i^{'} = \frac{x_i - min}{max}$$

정규화 방법 5

$$x_i^{'} = \frac{x_i}{\sqrt{\sum_j x_j^2}}$$

6.4 전달 함수(Transfer Function)

신경망 모델 뉴런은 전달 함수에 따라 출력값을 filtering한다. 뉴런의 전달 함수는 문제에 따라 다르게 정의된다. 3층을 사용하는 MLP인경우 function approximation을 목적으로하는 경우 중간층 뉴런의 전달함수는 sigmoid 형의 함수를 사용하며. 출력층 뉴런의 경우 선형함수로 filtering한다.

Logistic-sigmoid 함수

$$f(x) = \frac{1}{1 + \exp(-x)}$$

Tangent-sigmoid 함수

$$f(x) = \tanh(x)$$

Gaussian 함수

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

Linear 함수

$$f(x) = x$$

Saturated linear 함수

$$f(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x \geq 1 \end{cases}$$

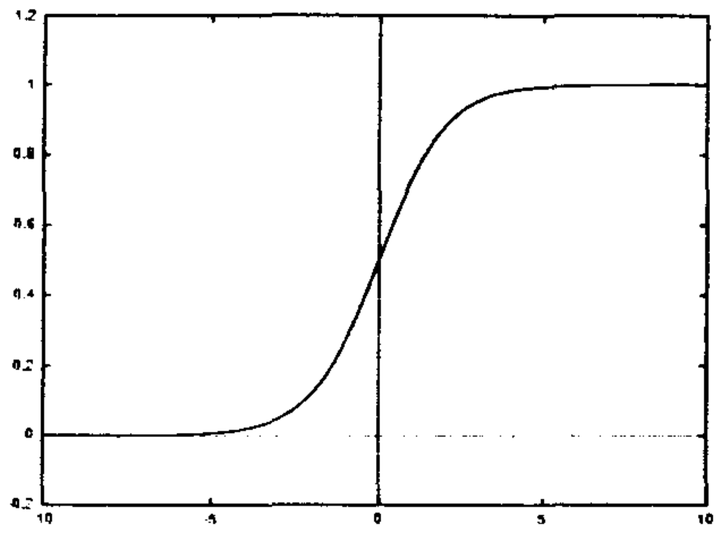
Step 함수

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ \frac{1}{1+\exp(-x)} & \text{otherwise} \end{cases}$$

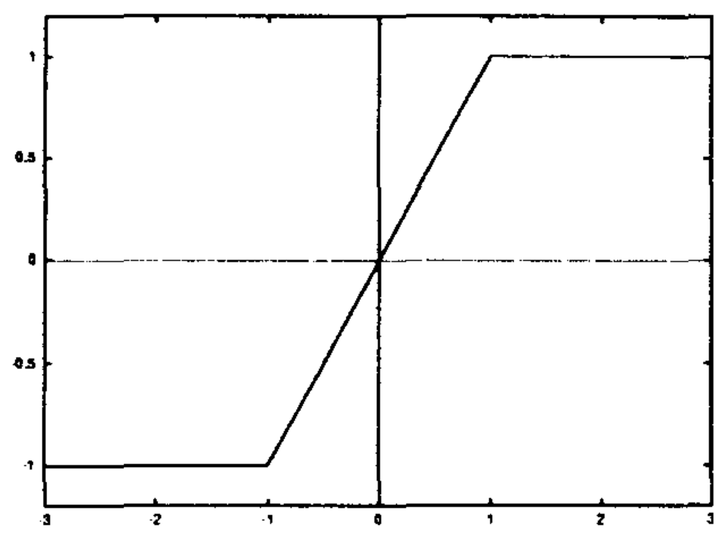
그림 6.2은 기술된 함수 그래프들 이다.

