인공지능을 위한 머신러닝 알고리즘

1. 머신러닝 개요

Tacademy

CONTENTS

학습 목표

- 인공지능을 달성하기 위한 수단으로써 머신러닝을 이해할 수 있다.
- 머신러닝 알고리즘의 종류와 차이점을 이해할 수 있다.
- 일상생활 속 머신러닝이 사용되고 있는 예제들을 파악할 수 있다.

1. 인공지능과 머신러닝

인공지능의 정의

지능 | **문제를 해결할 수 있는 능력**

지능 작업의 예

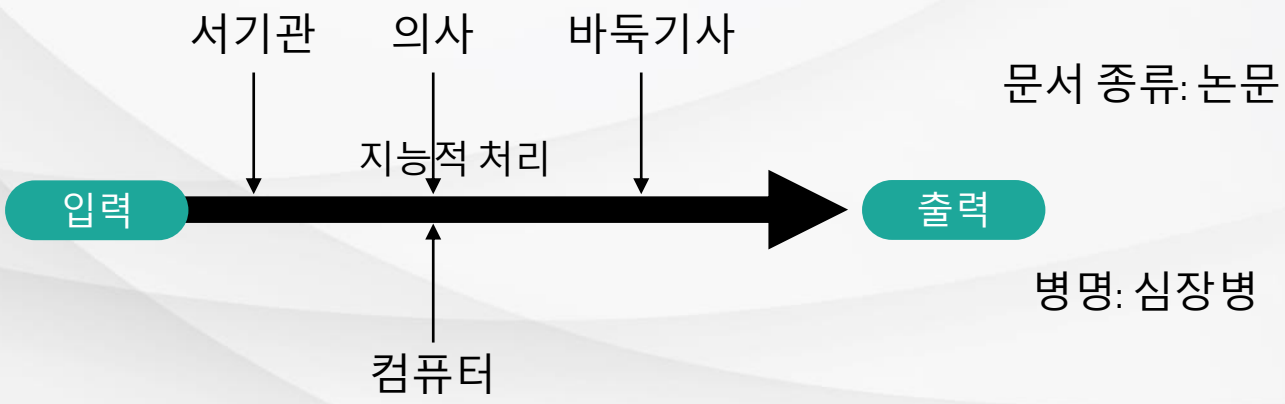
- 문서의 내용에 따라 항목 분류
- 병 진단
- 바둑

인공지능의 정의

인공지능 | 지능 작업을 수행할 수 있는 기계의 능력

In this paper, we proposed the video story QA model
…

환자 1번 기록
나이: 67
성별: 남자
가슴통증 종류: 무증상
혈압: 160mm Hg
혈중 콜레스테롤: 286mg/dl
혈당: < 120mg/dl
…



인공지능의 역사

Alan Turing(1912~1954)

1950 1960 1970 1980 1990 2006 2016~

인공지능의 서막을 열다

낙관의 시대

AI 용어등장('56년)

전문가 시스템

암흑기

('70년중반~'80년초)

암흑기

('80년후반~'90년초)

인공지능,
체스 챔피언을 이김

1997

인공지능,
DARPA 그랜드 챌린지에서 우승

2005

딥러닝

인공지능,
퀴즈 챔피언을 이김

2011

인공지능,
바둑 챔피언을 이김
4차 산업혁명 첫 언급

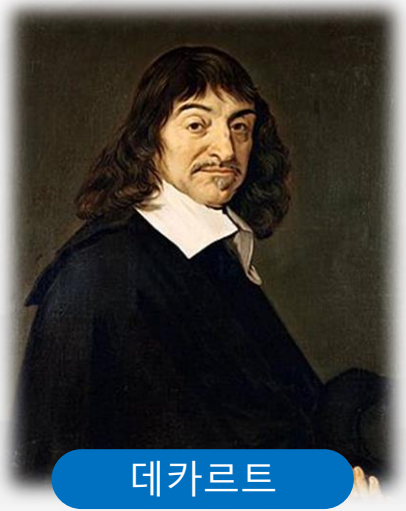
2016

인공지능 구현 방법: 합리주의자 vs. 경험주의자

- 지식의 근원은 어디서부터 왔을까?

- 지식공학 (합리주의자/이성주의자)

  - 대표적 인물: **Mavin Minsky, Noam Chomsky, Descartes**
  - **Top-down**
  - 특정 분야의 전문가나 장인들이 학문 연구, 오랜 실무 경험으로 터득한 지식을 사람이 직접 컴퓨터에게 제공
  - **1980**년대 전문가 시스템

