APPENDIX

부록

MATLAB과 인공지능의 만남

CONTENTS

LESSON 01

인공지능의 개요

인공지능의 역사

요즘 학문과 산업 분야에서 가장 따끈따끈한 주제 중 하나는 인공지능(AI: Artificial Intelligence)이다. 지능 시스템을 구축하고 지능 행위를 위한 많은 부분에 이용하고 있는 인공지능은 1996년에 대학 교수 그룹이 제안 및 창설하여 매년 열리는 국제 로봇 경쟁 대회인 RoboCup(2019년 대회는 호주 시드니에서 개최)이 그 시작점이다. 이 대회의 주요 목표는 엄청난 도전을 제공함으로써 로봇 기술과 인공지능 연구를 촉진하는 것이다.

1997년 당대 체스의 일인자 가리 카스파로프(Garry Kasparov)와 대전한 딥블루(Deep Blue)는 인공지능을 이용한 최초의 체스 게임 컴퓨터이다. 이후 2016년 구글 딥마인드(DeepMind)에서 개발한 바둑 컴퓨터 알파고(AlphaGo)와 바둑 기사 이세돌의 바둑 시합을 통해 우리나라에서도 인공지능이 큰 주목을 끌게 되었다.

인공지능의 역할

복잡한 세상을 슬기롭게 살아가려면 하루하루 엄청난 지식이 필요하다. 그런데 이러한 지식은 대부분 주관적이고 직관적이어서 다른 사람도 쉽게 이해할 수 있는 객관화된 일반적 지식으로 표현하기가 어렵다.

인간의 경험을 통해 쌓은 지식 중에 복잡한 것들을 공식화·규정화할 필요가 있을 때 컴퓨터를 이용하면 복잡한 개념의 계층 구조를 학습시켜서 간단한 개념을 얻을 수 있을 뿐만 아니라 인간의 수고를 더는 일석이조의 효과를 볼 수 있다. 결국 인공지능의 주요 도전 과제는 비정형적인 인간의 지식을 컴퓨터에 접목하여 정형화하는 데 있다.

사례 A-1

5G를 위한 인공지능

5G는 5세대 이동통신망 기술로 현재 사용하고 있는 이동통신의 혁신 분야이다. 기존의 이동통신망 기술인 LTE와 4G의 불편한 문제를 해결할 수 있는 고급 기능을 갖춘 5G 기술은 휴대전화를 4G보다 초고속으로 사용할 수 있게 해준다. 5G의 등장으로 이동통신망 기술이 진일보한 반면 더욱 복잡한 기술적 문제가 발생할 수 있으므로 이동통신 서비스 제공 회사는 이를 해결하기 위한 하나의 방법으로 인공지능 기술을 통신망에 통합할 필요성을 절감하고 있다. 따라서 전 세계 이동통신 서비스 제공 회사는 현재 및 향후에 인공지능을 채택하기 위한 계획을 수립한 상태이며, 일부 회사는 이미 인공지능을 구현했다. 인공지능은 자본 지출 감소, 통신망 성능의 최적화, 새로운 수익원 창출에 중점을 두어 사용되고 있으며, 통신망의 품질과 고객 개개인의 서비스를 향상하는 동시에 다양한 방법으로 고객 경험을 더욱 향상할 수 있는 터전을 마련할 것으로 기대된다.

사례 A-2

의료를 위한 인공지능

인공지능을 이용하면 원본 의료 데이터에서 의미 있는 데이터 관계를 판별할 수 있으므로 다양한 의료 상황에서 진단, 치료, 결과 예측이 가능해진다. 따라서 인공지능은 신약 개발, 환자 관찰, 맞춤형 환자 치료 계획 등을 비롯해 거의 모든 의료 분야에 적용 가능한 잠재력을 가지고 있다. 인공지능은 전자 건강 기록 데이터와 비구조화된 데이터를 종합하여 환자의 건강에 대한 예후를 가능하게 한다. 인공지능을 통해 의사는 환자의 치료를 위해 더 좋은 결정을 내릴 수 있는 간결한 데이터를 얻게 된다.

의료 분야에서 인공지능이 어떻게 이용될 수 있는지 두 가지 예를 통해 살펴보자. 첫째로, 인공지능은 구조화된 문서와 비구조화된 문서를 통해 환자의 과거 의료 기록에 명시된 문제를 식별할 수 있다. 이를 바탕으로 환자의 병력에 대한 치료 경력을 요약하고 환자의 기록에 대한 인지적 요약도 제공할 수 있다. 둘째로, 인공지능은 환자 사이의 미묘한 임상적 유사성 척도를 식별할 수 있다. 이를 통해 연구자는 정적 환자 집단보다 동적 환자 집단을 생성함으로써 환자 집단에서 어떤 진료 방법이 더 좋은지 판단하게 된다.

LESSON 02

머신러닝과 딥러닝의 기본 개념

머신러닝의 개념

머신러닝(ML: Machine Learning)은 데이터에서 형식을 추출하는 알고리즘이나 방법을 사용하는 인공지능의 한 유형이다. 머신러닝은 기계가 벡터 또는 행렬과 같은 수치 형태로 나타내는 데이터로부터 학습할 수 있게 해주는 알고리즘과 기술의 집합이다. 머신러닝을 이용하면 결과 예측이나 결정이 가능하고 경험에 의해 축적된 데이터를 이용하여 더 좋은 결과를 얻을 수 있다.

머신러닝 관련 기초 용어

지도학습

지도학습(supervised learning) 또는 함수 근사는 단순히 데이터를 다양한 함수에 적합시키고 훈련하여 데이터를 분류한다. 지도 학습 방법은 지식 발견에 접근하는 베이지안을 채택하고 이전에 관찰된 사건의 확률을 사용하여 새로운 사건의 확률을 추론한다. 그러므로 훈련 및 분류된 모든 데이터는 올바르게 추측되어 최상의 원하는 목푯값에 근접하게 된다.