6.5 학습 알고리즘

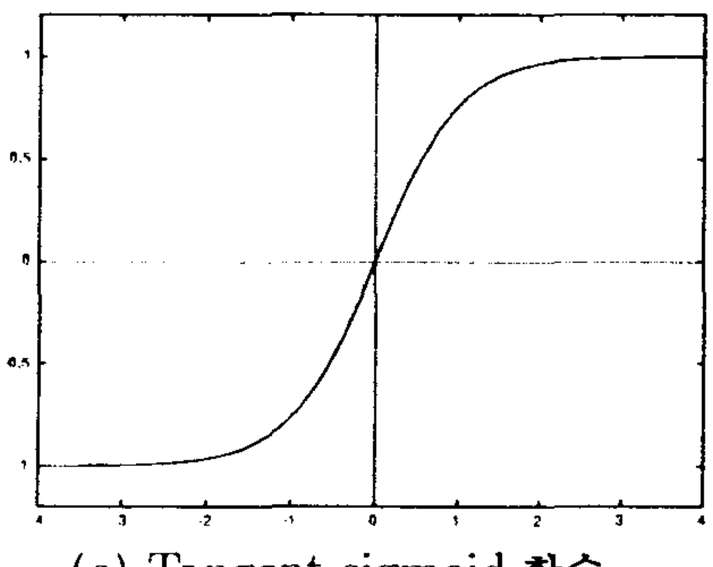
예측 시뮬레이터의 학습 알고리즘은 MLP topology로 적용되는 supervised 학습을 제공한다. 학습 알고리즘으로는 backpropagation 및 temporal backprop-