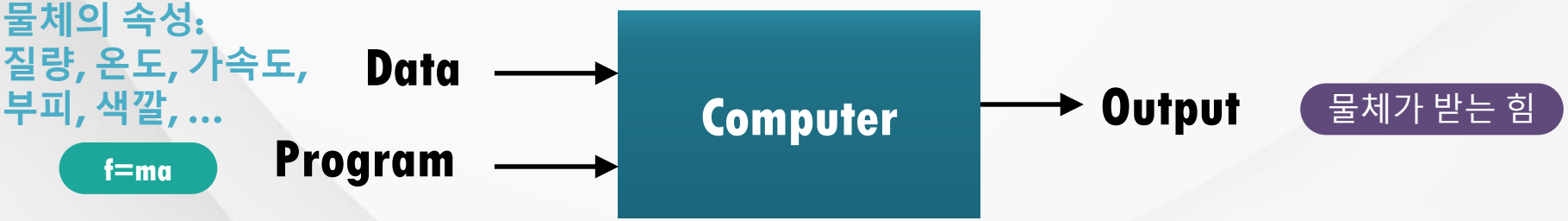


인공지능 구현 방법: 합리주의자 vs. 경험주의자

- 머신러닝 (경험주의자)
  - 대표적 인물: **Andrew ng, Pedro Domingos, David Hume**
  - **Bottom-up**
  - 컴퓨터가 데이터로부터 지식을 직접 학습
  - 현재의 딥러닝

데이비드 흄

인공지능 구현 방법: 지식공학 vs. 기계학습



기계학습과 농작물 재배

씨앗 = 알고리즘

영양분 = 데이터

농부 = 프로그래머

식물 = 프로그램

2. 머신러닝 알고리즘의 분류

2. 머신러닝 알고리즘의 분류

학습의 방법

- **지도학습 (Supervised Learning)**
  - 학습 데이터마다 레이블을 가지고 있음
- **비지도학습 (Unsupervised Learning)**
  - 학습 데이터가 레이블을 가지고 있지 않음
- **준지도학습 (Semi-Supervised Learning)**
  - 학습 데이터가 약간의 레이블을 가지고 있음
- **강화학습 (Reinforcement Learning)**
  - 최종 출력이 바로 주어지지 않고 시간이 지나서 주어지는 경우

지도학습 (Supervised Learning)

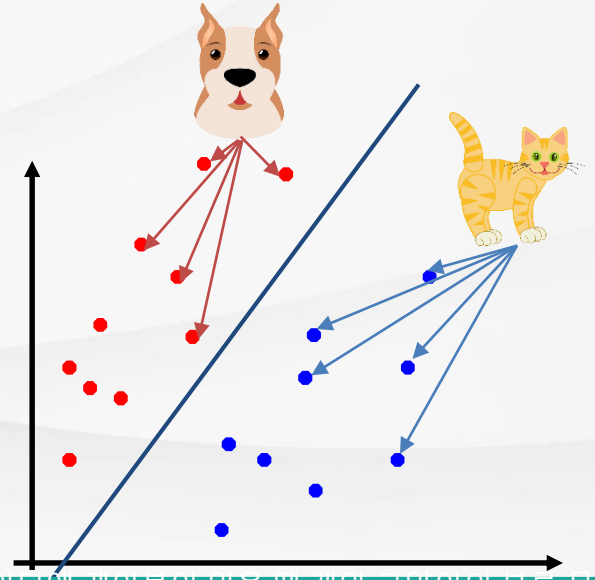
- 주어진 입력-출력 쌍들을 매핑해주는 함수를 학습

- D={X, Y}로부터 F(X)=Y를 만족시키는 함수 F를 학습

  - X의 예> 물체의 속성들. 질량, 온도, 가속도, 부피, 색깔 등.. / 숫자 이미지 파일
  - Y의 예> 물체의 힘/숫자 레이블 (0~9)
  - F의 예> 뉴턴 운동 법칙/숫자 자동 인식기

지도학습 (Supervised Learning)

- 새로운 데이터 X'의 출력을 함수 F를 사용하여 예측
  - F(X)가 이산적 (Discrete)일 때: 분류 (Classification) 문제
  - F(X)가 연속적 (Continuous)일 때: 회귀 (Regression) 문제
  - F(X)가 X의 확률 P(X)일 때: 확률 추정 (Estimation) 문제
- 보다 정확한 학습을 할 수 있음
  - 딥러닝이 대표적인 예
  - 사용할 수 있는 데이터에 한계가 있음
  - 데이터를 생성하는데 비용이 많이 듦



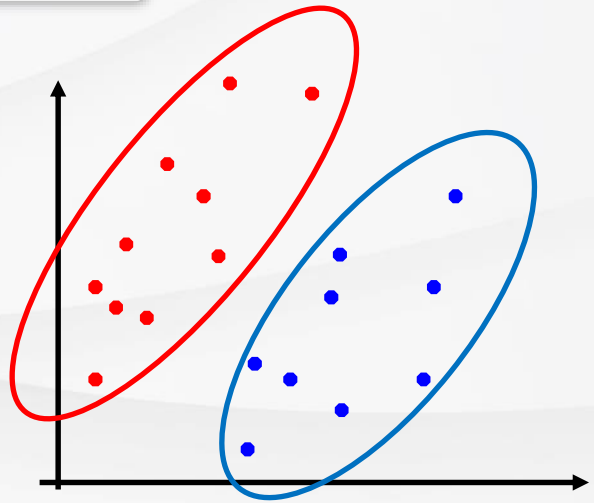
지도 학습의 예: 데이터에 레이블이 있을 때 개와 고양이의 분류 문제

비지도학습 (Unsupervised Learning)