- **분류(classification)** 예측하려는 목표 변수가 이산값(discrete value)인 것을 각각 예측하고 관측 결과를 범주로 지정한다.
- **회귀(regression)** 예측하려는 목표 변수가 연속값(continuous value)인 것을 각각 예측한다.

자율학습

지도학습과 반대로 자율학습(unsupervised learning)은 목푯값을 알지 못하는 상태에서 관찰을 한다. 목표가 없다는 것은 아무것도 예측할 수 없음을 의미한다. 자율학습 방법은 레이블이 없는 자료 집합으로부터 추상적 개념을 이끌어내고 새로운 자료에 적용하므로 레이블이 없는 데이터를 다루기 위해 제안된다. 웹에서 보여주는 다양한 뉴스 기사 중에서 특정 기사를 모아 그룹화하는 것은 자율학습을 이용하는 대표적인 예이다. 대규모 자료를 토대로 새로운 정보를 찾아내는 자율학습을 데이터 마이닝(data mining)이라고 부른다.

- **레이블** 일련화된 희소(sparse) 벡터에서 나온 출력을 각 행에 대한 레이블로 나타낸다.
- **특징** 머신러닝에서 하나의 개체 또는 행을 표본 또는 자료점이라 하고, 표본의 속성을 나타내는 열을 특징(feature)이라 한다. 특징은 원본 데이터로부터 직접 가져올 수 있지만, 대부분의 경우 원본

입력 데이터를 좀 더 모델링에 적합한 형식으로 가져오기 위해 숫자로 변환하여 사용한다.

- **클러스터링(clustering)** 유사성 등의 개념에 기초하여 데이터를 몇몇 그룹으로 분류하는 방법이다. 즉 클러스터링은 거리 측정을 사용하고 반복적으로 유사한 항목을 더 가깝게 만드는 자율학습 기술이다. 클러스터링 알고리즘은 자료가 공간에 배치되는 방법에 따라 집단화하며, 자료점 사이의 근접성 측정을 나타내는 메트릭(metric)에 의존한다.

준지도학습

준지도학습(semi-supervised learning)은 지도학습과 매우 흡사하지만 일부 데이터는 무지도학습처럼 목푯값을 알지 못한다. 예를 들면 기록된 의료 자료로부터 수많은 환자의 데이터를 쉽게 얻을 수 있지만, 환자의 증상에 대한 결과까지 얻는 경우는 극히 드물다.

강화학습

강화학습(RL: reinforcement learning)은 소프트웨어 에이전트가 누적 보상의 개념을 최대화하기 위해 환경에 적절한 조치를 취하는 방법과 관련된 것을 학습하는 머신러닝의 한 영역이다. 이는 단지 좋고 나쁘다는 식으로 간접적으로 결과를 피드백할 뿐 결과에 대한 정확한 답을 제시하지는 않으며, 이때 피드백이 지연될 수 있다.

딥러닝의 개념

복잡한 계층 구조를 그래프로 시각화하면 많은 층을 깊이 있게 보여줄 수 있는데 이러한 접근을 인공지능 딥러닝(deep learning)이라고 부른다. 계층적 학습(hierarchical learning) 및 심화 구조 학습이라고도 일컫는 딥러닝은 2006년 이후 머신러닝에서 새롭게 떠오르는 연구 분야로 자리 잡고 있다. 이후 딥러닝 연구는 머신러닝과 인공지능의 핵심으로서 광범위한 분야로 확대되어 영향을 미치고 있다. 이처럼 딥러닝의 인기가 확대된 것은 급속히 향상된 반도체 칩 처리 능력, 훈련에 사용되는 빅데이터 크기의 증가, 머신러닝, 신호 및 정보 처리의 급격한 발전 덕분이다.

딥러닝은 머신러닝 연구의 새로운 영역으로, 머신러닝을 목표에 더 가깝게 접근시키려는 목적으로 개발되었다. 따라서 딥러닝은 머신러닝의 알고리즘과 유사한 기능을 가진 머신러닝의 하위 집합으로 간주된다. 그러나 이러한 알고리즘에서 여러 계층의 데이터에 대한 해석이 필요한 경우에는 인공 신경망(ANNs: Artificial Neural Networks) 알고리즘을 사용하는데, 딥러닝은 인공 신경망의 알고리즘을 기반으로 하고 있다. 딥러닝은 신경망, 인공지능, 그래픽 모델, 최적화, 패턴 인식, 신호 처리와 같은 연구 분야의 교차점이다.

LESSON 03

MATLAB 예제를 통해 신경망 이해하기

신경망의 개념

초기의 인공지능에서 다루려던 학습 알고리즘은 어떻게 학습이 뇌의 신경망(neural network)을 따라 발생할 수 있는가를 유형화하는 생물학적 학습의 계산 모델이었다. 그 결과 딥러닝을 인공 신경망이라고도 일컫는다. 인공 신경망은 복수 계층에서 이전의 데이터를 이용하여 학습한 후 어떻게 분류하고 예측할지를 가르치는 기능을 가지고 있다.

신경세포(neuron)라고 부르는 충분한 원소를 내포하는 신경망은 임의의 정확도로 모든 데이터를 적합시킬 수 있다. 따라서 신경망은 특히 비선형 문제를 해결하는 데 매우 적합하다.

신경망 관련 기초 용어

신경세포

신경세포는 신경망의 계산적 단위이다. 뇌에 있는 신경세포는 입력과 자극을 처리하고 이해하기 위한 네트워크에서 함께 작용한다. 신경세포는 활성화 함수에 둘러싸인 가중 선형 조합이다. 인공 신경세포는 퍼셉트론 전구체(precursor)와 비슷하지만 신경망에서 계층의 특정한 목적에 의존하는 서로 다른 활성화 함수를 가지고 있다. 신경세포는 활성화 함수를 이용하여 숫자 입력을 다음 신경망의 신경세포 계층으로 전달하는 매우 간단한 원소이다. 신경망에서 신경세포는 노드(node)와 동일한 개념으로 사용된다.