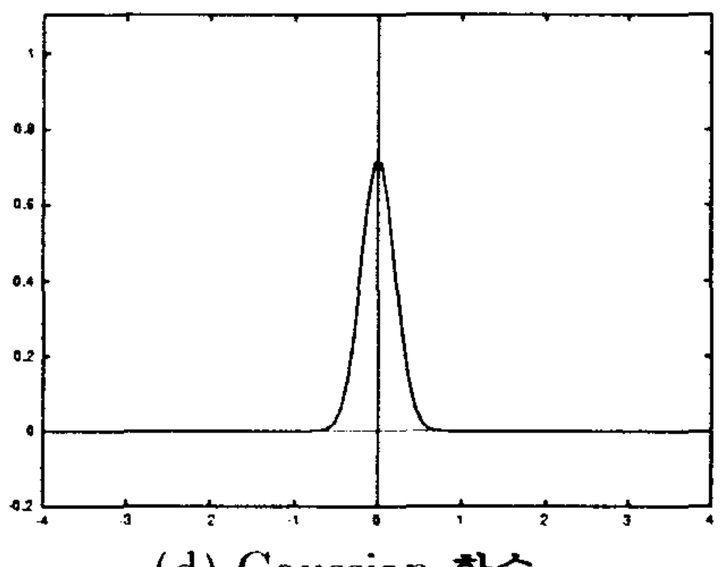
(a) Sine-sigmoid 함수



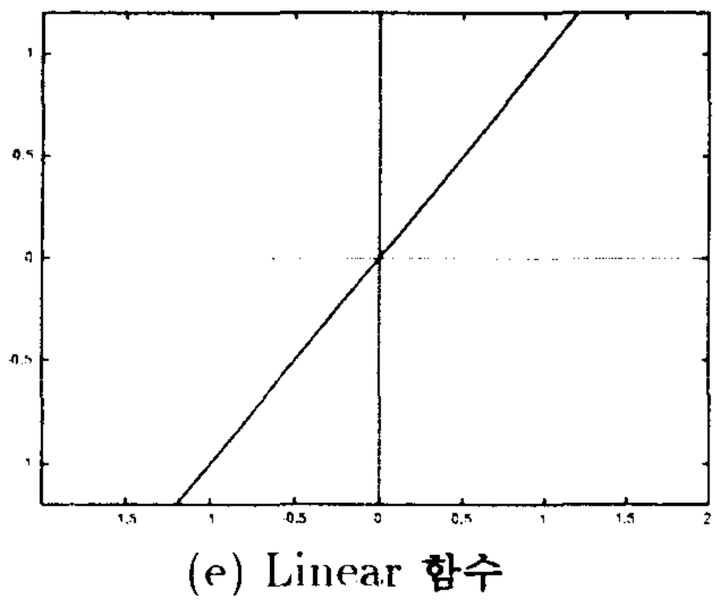
(b) Saturated linear 함수



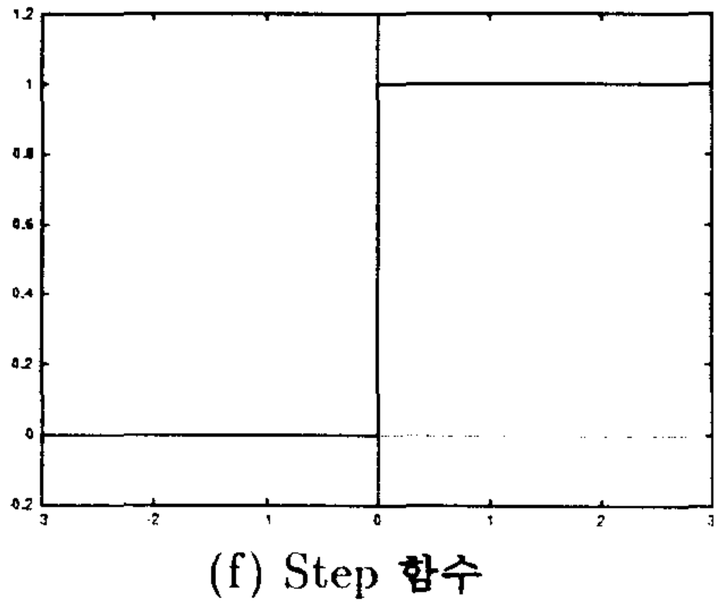
(c) Tangent-sigmoid 함수



(d) Gaussian 함수



(e) Linear 함수



(f) Step 함수

그림 6.2: 전달함수 그래프의 예

agation이 있으며 학습방법에 따라 gradient 및 steepest 학습 방법이 사용가능하다. 학습 알고리즘에는 bias 및 momentum 항의 이용이 가능하며 신경망 모델 script 화일에서 사용 여부를 정의하면 된다.

Backpropagation

출력층 뉴런의 transfer 함수에 따라 학습방법에 차이가 있다. 중간층의 뉴런은 semilinear 함수을 사용하며 입력층의 transfer 함수는 고려하지 않는다. η를 learning rate, α를 momentum rate라 할때 연결강도 W의 변화식은 다음과 같다.

1. 출력층 뉴런이 linear 함수를 사용

   - 중간층과 출력층 사이 연결강도

$$w_{ji}(t+1) = w_{ji}(t) + \eta o_j(t_j^p - o_j) + \alpha \Delta w_{ji}(t)$$

   - 입력층과 중간층 사이 연결강도

$$w_{ji}(t+1) = w_{ji}(t) + \eta o_j(1-o_j) \sum_k \delta_k w_{kj}(t+1) + \alpha \Delta w_{ji}(t)$$

2. 출력층 뉴런이 semilinear 함수를 사용

   - 중간층과 출력층 사이 연결강도

$$w_{ji}(t+1) = w_{ji}(t) + \eta o_j(t_j^p - o_j)o_j(1-o_j) + \alpha \Delta w_{ji}(t)$$

   - 입력층과 중간층 사이 연결강도

$$w_{ji}(t+1) = w_{ji}(t) + \eta o_j(1-o_j) \sum_k \delta_k w_{kj}(t+1) + \alpha \Delta w_{ji}(t)$$

Temporal Backpropagation

층과 층사이의 뉴런들이 시간의 정보를 고려한 연결강도를 갖게 되므로 각 뉴런은 하위 층과 상위 층과 연결된 뉴런들에 대한 인덱스를 갖고 있다. 각 뉴런의 인덱스에 대한 정보는 사용자 메뉴 File의 Open Toploogy...에서 topology 화일을 읽어 설정한다. Topology 화일의 확장명은 "fn.nn"이다. 학습 오차에 대한 adapation은 층사이에연결된 각 뉴런의 인덱스를 가지고 backpropagation 학습을 진행한다.

위 학습을 진행시키는데는 다음과 같은 학습 방법을 제공한다.

Steepest descent 또는 batch learning 전체 학습패턴들에 대한 total average error의 derivative는 connection 에 대한 누적된 error derivative 정보를 갖게 된다. 이 derivative를 이용한 학습은 전체 error를 최소화하는 방향으로 학습이 진행된다고 볼수 있다.

Example-based learning 각 학습에 대한 출력을 구한 후 weight adaptation을 진행 시킨다. 이 방법은 초기에 학습이 빠르게 진행되지만 학습시 다음 사항을 고려해야 한다.

- 가장 최근에 나타난 패턴에 대한 학습이 이전의 패턴에 대한 학습보다 좋음
- 학습패턴이 보여지는 순서에 따라 학습 진행속도가 달라짐

Epoch-based learning 전체 학습패턴이 반복되어 나타나는 일정 횟수에 도달하면 그때까지의 average error의 derivative을 이용하여 학습시키는 방법.