- 입력만 있고 출력은 없는 상태에서 이뤄지는 학습
  - 데이터에 내재되어 있는 고유의 특징을 탐색하고자 함
- D={X}로부터 F(X)=X를 만족시키는 함수 F를 학습
  - X의 예> 유투브 비디오
  - F의 예> 비디오 항목 자동 분류기

비지도학습 (Unsupervised Learning)

- 클러스터링이 주로 사용됨: 비슷한 데이터끼리 묶음
- 지도학습에 비해서 학습하기 어려움
- 우리 주변에 있는 대부분의 데이터는 레이블이 없음
  - 대표적인 예> 비디오
- 앞으로 머신러닝이 풀어야 할 중요한 숙제



비지도 학습의 예: 데이터에 레이블이 없을 때 개와 고양이의 클러스터링 문제

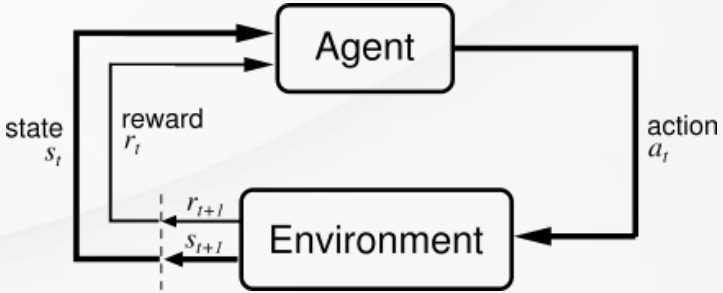
강화 학습 (Reinforcement Learning)

- 결과(출력)이 바로 주어지지 않고 시간이 지나서 주어지는 경우

  - 예> 바둑: 승/패의 결과(**Final Outcome**)이 바둑 기사가(**Agent**) 한 수 두자 마자(**Action**) 주어지지 않고 시간이 지나고 나서 주어짐, 바둑 기사는 매 순간 바둑판의 상황(**State**)을 읽고 어떤 수를 두어야 할지 고민함 (**Agent**가 받는 **Reward**를 최대화하는 쪽으로)
  - 게임, 미로 찾기 등...
  - 어떤 **Action**이 최종 출력에 영향을 주었는지 불명확한 문제에 사용됨

강화 학습 (Reinforcement Learning)

- 컴퓨터가 계산해야 할 중요한 이슈
  - 매 순간 어떤 **Action**을 선택해야 하는가?
  - **State**에 대한 평가



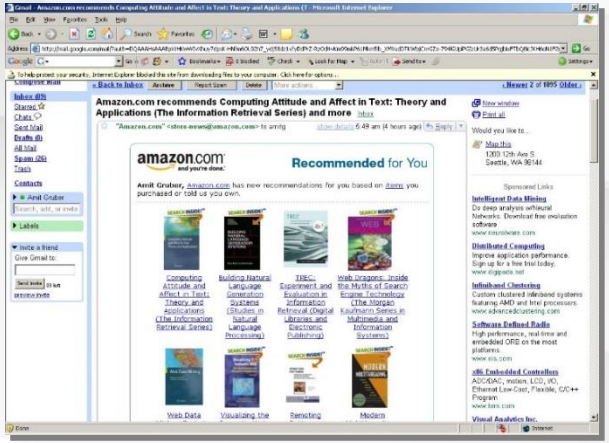
- 우리 주변에 있는 많은 문제들이 강화 학습으로 풀어야 할 문제들
  - 예> 대화 - 상점에서 주문, 고객 상담 등..
  - 좋은 대학에 입학하기 위한 전략은?
  - 축구를 잘 하는 로봇을 만들려면?

3. 일상 생활 속 머신러닝의 예

Netflix의 추천시스템

❖ 온라인 DVD 대여 업체

- 10만 개의 영화 보유
- 고객들이 Neflix가 보유하고 있는 영화를 골고루 볼 수 있도록 관리를 해야 함
- 고객의 취향에 맞는 비디오 추천
- 고객의 평점, 구매 내역을 활용



10만 개 영화

4만 명의 고객

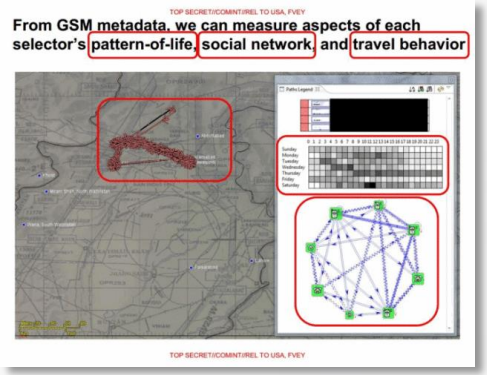
1	3	4			??
1	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