신경세포 가중치

신경세포 가중치(weight)는 신경세포 사이의 연결 세기를 나타낸다. 가중치는 입력이 출력에 미치는 영향을 결정한다. 가중치가 0에 가까운 값이면 입력 데이터를 바꾸어도 출력이 바뀌지 않는다. 신경망에서 가중치는 매개변수와 동일한 개념으로 사용된다.

신경세포 편차

신경세포 편차는 원본 데이터를 훈련하기 이전에 신경망 모델을 평가하기 위한 통계적 개념이다. 신경망에서 편차는 상쇄(offset)와 동일한 개념으로 사용된다.

활성화 함수

신경세포의 행동을 통제하고 각각의 신경세포를 간단한 수학적 계단 함수로 모델화하는데 이것을 활성화 함수(activation function)라고 한다. 활성화는 이전의 각 계층으로부터 다음 계층으로 전달되는 값이며, 이 값은 각 신경세포의 활성화 함수의 출력이다. 신경망에서 입력의 전송은 선행 전파라고 하므로 활성화 함수를 전달 함수라고 부르기도 한다.

활성화 함수에서 입력 및 가중치와 추정치 사이의 차이를 나타내는 편차의 조합을 변환한 결과는 다음 노드 계층의 입력이 된다. 모든 경우는 아니지만 신경망에 사용되는 많은 비선형 변환은 데이터를 0에서 1 또는 −1에서 1과 같은 편리한 범위로 변환한다. 신경세포가 0이 아닌 값을 다른 신경세포에 전달할 때 이를 활성화되었다고 일컫는다. 활성화 함수의 큰 장점은 각 계층에 입력되는 값을 완충하는(buffering) 방법으로 사용하는 데 있다.

퍼셉트론

신경망에서 다루는 퍼셉트론(perceptron)은 활성화 함수에 대한 헤비사이드 계단 함수(Heaviside step function)를 사용하는 신경세포로 간주된다. 퍼셉트론은 2진 분류(binary classification)를 사용하여 간단하게 입력과 출력 관계를 보여주는 모델이다. 그러므로 기본 선형 분류기로서 단일 계층 퍼셉트론은 피드포워드 신경망 계열의 가장 간단한 형태로 간주할 수 있다.

피드포워드 망

피드포워드 망은 연속적으로 계층(layer)이 연결된 구조이다. 첫 번째 계층은 망 입력에서 연결된다. 각각의 후속 계층은 바로 앞에 있는 계층과 연결되어 있다. 망의 출력은 마지막 계층과 연결되어 있다.

신경망의 계층 요약

입력 계층

설계된 신경망에서 입력되는 벡터 데이터는 입력 계층(input layer)에서 처음으로 사용된다. 전형적으로 입력 계층의 신경세포 수는 신경망의 입력 특성과 동일하다. 고전적인 피드포워드 신경망의 입력 계층은 다음에 이어진 은닉 계층(hidden layer)과 연결된다. 입력에는 표준형(standard)과 대칭형(symmetric)이 있다.

은닉 계층

은닉 계층은 입력 계층과 출력 계층(output layer) 사이에 위치한다. 순방향으로 전달되는 피드포워드 신경망에서는 은닉 계층이 다중 계층이다. 은닉 계층은 신경망이 비선형 데이터를 모델화하는 것을 허용한다.

출력 계층

신경망 모델의 출력 계층은 입력 계층의 입력을 기반으로 나오는 출력 결과를 얻는다. 최종 출력 결과는 실숫값인 회귀 혹은 확률의 집합인 분류로 나타낸다. 이는 출력 계층의 신경세포에서 사용되는 활성화 함수의 형태에 의해 제어된다.

신경망 분류

컨벌루션 신경망

컨벌루션 신경망(CNN: Convolutional Neural Network)은 하나 이상 컨벌루션 계층으로 구성되어 있어 특히 영상(image)을 분류할 때 사용하기 좋은 구조이다. 완전하게 연결된 신경망의 경우 각 구성 단위(unit)는 이전 계층의 모든 구성 단위에 연결되어 있다. 그러므로 완전하게 연결된 신경망과 컨벌루션 신경망 사이의 근본적인 차이는 연속적인 계층 사이의 연결 형태에 있다.

재귀 신경망

재귀 신경망(RNN: Recuurent Neural Network)은 연속 데이터를 모델링하기 위해 강력하고 넓게 사용하는 신경망 구조의 한 부류이다.

논리적 AND

[그림 A-1]은 입력 x_1과 x_2에 가중치 w_1과 w_2를 곱한 함수는 출력 y가 되는 것을 보여준다.

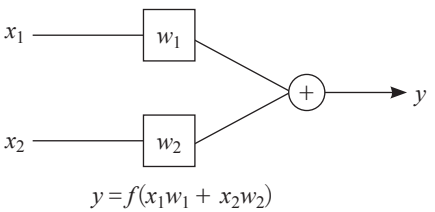

그림 A-1 입력에 가중치를 곱하여 얻는 출력 함수

가중치를 $w_1 = \frac{3}{2}$과 $w_2 = 1$로 지정하고 함수를 $\frac{3}{2}x_1 + x_2 = 1.8$로 정의하자. $\frac{3}{2}x_1 + x_2 < 1.8$일 때의 출력은 $y = 0$, $\frac{3}{2}x_1 + x_2 \geq 1.8$일 때의 출력은 $y = 1$이라고 가정하면 0과 1을 조합한 입력에 따라 다음과 같은 출력을 얻게 된다.