6.6 학습 성능 평가

학습정도를 평가하는 식은 학습이 끝나는 시점을 알려주는 척도가 된다. 학습 종결 조건은 학습패턴 반복수을 이용하지만 조기 학습 종결을 아는데는 도움을 주지 못한다. 문제에 따라 성능 평가식 선택은 적절하게 이루어 져야 한다. 평가식 선택 방법

은 2-3번의 초기학습으로부터 설정되는 것이 일반적이다. 즉 단순분류 문제의 경우 error 값이 어느정도 높아도 패턴을 구별할수 있는 반면, function approximation 문제의 경우는 낮은 error를 얻는데 유용한 학습 평가식을 사용하여야 한다. 선택된 평가식을 가지고 현재 학습상태 error 그래프를 보여준다. p가 학습패턴의 인덱스, k를 벡터 성분에 대한 인덱스라 하자. 다음은 가능한 학습 평가식들을 정의하고 있다.

Least Mean Square

$$E = \sum_p \sum_k (t_k^p - o_k)^2$$

Average Error per Neuron

$$E = \frac{\sum_p \sum_k (t_k^p - o_k)^2}{N}$$

Root Mean Square

$$E = \sqrt{\frac{\sum_p \sum_k (t_k^p - o_k)^2}{N}}$$

Normalized Mean Squared Error

$$E = \frac{1}{\sigma^2 N} \sum_{k=1}^{N} (t_k^p - o_k)^2$$

6.7 실험 결과

기업 도산 예측

1992년부터 1994년까지 3년간 상장된 의류 관련업 중소기업의 도산예측이 실험되었다. 중소기업의 재무, 경영, 재고 등의 년간 평균 수치를 계산하여 해당 기업 관련 정보 데이타가 생성되었다. 이 데이타는 24개의 항으로 구성된다. 예측 실험을 위하여 2개의 도산된 기업과 3개의 도산되지 않은 기업 데이타를 학습후 cross validation에 이용하는 검증데이타로 구분하였다. 학습에 이용된 신경망

topology는 3층의 MLP이며 학습 알고리즘은 backpropagation으로 실험을 하였다. 학습방법은 example-based 학습이며 다음 두가지 신경망 topology에서 학습이 진행되었다. 출력값은 2 진값으로 하여 안정된 기업 경영을 0 으로 나타내고 도산 가능성은 1 로 하였다.

실험 1 준비된 데이타를 전처리 하지 않고 정규화만하여 $24 \times 2 \times 1$인 topology에 학습

실험 2 준비된 데이타성분을 통계적인 방법으로 분석하여 도산 가능성에 유의하지 않은 8개 성분을 제외한 16-component 학습 벡터를 구성시켜 $16 \times 2 \times 1$로 구성된 topology에 학습

예측실험 결과로 **실험 1**에서는 cross validation 집합에 대해 정확한 예측을 수행한 반면, **실험 2**에서는 도산되지 않은 1개의 기업에 대하여 도산으로 판별하였다. 여기서 통계적 분석이 유의성이 없음을 알수 있었으며 raw 데이타을 학습에 이용시 학습성능이 우수함을 알수 있다.

NH3 Laser 데이타 예측

Santa Fe Institute에서 제공한 벤치마크용 데이타 집합 A의 처음 1000개 시계열 데이타를 추출하여 FIR 신경망 모델에서 예측성능을 실험하였다. 사용한 FIR 신경망 모델은 momentum와 bias 항을 가지고 학습된다. 학습방법은 처음 900개의 데이타를 신경망 모델에 학습시키고 나머지 100개의 시계열 데이타에 대하여 예측을 한다. 예측방법은 iterative prediction을 사용하여 t에서 예측치 $\tilde{x}(t)$를 $t+1$에서 예측치 $\tilde{x}(t+1)$를 예측하는데 이용되었다. Topology는 $36 \times 11 \times 6 \times 1$이고 time delay는 25-5-5로 하였다. 그러므로 학습벡터의 time embedding은 36이 된다. 출력뉴런의 transfer 함수는 sigmoid 로 실험되었다. 학습 진행은 error값이 0.04까지 하였다. 그림 6.3는 시간 901에서 1000까지 예측결과를 보여준다.