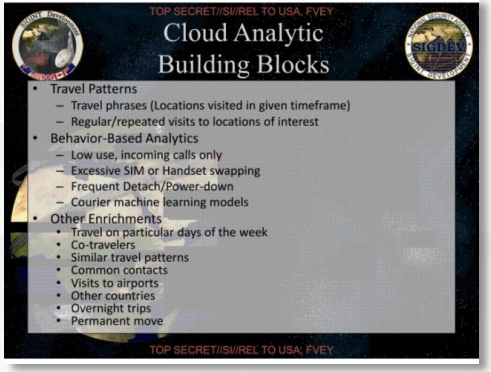
3. 일상 생활 속 머신러닝의 예

Tacademy

미국 국가안보국의 SKYNET

- 파키스탄의 테러리스트을 식별하여 사살하기 위한 프로그램
- 2004년부터 2,500명에서 4,000명 가량의 '식별된' 사람들이 사망
- 5,500만 명의 파키스탄 사람들의 휴대전화 기록을 활용
  - 전화 시간, 길이, 수신자/발신자 정보, GPS, 이동 기록 등..
  - 전화기를 끄거나 SIM 카드를 바꾸면 별도 관리 대상이 됨
- 잘못된 알고리즘 설계로 무고한 많은 사람들이 희생되어 비판 받음





3. 일상 생활 속 머신러닝의 예

Tacademy

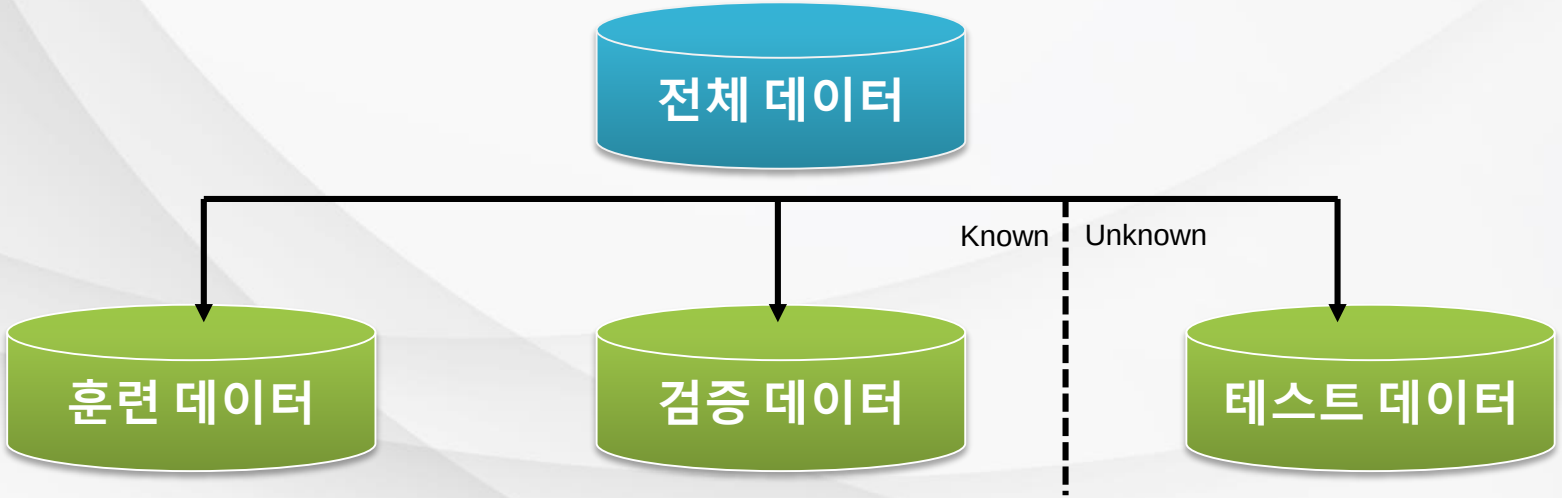
다양한 적용 사례들

응용 분야	적용 사례
인터넷 정보검색	텍스트 마이닝, 웹로그 분석, 스팸필터, 문서 분류, 여과, 추출, 요약, 추천
컴퓨터 시각	문자 인식, 패턴 인식, 물체 인식, 얼굴 인식, 장면전환 검출, 화상 복구
음성인식/언어처리	음성 인식, 단어 모호성 제거, 번역 단어 선택, 문법 학습, 대화 패턴 분석
모바일 HCI	동작 인식, 제스쳐 인식, 휴대기기의 각종 센서 정보 인식, 떨림 방지
생물정보	유전자 인식, 단백질 분류, 유전자 조절망 분석, DNA 칩 분석, 질병 진단
바이오 메트릭스	홍채 인식, 심장 박동수 측정, 혈압 측정, 당뇨치 측정, 지문 인식
컴퓨터 그래픽	데이터기반 애니메이션, 캐릭터 동작 제어, 역운동학, 행동 진화, 가상현실
로보틱스	장애물 인식, 물체 분류, 지도 작성, 무인자동차 운전, 경로 계획, 모터 제어
서비스업	고객 분석, 시장 클러스터 분석, 고객 관리(CRM), 마케팅, 상품 추천
제조업	이상 탐지, 에너지 소모 예측, 공정 분석 계획, 오류 예측 및 분류