1. $x_1 = 0$, $x_2 = 0$: $0 < 1.8$, 따라서 $y = 0$
2. $x_1 = 0$, $x_2 = 1$: $1 < 1.8$, 따라서 $y = 0$
3. $x_1 = 1$, $x_2 = 0$: $\frac{3}{2} < 1.8$, 따라서 $y = 0$
4. $x_1 = 1$, $x_2 = 1$: $\frac{5}{2} > 1.8$, 따라서 $y = 1$

이를 [표 A-1]에 정리했는데, 이 표는 두 입력의 같은 위치에 놓인 원소가 모두 '0'(또는 거짓값)이 아닌 경우에만 '1'(또는 참값)의 값을 지정하는 논리적 AND와 동일하다.

표 A-1 논리적 AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

[그림 A-2]에서 보듯이 처음 가정한 함수 $\frac{3}{2}x_1 + x_2 = 1.8$에 의해 생성된 결정 경계(decision boundary)를 중심으로 논리적 AND의 참값 집단과 거짓값 집단으로 분류된다.

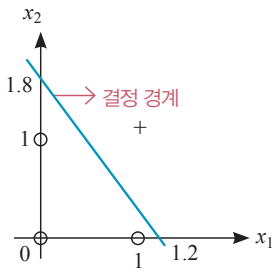

그림 A-2 결정 경계에 의해 분류되는 참값 집단과 거짓값 집단

예제 A-1

M-File | nn_AND.m

MATLAB에서 제공하는 신경망을 이용하여 논리적 AND의 출력 결과를 두 집단으로 분류하는 결정 경계를 나타내라.

배경 지식

■ configure

입력 데이터와 목표 데이터가 가장 잘 일치하도록 네트워크 입력 및 출력을 구성한다. 기본 형태는 다음과 같다.

net = configure(net, p, t)

- **p:** 입력 데이터
- **t:** 목표 데이터

■ perceptron

입력 공간을 선형 결정 경계로 나누는 간단한 단일 계층 2진 분류기이다. 주어진 문제의 계층을 신뢰성 있게 풀 수 있는 최초의 신경망이므로 간단한 학습 규칙을 갖는다는 것이 장점이다. 기본 형태는 다음과 같다.

net = perceptron

■ rands

무작위로 부호가 있는 초기 가중치를 계산하는 함수이며 기본 형태는 다음과 같다.

rands(n, m)

- **n:** 신경세포 계층의 수
- **m:** 입력 원소의 수

■ plotpv

퍼셉트론의 입력 벡터와 목표 벡터를 그리는 명령어이다. 입력 벡터에 따라 나온 결과인 목표 벡터가 0인 경우는 원, 1인 경우는 +로 나타낸다. 기본 형태는 다음과 같다.

plotpv(p, t)

- **p:** 입력 벡터
- **t:** 목표 벡터

■ plotpc

명령어 plotpv를 사용하여 그린 퍼셉트론의 목표 벡터상에 분류 직선을 덧붙여서 그리는 명령어이며 기본 형태는 다음과 같다.

plotpc(w, b)

- **w:** 가중치 벡터
- **b:** 바이어스 벡터

■ train

신경망을 훈련하는 명령어이며 기본 형태는 다음과 같다.

net = train(net, p, t)

- **우측 net:** 새롭게 훈련된 네트워크
- **좌측 net:** 신경망에 사용하는 네트워크
- **p:** 입력 벡터
- **t:** 목표 벡터

■ net.iw

신경망 계층 입력의 가중치이며 기본 형태는 다음과 같다.

net.iw{i, j}

입력 j에서 계층 i에 지정되는 가중치를 나타낸다. 따라서 일반적인 두 계층으로 이루어진 망에서는 입력 1에서 계층 1로 지정되는 가중치를 표시하는 net.iw{1,1}이 존재한다. 반면에 입력 1에서 계층 2로 지정되는 가중치를 표시하는 net.iw{2,1}은 입력이 계층 1로만 전달되기 때문에 비어 있다.

■ net.b

신경망 각 계층의 편차이며 기본 형태는 다음과 같다.

net.b{i}

net.b{1}은 첫 번째 계층의 편차이다. 참고로 두 계층으로 이루어진 망의 경우 편차가 net.b{1}, net.b{2}이다.

풀이

nn_AND.m

```
1 -  p=[0 0 1 1;0 1 0 1]; ❶
2 -  t=[0 0 0 1]; ❷
3    net = perceptron;
4 -  net = configure(net,p,t);
5 -  net.iw{1,1} = rands(1,2); ❸
6 -  net.b{1} = 1; ❹
7 -  net = train(net,p,t);
8 -  w = net.iw{1,1};
9 -  b = net.b{1};
10 - plotpv(p,t);
11 - plotpc(w,b);
12 -
```

그림 A-3 논리적 AND 코딩 결과(nn_AND.m)

❶ 입력 x_1과 x_2를 입력 벡터 p로 지정한다.

❷ 입력 벡터에 논리적 AND를 적용한 후 나오는 출력 y를 목표 벡터 t로 지정한다.

❸ 초기 두 가중치를 무작위로 지정한다.

❹ 초기 편차는 1로 지정한다.

명령 창에 >>nn_AND를 입력하면 [그림 A-4]의 실행 결과를 얻는데 정확하게 두 집단으로 분류된다. 여섯 번 반복 훈련하여 생성된 두 가중치는 2.5570, 1.0938이고 편차는 −3이다. 명령 창에 >>nn_AND를 여러 번 입력하면 매번 반복 훈련 횟수가 바뀌므로 실행 결과가 다르게 나온다.

[그림 A-4]에서 두 집단을 분류하는 결정 경계의 직선식을 구하기 위해 [그림 A-4]의 그림 창에서 [Tools]-[Basic Fitting]을 선택하면 [그림 A-5]와 같은 새로운 창이 열리고, linear와 Show equations에 체크하면 [그림 A-4]에 이미 그려진 직선에 대한 선형방정식 $y = -2.3x + 2.7$을 새롭게 보여준다.