예측 결과로부터 0에서 50까지(시간 901에서 951)까지는 차이는 있으나 예측 경향이 비슷함을 알수 있다. 그러나 50 번째 이후의 값은 누적된 error 값으로 인

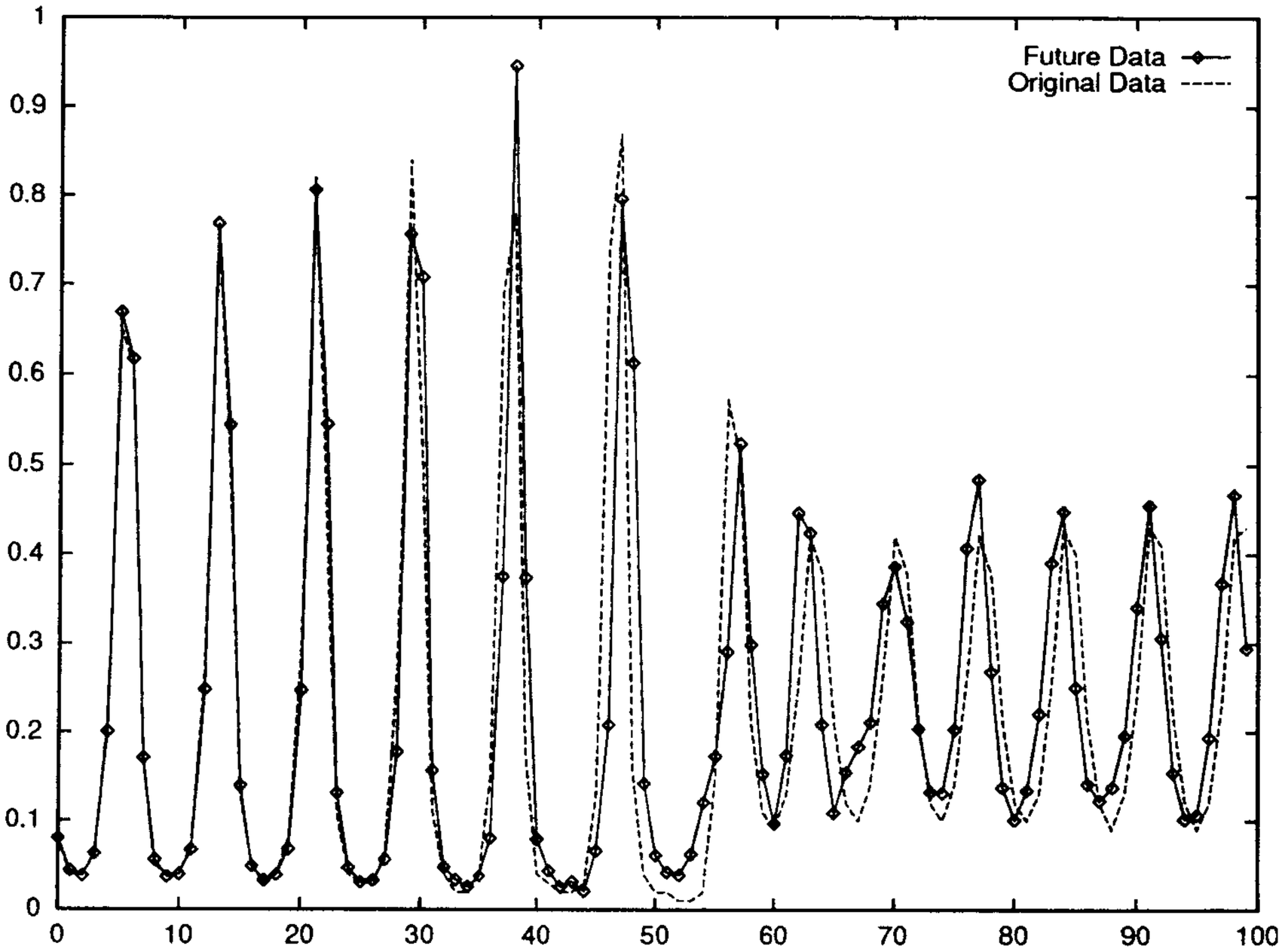


그림 6.3: NH3 laser 데이타 예측 예

하여 예측치의 보정이 요구된다. 그러나 경향은 비슷하게 나타나고 있다.