4. 머신러닝의 구성 요소

데이터 준비

❖훈련/검증/테스트 데이터 분리



- 모델 학습을 위해 사용하는 데이터

- 모델 검증을 위한 데이터
- 검증 데이터에 대한 성능에 따라 모델을 바꿔보고 통상적으로 성능이 제일 높은 모델을 선택

- 현재 우리가 갖고 있지 않지만 실제 맞닥뜨릴 데이터
- 학습된 모델의 최종 성능 검사를 위해 따로 구별해 놓은 데이터

모델 표현의 방법

❖ 의사 결정 트리

- 기호주의자 (**Symbolists**)
- 귀납적 추론, 철학과 심리학, 논리학에서 아이디어를 얻음

❖ 신경망 기반

- 연결주의자 (**Connectionists**)
- 두뇌를 분석하고 모방하며 신경과학과 물리학에서 영감을 얻음

❖ KNN, 서포트 벡터 머신

- 유추주의자 (**Analogizers**)
- 유사성 판단을 근거로 추정하면서 배우며 심리학과 수학적 최적화의 영향을 받음

모델 표현의 방법

❖ 베이지안 모델

- 베이즈주의자 (**Bayesians**)
- 학습이 확률 추론의 한 형태라고 믿으며 통계학에 뿌리를 두고 있음

❖ 유전 알고리즘

- 진화주의자 (**Evolutionaries**)
- 컴퓨터에서 진화를 모의시험하며 유전학과 진화생물학에 의존

❖ 모델 앙상블

모델 평가 방법

❖ 에러의 제곱 **(Squared Error)**

$$sum\{y - f(x)\}^2$$

❖ 정확도 **(Accuracy)**

- 맞힌 테스트 데이터 개수/전체 테스트 데이터 개수

❖ 우도 **(Likelihood)**

$$log_2(P(y))$$

❖ 정밀도와 재현률**(Precision and Recall)**

- 정보 검색에서 주로 사용

❖ 엔트로피 **(Entropy)**

❖ 사후 확률 **(Posterior Probability)**

학습정리

학습정리

Tacademy

지금까지 [머신러닝 개요]에 대해서 살펴보았습니다.

인공지능과 머신러닝

머신러닝은 인공지능을 구현하기 위한 하나의 방법
지식을 구현하기 위한 두 가지 대립적 방법
지식공학 (합리주의자/이성주의자): 지식을 사람이 직접 컴퓨터에게 제공
머신러닝 (경험주의자): 컴퓨터가 데이터로부터 지식을 직접 학습

머신러닝 알고리즘의 분류

지도학습: 학습 데이터마다 레이블을 가지고 있음
비지도학습: 학습 데이터마다 레이블을 가지고 있지 않음
준지도학습: 학습 데이터가 약간의 레이블을 가지고 있음
강화학습: 최종 출력이 바로 주어지지 않고 시간이 지나서 주어지는 경우

머신러닝의 구성 요소

데이터 준비: 훈련/검증/테스트 데이터 집합으로 분리
모델 표현: 기호/신경망/유추/베이지안/유전 기반 알고리즘 존재
모델 평가 방법: 정확도/에러 제곱/우도 등 각 상황에 맞는 평가 방법 선택

인공지능을 위한 머신러닝 알고리즘

2. 선형 회귀 모델

Tacademy

CONTENTS

학습 목표

- 선형 회귀의 분류 원리에 대해 이해할 수 있다.
- 선형 회귀의 모델 파라미터를 계산할 수 있다.
- 모델이 가정하고 있는 선형성에 대해서 이해할 수 있다.

1. 선형 회귀 모델

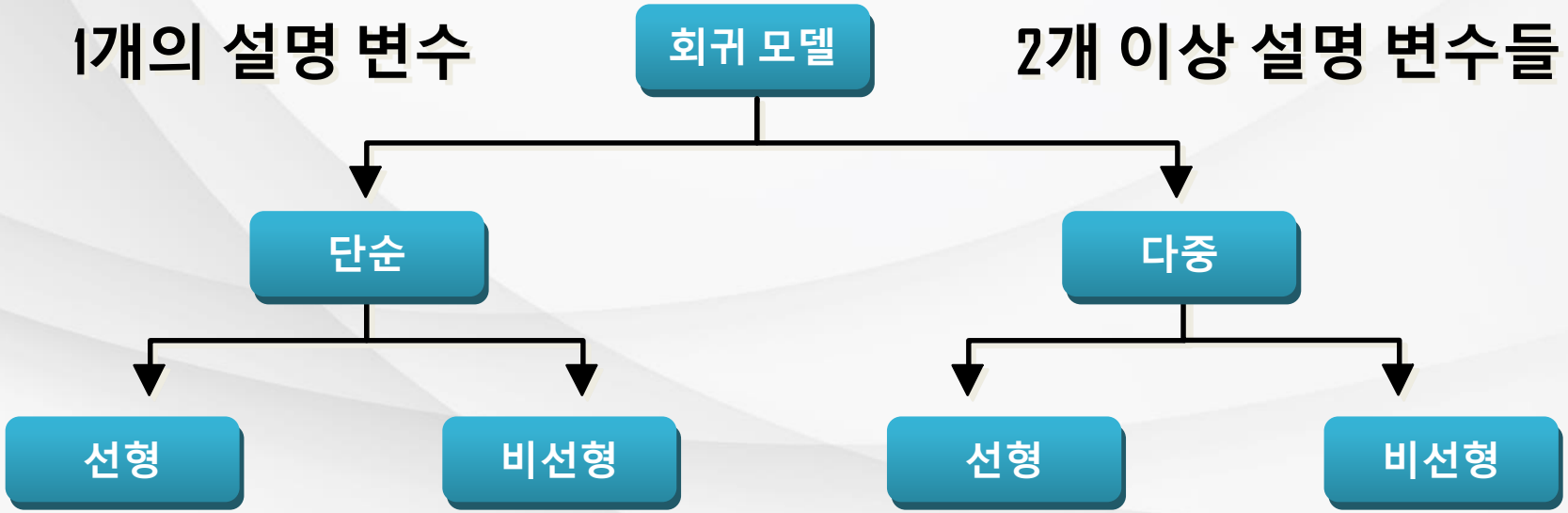
회귀 모델의 정의

- 한 개의 종속 변수(**dependent variable**)와 설명 변수들(**explanatory variable(s)**)과의 관계를 모델링