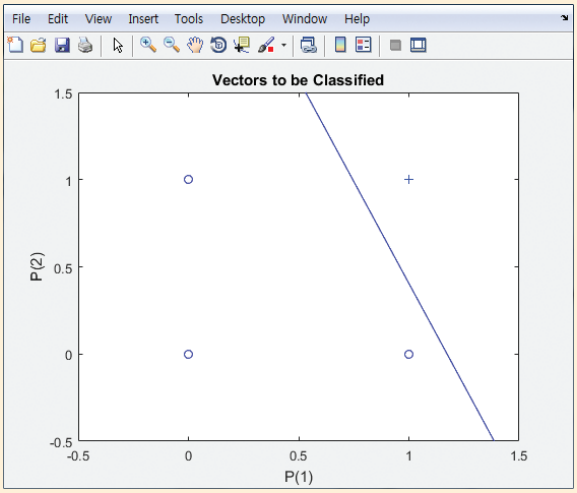

그림 A-4 반복 훈련한 논리적 AND의 실행 결과

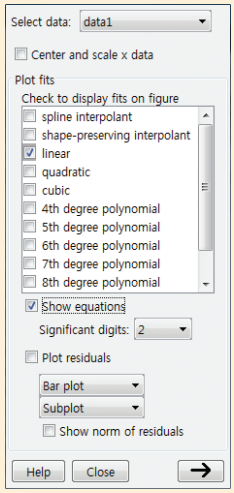

그림 A-5 결정 경계의 선형방정식 구하기

예제 A-2

M-File | nn_OR.m

MATLAB에서 제공하는 신경망을 이용하여 논리적 OR의 출력 결과를 두 집단으로 분류하는 결정 경계를 나타내라.

배경 지식

[표 A-2]에서 보듯이 논리적 OR는 두 입력 벡터의 원소가 모두 '0'인 경우를 제외하고 그 결과를 '1'로 지정한다.

표 A-2 논리적 OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

풀이

[예제 A-1]의 nn_AND.m 스크립트 파일에서 목표 벡터의 값만 t = [0 1 1 1]로 바꾸면 다음과 같은 논리적 OR에 대한 스크립트 파일 nn_OR.m을 쓸 수 있다.

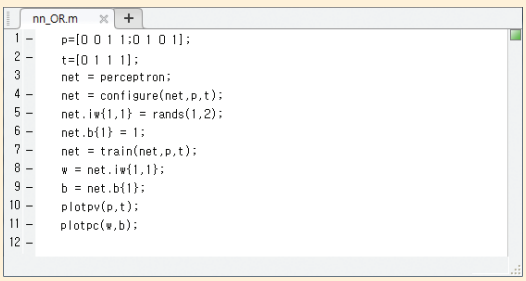

그림 A-6 논리적 OR 코딩 결과(nn_OR.m)

명령 창에 >>nn_OR를 입력하면 [그림 A-7]에서 보듯이 두 집단으로 분류된다. 다섯 번 반복 훈련하여 생성된 두 가중치는 1.3152, 1.9412이고 편차는 −1이다. 그리고 두 집단을 분류하는 결정 경계의 식은 $y = -0.68x + 0.52$이다. [예제 A-1]과 마찬가지로 명령 창에 >>nn_OR를 여러 번 입력하면 반복 훈련 횟수가 바뀌므로 실행 결과 그래프도 매번 바뀐다.

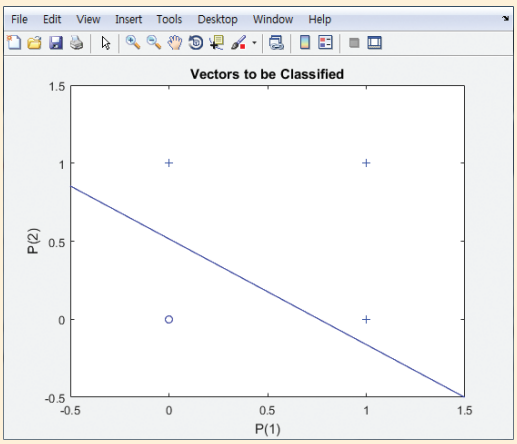

그림 A-7 반복 훈련한 논리적 OR의 실행 결과

논리적 XOR

MATLAB을 사용하여 [그림 A-1]을 설계하면 [그림 A-8]과 같이 논리적 AND와 OR를 구성하는 신경망의 단층 구조를 보여준다. 그러므로 하나의 노드로 구성된 단층 퍼셉트론(SLP: Single Layer Perceptron)만을 사용해도 반복 훈련 후에 결정 경계를 나타내는 한 직선을 중심으로 두 집단을 쉽게 분류할 수 있다.

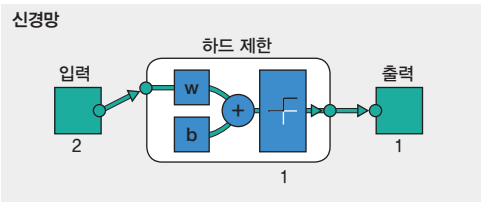

그림 A-8 논리적 AND와 OR를 위한 단층 신경망 구조

hardlim

하드 제한(hard limit) 함수는 단위 계단 함수와 동일한 특성을 지닌 함수 형태로, 총입력이 임곗값(threshold)에 도달하면 신경세포는 1을 출력하고 그 외의 경우는 0을 출력한다. 이 함수는 도함수의 임곗값이 존재하지 않는다. 신경망의 i번째 계층에 logsig 함수를 지정하는 기본 형태는 다음과 같다.

```
net.layers{i}.transferFcn = 'hardlim'
```

[표 A-3]은 논리적 XOR를 보여준다. 이를 이용하여 그린 [그림 A-9]에서 두 집단으로 분류하는 결정 경계는 2개의 직선이 필요하다.