7 장

결 론

본 연구과제에서는 시계열 예측문제를 해결하기 위한 신경망 예측모델, 퍼지이론 및 통계이론과 연계성에 대한 연구가 진행되었다. 또한 회전기계의 고장진단 및 수명예측 응용시스템에 대한 연구, 기업도산 예측 및 비선형 데이타 예측에 대한 연구가 병행되었다. 신경망. 퍼지 논리, 유전 알고리즘 및 혼돈 이론 각각에 대한 장 단점을 분석하고, 이를 바탕으로 각각의 방법들을 유기적으로 결합하는 새로운 모델 개발 및 기존의 예측 모델들의 방법들을 비교 검토하고, 개발된 새로운 모델들을 이용하여 보다 우수한 성능을 갖는 신경망 예측모델이 개발되었으며, 성능향상을 위한 실험이 진행되고 있다. 또한, 개발된 예측 모델의 성능을 Santafe competition data set를 이용하여 다른 방법들과 성능비교 및 산업체에 응용할 수 있는 기술로 발전시키는 방안에 대하여 연구가 진행되고 있다.

지금까지 연구 및 개발 내용은 다음과 같이 요약된다.

- 예측시스템을 위한 신경망 모델 및 학습 알고리즘에 대한 이론 연구가 수행되었다. 개발된 알고리즘은 backpropagation의 변형으로 각각 기존의 시계열 예측방법보다 실험을 통하여 좋은 예측률을 보여 주었다.
  - 우수한 성능을 갖는 backpropagation 학습 알고리즘
  - 시간의 정보를 포함하는 temporal backpropagation 학습 알고리즘
- 개발된 신경망 모델을 이용하여 기존의 예측 모델들의 성능 비교를 위해 많이 사용되는 Santafe data set A(Chaotic intensity pulsations of a NH3

laser)에 적용하여 예측 성능을 비교한 결과, 개발된 알고리즘을 갖는 신경망 예측 모델이 single-step time series인 경우 좋은 결과를 얻을 수 있었다.

- 기존의 신경망 모델은 비구조적인 정보를 수치적으로 표현할 수 있지만, 실세계에서는 여러가지 구조적인 정보가 존재하여 주어진 문제를 해결하기 위해서는 이러한 정보를 최대한 이용하는 것이 바람직하다. 따라서, 구조적인 정보를 처리할 수 있는 퍼지 논리를 신경망에 융합한 모델 개발이 연구되었으며, 여러가지 비선형 동적 특성을 갖는 시스템을 대상으로 그 성능을 확인 하였다.

- 다층 구조 신경망의 각 층의 뉴런들의 입출력 전달 특성으로 sigmoid 함수와 radial basis function 뿐만 아니라 Garbor 함수를 함께 이용하는 방법을 연구함으로써, 각각의 전달 함수만을 이용하는 경우의 단점들을 극복하며, 또한 장점들을 적절히 융합하는 새로운 신경망 모델을 제시 하였다.

- Sigmoid 함수의 일차미분이 Gaussian 함수와 유사한 형태를 가지고, 삼차미분값에 부호를 바꾼 함수가 Garbor 함수의 형태가 된다. 한편, Gaussian 형태의 함수는 지역 패턴 수록 능력이 뛰어나 신경망 모델에 추가 학습의 가능성을 주며, Garbor함수를 이용함으로써 chaotic 동작을 표현하는 신경망을 구성할 수 있을 것으로 기대된다. 따라서, sigmoid 함수의 일차 미분과 삼차 미분 또는 그 이상의 미분값을 적절히 이용함으로써, local minimum문제와 같은 기존의 신경망의 단점을 극복하고, chaotic 동작을 갖는 예측 문제들을 효과적으로 해결할 가능성을 보였다.

- 전통적인 시계열 예측과 신경망 예측 방법의 여러 응용분야에서 비교 및 성능분석(비선형 데이타, 기업 도산 예측 등), 회전기기에서 주파수 분석을 통한 편심, 미스 얼라인먼트, 볼트풀림, 기어현상, 베어링 이상을 진단하는 신경망 진단시스템 및 수명예측시스템 개발 연구가 병행되었다.

앞으로 iterative prediction에서 예측 데이타의 속성에 따라 여러개의 신경망을 이용하거나 또는 다른 지능형 기술과 접목되는 융합 예측시스템 구현에 대한 연구가 필요하다. 본 과제에서 수행한 연구 결과들이 미래의 정보처리 기술의 핵심 방법이 될 신경망, 퍼지 논리, 유전알고리즘 및 혼돈 이론의 통합형 모델 개발의 토

대를 마련하였으며, 보다 우수한 성능을 갖는 예측모델 개발에 기반이 될수 있는 가능성을 얻었다. 향후, 개발된 기술을 증시 예측, 전력 수요 예측, 기후/기상 예측, 재정 및 경제 분야 예측등의 예측 모델뿐만 아니라, 고성능 패턴 인식, 최적화, 로보트나 전자 기기의 최적 제어등에 폭넓게 적용될 수 있을 것이다.