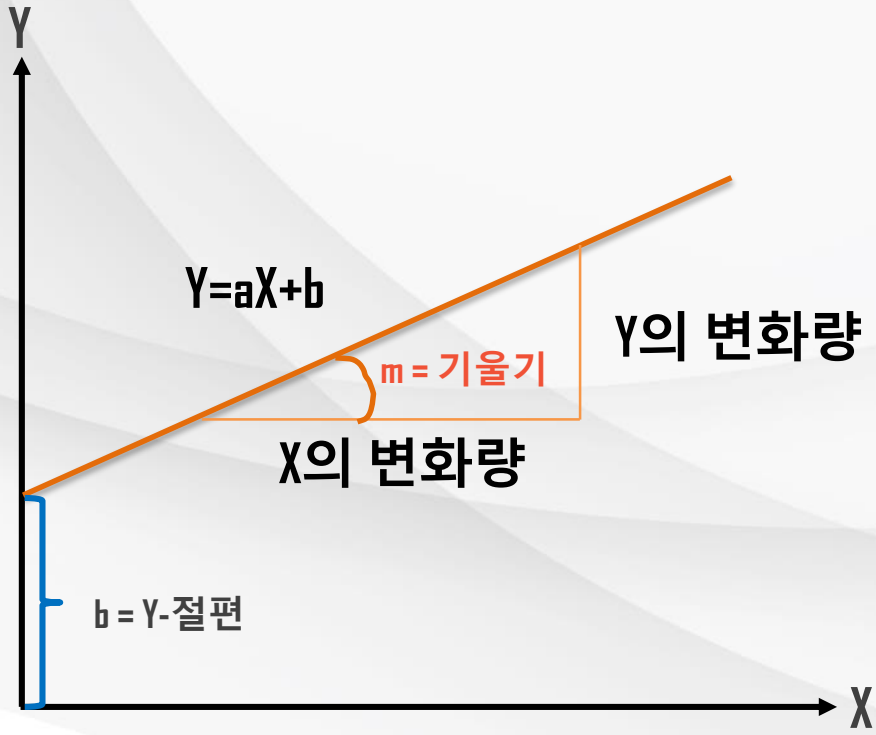
- 관계를 정의하기 위해 방정식 사용
  - 수치적(**numerical**) 종속 변수
  - 한 개 또는 그 이상의 수치적 설명 변수

- 예측 **&** 추정 시에 사용

회귀 모델의 종류들



선형성이란



예> Y=1/2X+3

변수들 사이 관계 - 선형 함수

Y절편

기울기

무작위 에러 (노이즈)

$$Y_i = \beta_o + \beta_i X_i + \varepsilon_i$$

종속 (응답) 변수

예> 와인 등급

독립 (설명) 변수

예> 날씨, 토양, 포도의 품질

회귀 모델의 예측 대상

모집단 (참값)