표 A-3 논리적 XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

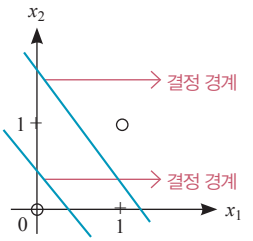

그림 A-9 논리적 XOR의 결정 경계

단층 퍼셉트론만을 이용한 논리적 AND의 코딩에서 목표 벡터의 값만 t = [0 1 1 0]으로 대체한 후 논리적 XOR를 실행하면 [그림 A-10]과 같은 그래프를 얻는다. [그림 A-10]은 1000번 반복 훈련을 실행했지만 두 집단으로 분류되지 않았다. 단층 퍼셉트론만을 사용한 경우에는 반복 훈련의 횟수가 증가해도 논리적 XOR의 결정 경계를 찾을 수 없다.

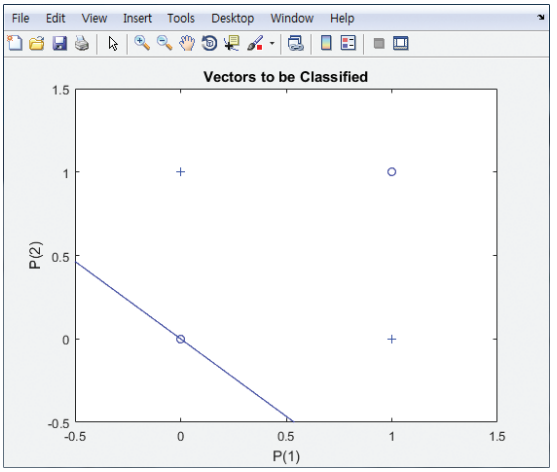

그림 A-10 반복 훈련한 논리적 XOR의 실행 결과

결국 이 문제를 해결하기 위해 단층 퍼셉트론 대신에 더 많은 노드를 사용하는 다층 퍼셉트론(MLP: Multi Layer Perceptron)이 필요하다. 다층 퍼셉트론 신경망은 입력된 데이터가 은닉 계층을 통해 입력 계층에서 출력 계층의 순방향으로 전달되는 피드포워드 망을 처리한다. 각 계층의 각 신경세포는 다음 계층의 다른 신경세포에 연결되어 있지만 동일 계층 내의 신경세포는 서로 연결되어 있지 않은 구조이다.

역전파 학습 규칙

역전파(BP: Back Propagation)는 신경망의 출력에서 발생하는 오류를 최소화하기 위해 신경망에서 연결의 가중치에 기울기 하강(gradient descent) 방법을 사용한다. 역전파 학습은 퍼셉트론 학습의 알고리즘과 유사하다. 신경망을 통해 입력 데이터에 대한 출력을 순방향 전달로 계산한다. 출력이 레이블과 일치하면 다른 작업을 하지 않고, 일치하지 않으면 신경망과 연결된 가중치를 조정한다. 역전파를 이용하면 훈련 입력과 연관된 레이블 출력과 신경망 출력에서 생성된 값 사이의 오류를 최소화할 수 있다.

기울기 하강은 학습률(learning rate)의 선택과 많은 반복이 필요하다. 특성의 수 n이 매우 큰 경우에도 잘 작동한다. 반면에 일반 방정식은 학습률과 반복이 필요하지 않다.

예제 A-3

M-File | nn_XOR1.m

다음 훈련을 위한 초기 조건을 사용하여 논리적 XOR 문제를 학습하기 위한 최소 수의 노드로 이루어진 신경망을 설계하라. MATLAB의 역전파를 이용하여 논리적 XOR의 출력 결과를 두 집단으로 분류하는 결정 경계를 찾아라. 이때 전달함수는 MATLAB의 선형함수 purelin을 사용하라.

$$I_1 = [w_{11}, w_{12}, w_{13}] = [1, 1, 1]$$
$$I_2 = [w_{21}, w_{22}, w_{23}] = [1, -1, 1]$$
$$I_3 = [w_{31}, w_{32}, w_{33}] = [1, 1, -1]$$

여기서 노드 1과 노드 2는 은닉 노드이고 노드 3은 출력 노드이며, w_{ji}는 송신 노드 i와 수신 노드 j 사이의 가중치를 나타낸다.

배경 지식

- net.lw

계층 q로부터 계층 p에 지정되는 가중치이며 기본 형태는 다음과 같다.

net.lw{p, q}

일반적으로 두 계층망 net.lw{2,1}은 계층 1에서 계층 2로 전달되는 가중치를 포함하고 다른 계층의 가중치는 비어 있다.

- feedforwardnet

피드포워드 신경망을 설정하는 명령어이다. 피드포워드 망은 일련의 계층으로 이루어져 있다. 망의 입력값에서 시작하는 첫 번째 계층은 망의 입력값과 연결되고, 뒤에 오는 각 계층은 바로 직전 계층과 연결된다. 마지막 계층은 네트워크의 출력값을 생성한다. feedforwardnet 명령어의 기본 형태는 다음과 같다.

feedforwardnet(hiddenSizes, trainFcn)

- **hiddenSizes:** 하나 이상의 은닉 계층 크기로 구성된 행 벡터(기본값은 10)
- **trainFcn:** 훈련 함수(기본값은 'trainlm')

■ trainlm

레벤버그-마쿼트(Levenberg-Marquardt) 최적화 방법에 따라 가중치와 편차의 데이터를 변경하는 신경망 훈련 함수이다. 신경망 도구상자에서 가장 빠른 역전파 알고리즘으로 다른 알고리즘보다 더 많은 메모리가 필요한 것이 단점이지만 지도학습의 알고리즘에 자주 사용한다. 기본 형태는 다음과 같다.

```
net = trainlm(net, p, t)
```

train과 함께 사용하는 경우의 기본 형태는 다음과 같다.

```
net.trainFcn = 'trainlm'
net = train(net, p, t)
```

풀이

[그림 A-11]은 훈련을 위한 초기 조건을 사용하여 논리적 XOR 문제를 학습하기 위한 최소 수의 노드로 이루어진 신경망의 설계를 보여준다.