여 백

참고 문헌

[1] POSCON, 신경회로망을 이용한 회전기계의 이상진단 시스템 개발 , Technical report, 1993.

[2] Krogh A. and Hertz J. A. *Advances in Neural Information Processing Systems 4*, pages 950–957, 1992.

[3] Waibel A., Hanazawa T., Hinton G., Shikano K., and Lang K. *IEEE Trans. Acoustics, Speech, Signal Processing*, pages 328–339, 1989.

[4] Weigend A., Rumelhart D., and Huberman B. *Advances in Neural Information Processing Systems 3*, pages 875–882, 1991.

[5] E.B. Baum and D. Hausser. What size net gives valid generalization ? In *Neural Information Processing Systems*, pages 81–90, 1989.

[6] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control.* Prentice Hall, third edition, 1994.

[7] E. A. Peck Douglas C. Montgomery. *Introduction to Linear Regression Analysis.* John Wiley & Sons, Inc., 1982.

[8] Lynwood A. Johnson Douglas C. Montgomery and John S. Gardiner. *Forecasting and Time Series Analysis.* McGraw-Hill, Inc, 1990.

[9] D.E. Goldberg. *Genetic Algorithm in search, optimization and machine learning.* 1989.

[10] S. Horikawa, T. Furuhashi, and Y. Uchikawa. *IEEE Trans. Neural Networks*, pages 801–806, 1992.

[11] Sietsma J. and Dow R. J. F. *Neural Newtorks*, pages 67–79, 1991.

[12] Fukushima K. *Proc. International Joint Conference on Neural Networks*, pages 2049–2054, 1993.

[13] Rhee M. Kil. Function approximation based on a network with kernel functions of bounds and locality : an approach of non-parametric estimation. Technical report, 161 Ka-Jeong Dong, Yu-Seong, Taejon, Korea, 1993.

[14] Rhee M. Kil and Jin Y. Choi. Time series prediction based on fractal theory and nonelinear estimation using a network with gaussian kernel functions. *Neural. Parallel and Scientific Computaions*, pages 477–500. 1993.

[15] B. Kosko. *Neural networks for signal processing.* 1992.

[16] S. Lee and R.-M. Kil. A gaussian potential function network with hierarchically self-organizing learning. *Neural Networks*, 1991.

[17] S. Y. Lee and M. Lee. *World Congress on Neural Network*, 1995.

[18] S.Y. Lee and D.G. Jeong. In *Proc. Int'l Conf. on Neural Information Processing*, pages 189–194, 1994.

[19] Ishikawa M. *Proc. International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, pages 37–44, 1994.

[20] M.B. Priestley. *Spectral Analysis and Time Series.* Academic Press, 1881.

[21] Reed R. *IEEE Transction on Neural Information Processing*, 4:882–994, 1994.

[22] G. E. Hinton Rumelhart, D. E. and R. J. Williams. Learning internal representations by error propagation. 1986.

[23] LeCun Y.and Denker J. S. and Solla S. A. *Advances in Neural Information Processing Systems 2*, pages 598–605, 1990.

[24] Eric A. Wan. Time series prediction by using a connectionist neural network with internal delay lines. *SFIS Studies in the Sciences of Complexity, Proc. Vol. XV*, pages 195–217, 1993.

[25] Andreas S. Weigend and Neil A. Gershenfeld. *Time Series Prediction: Forecasting The Future and Understanding The Past.* Addision-Wesley Publishing Company, 1994.

[26] Andreas S. Weigend, Bernardo A. Huberman, and David E. Rumelhart. Predicting the future: A connectionist approach. Technical report, Stanford University, Stanford, California 94305-2130, 1990.

[27] LeCun Y., Boser B., Denker J. S., Howard R.E. Henderson D., Hubbard W.. and Jackel L. D. *Neural Computation*, pages 541–551. 1989.

[28] T. Yamakawa. E. Uchino, T. Miki, and H. Kusanagi. In *Proc. of the 2nd Int'l Conf. on Fuzzy Logic and Neural Networks*, pages 477–484, 1992.

[29] L.A. Zadeh. *IEEE Computer Magazine*, pages 83–93, 1988.