회귀 모델의 예측 대상

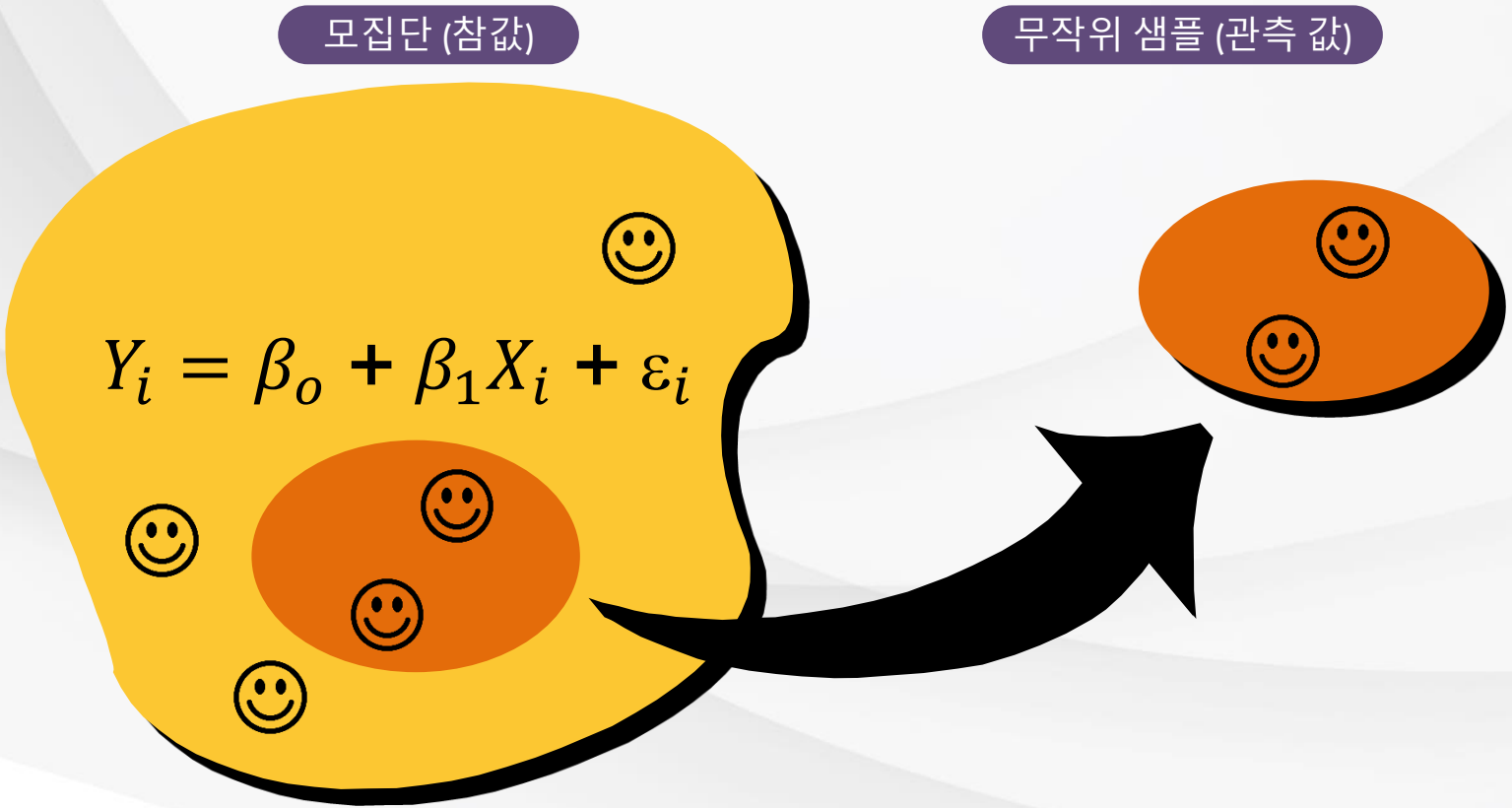
모집단 (참값)

무작위 샘플 (관측 값)

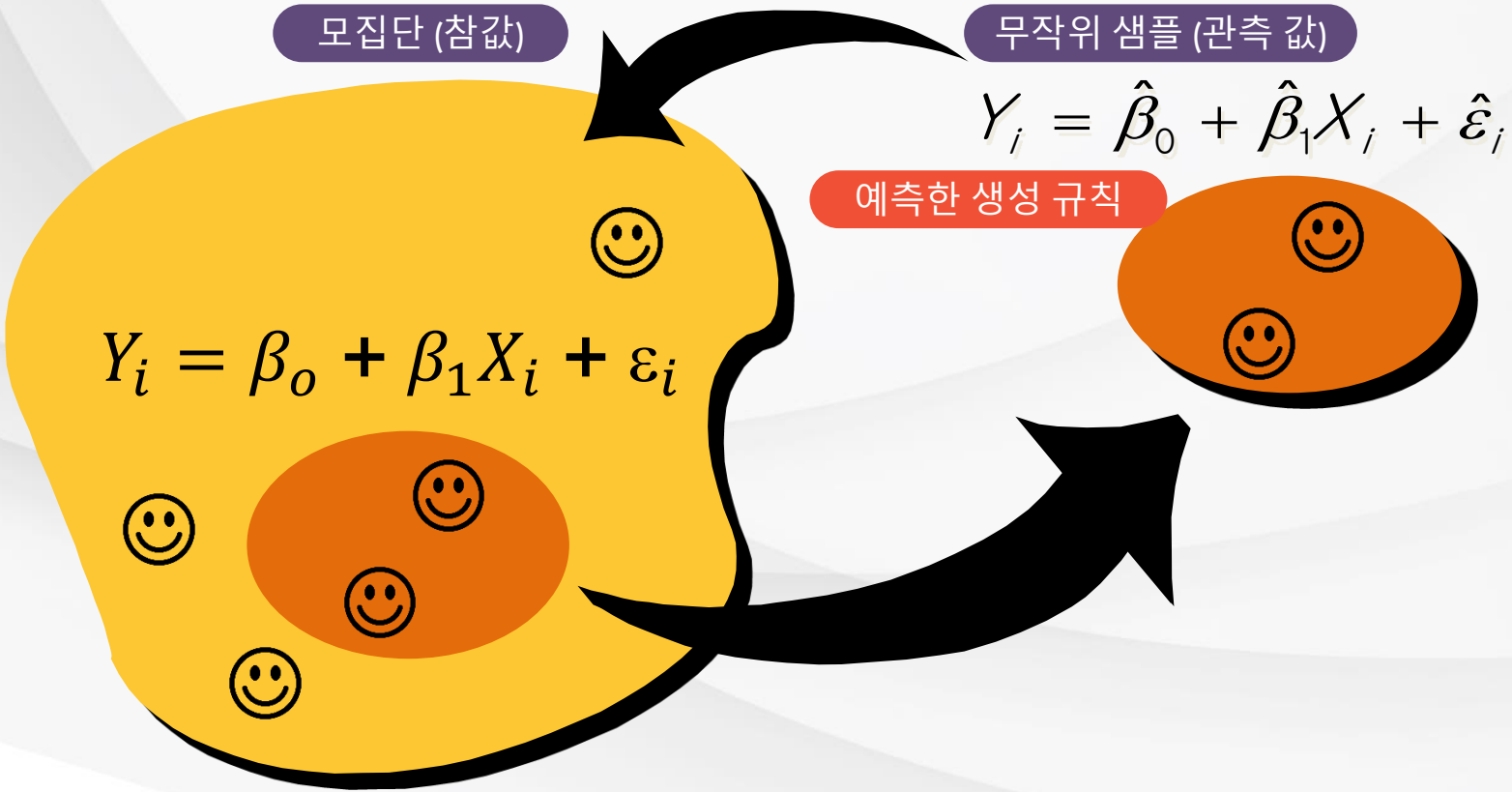
$$Y_i = \beta_o + \beta_1 X_i + \varepsilon_i$$