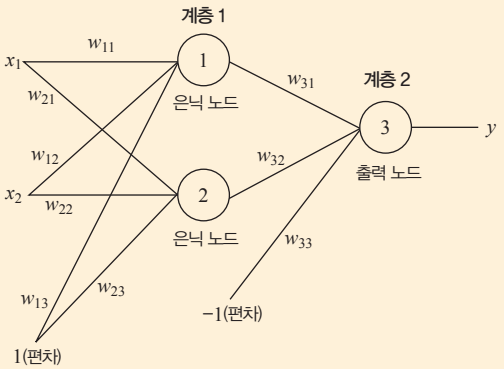

그림 A-11 주어진 초기 조건을 이용한 신경망 설계

논리적 XOR의 입력 벡터와 목표 벡터는 다음과 같다.

$$p = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad t = [0 \quad 1 \quad 1 \quad 0]$$

[그림 A-11]의 설계를 기초로 가중치와 편차는 다음과 같이 나타낼 수 있다.

$$W_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad W_2 = [1 \quad 1]$$

$$B_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad B_2 = [-1]$$

```
nn_XOR1.m
1 –  p = [0 0 1 1;0 1 0 1];
2 –  t = [0 1 1 0];
3 –  net.iw{1,1} = [1 1;1 -1]; ❶
4 –  net.b{1} = [1;1]; ❷
5 –  net.lw{2,1} = [1 1]; ❶
6 –  net.b{2} = -1; ❷
7 –  net = feedforwardnet(2);
8 –  net = configure(net,p,t);
9 –  net.layers{1}.transferFcn = 'purelin'; ❸
10 –  net.layers{2}.transferFcn = 'purelin'; ❸
11 –  net = train(net,p,t);
12    W1 = net.iw{1,1};
13    B1 = net.b{1};
14    W2 = net.lw{2,1};
15    B2 = net.b{2};
16    plotpv(p,t);
17    plotpc(W1,B1);
18    hold on;
19    plotpc(W2,B2);
```

그림 A-12 선형 전달함수를 사용한 논리적 XOR 코딩 결과(nn_XOR1.m)

❶ 가중치 벡터 W_1, W_2를 입력한다.

❷ 편차 B_1, B_2를 입력한다.

❸ 계층 1과 계층 2에 전달함수 purelin을 지정한다.

명령 창에 〉〉nn_XOR1을 입력하면 [그림 A-13]과 같이 여러 번 반복 훈련을 실행한 결과를 보여주는데, 두 집단의 분류는 발생하지 않은 상태이다. 〉〉nn_XOR1을 여러 번 실행하면 결과 그래프가 바뀌지만 두 집단으로 분류되지는 않는다.

초기 조건으로 사용한 가중치와 편차는 반복 훈련 이후 다음과 같은 결과를 보여준다.

$$W_1 = \begin{bmatrix} -1.1213 & 1.6318 \\ 1.7415 & 0.9419 \end{bmatrix}, \quad W_2 = [0.4801 \quad -0.5303]$$

$$B_1 = \begin{bmatrix} 1.9799 \\ 1.9799 \end{bmatrix}, \quad B_2 = [0.4699]$$

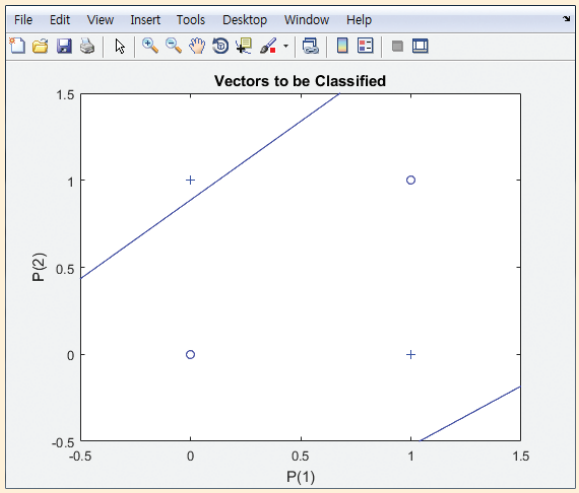

그림 A-13 선형 전달함수를 사용한 논리적 XOR의 실행 결과

MATLAB을 사용하여 [그림 A-11]을 설계하면 다음과 같이 논리적 XOR를 구성하는 신경망의 2계층 구조를 보여준다.

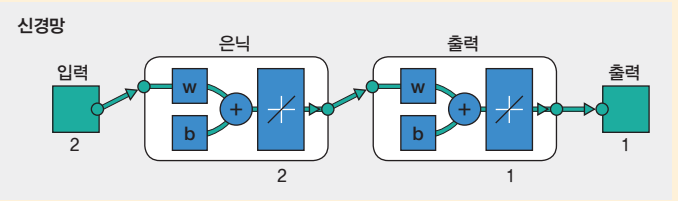

그림 A-14 선형 전달함수를 사용한 2계층 신경망 구조

시그모이드 함수

[예제 A-3]의 결과는 다층 신경망 구조에 선형 전달함수를 사용하여 반복 훈련한 논리적 XOR 시스템이라도 수렴할 수 없음을 보여준다. 입력 벡터를 가장 단순하게 출력으로 전달하는 선형 함수는 신경세포의 편차에 따라서 변경 또는 추가될 수 있다.

반면에 로그 시그모이드(log-sigmoid) 함수는 구간 $(-\infty, +\infty)$에서 구간 $(0, +1)$ 사이의 신경세포 입력을 매핑(mapping)하는 데 사용된다. 이 함수는 역전파 학습에 의해 훈련되는 신경세포에 적합하게 미분 가능한 함수이다. 그러므로 선형 함수에 반하여 시그모이드 함수를 비선형 함수라고 부른다. 시그모이드 함수는 다음과 같이 정의한다.

$$f(z) = \frac{1}{1 + e^{-z}}$$

예제 A-4

M-File | nn_XOR2.m

[예제 A-3]을 비선형 전달함수 logsig를 사용하여 반복 수행하라.

배경 지식

■ logsig

MATLAB에서 시그모이드 함수는 logsig 전달함수를 사용한다. 신경망의 i번째 계층에 logsig 함수를 지정하는 기본 형태는 다음과 같다.

```
net.layers{i}.transferFcn = 'logsig'
```

```
nn_XOR2.m
1 –  p = [0 0 1 1;0 1 0 1];
2 –  t = [0 1 1 0];
3 –  net.iw{1,1} = [1 1;1 -1]; ❶
4 –  net.b{1} = [1;1]; ❷
5 –  net.lw{2,1} = [1 1]; ❶
6 –  net.b{2} = -1; ❷
7 –  net = feedforwardnet(2);
8 –  net = configure(net,p,t);
9 –  net.layers{1}.transferFcn = 'logsig'; ❸
10 –  net.layers{2}.transferFcn = 'logsig'; ❸
11 –  net = train(net,p,t);
12    W1 = net.iw{1,1};
13    B1 = net.b{1};
14    W2 = net.lw{2,1};
15    B2 = net.b{2};
16    plotpv(p,t);
17    plotpc(W1,B1);
18    hold on;
19    plotpc(W2,B2);
```

그림 A-15 비선형 전달함수를 사용한 논리적 XOR 코딩 결과(nn_XOR2.m)

❶ 가중치 벡터 W_1, W_2를 입력한다.

❷ 편차 B_1, B_2를 입력한다.

❸ 계층 1과 계층 2에 전달함수 logsig를 지정한다.

명령 창에 〉〉nn_XOR2를 입력하면 [그림 A-16]과 같이 여러 번 반복 훈련을 실행한 결과를 보여주는데, 두 집단의 분류는 발생하지 않은 상태이다. 〉〉nn_XOR2를 여러 번 실행하면 결과 그래프가 바뀌지만 두 집단으로 분류되지는 않는다.

초기 조건으로 사용한 가중치와 편차는 반복 훈련 이후 다음과 같은 결과를 보여준다.

$$W_1 = \begin{bmatrix} 0.0979 & -2.4435 \\ 0.3966 & -2.1888 \end{bmatrix}, \quad W_2 = [-6.0207 \quad 0.6093]$$

$$B_1 = \begin{bmatrix} 4.3103 \\ 2.3353 \end{bmatrix}, \quad B_2 = [0.4489]$$

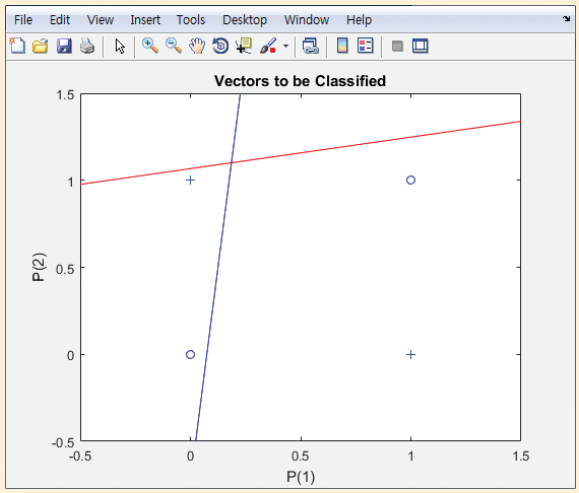

그림 A-16 비선형 전달함수를 사용한 논리적 XOR의 실행 결과

[그림 A-17]은 시그모이드 전달함수를 사용한 논리적 XOR를 구성하는 신경망의 2계층 구조를 보여준다.

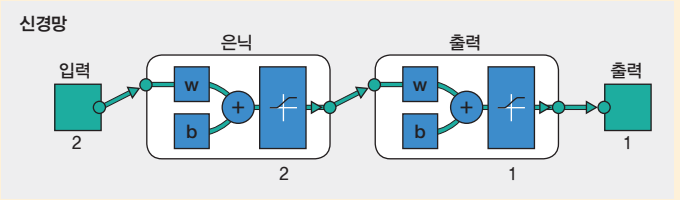

그림 A-17 비선형 전달함수를 사용한 2계층 신경망 구조

[예제 A-4]의 결과도 주어진 초기 조건의 값만 사용하면 아무리 반복 훈련을 해도 수렴되지 않기 때문에 논리적 XOR에 대한 신경망이 두 집단으로 분류되지 않는다.

여기서는 더 이상 소개하지 않지만, 시그모이드 전달함수를 계속 사용하여 논리적 XOR의 출력 결과를 두 집단으로 분류하려면 다음과 같이 매개변수를 조정해야 한다.

1. 초기 조건을 변환한다.
2. 은닉 노드의 수를 증가시킨다.
3. 더 많은 은닉 노드의 수를 가정해야 하는 경우에는 직접 초기 가중치를 선택한다.
4. 수렴하는 오차의 범위는 설정하지 않았지만 오차의 범위를 증가시킨다.
5. 학습률과 모멘트를 조정한다.

선형 전달함수는 비선형 함수보다 좀 더 빠르게 움직인다는 장점이 있지만 논리적 XOR의 수렴이 쉽게 이루어지지 않는다. 반면에 시그모이드 함수를 사용하면 논리적 XOR를 설계한 신경망은 어떤 경우에도 수렴이 가능하다.