실제 데이터 생성 규칙

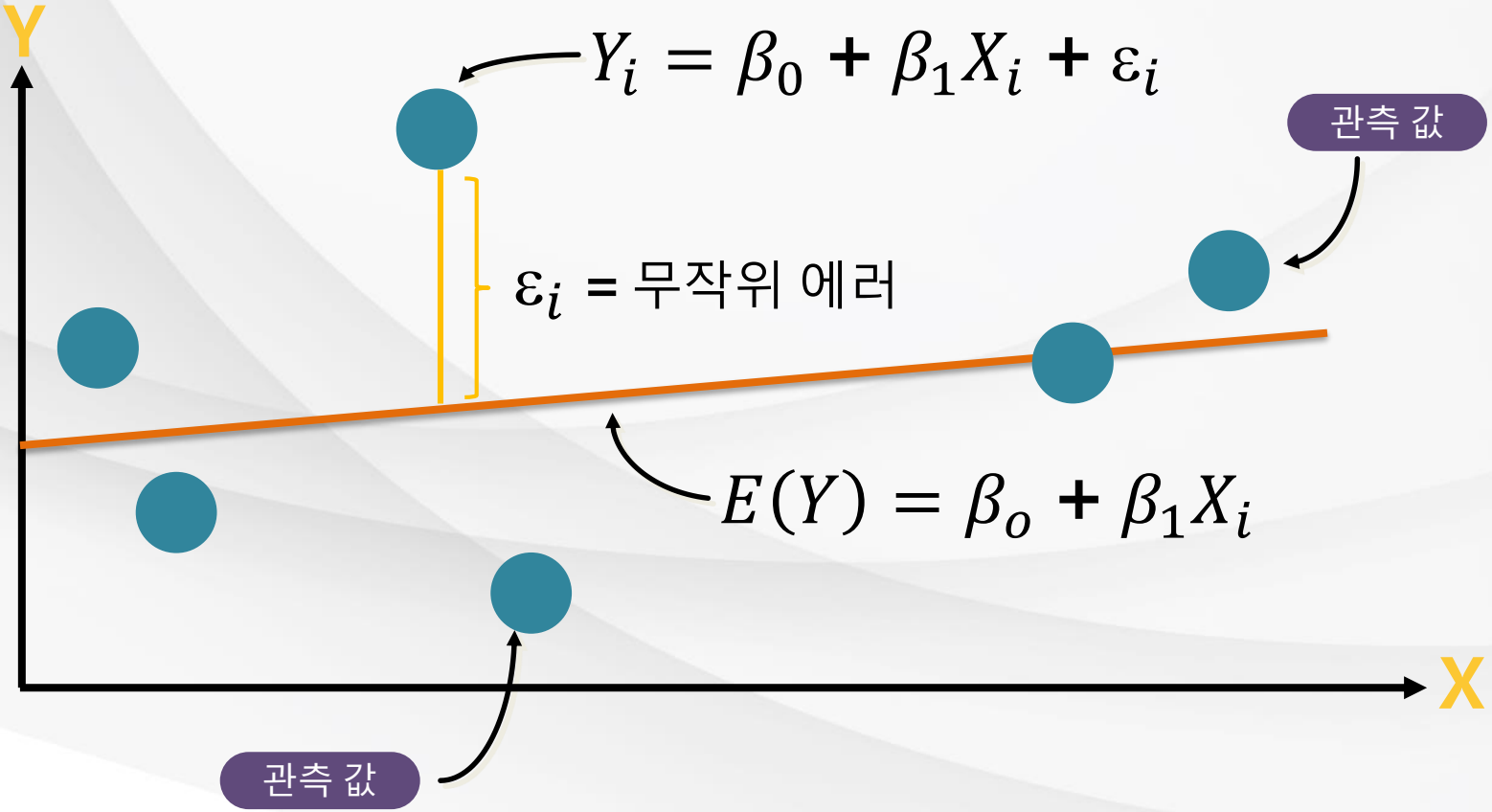
회귀 모델의 예측 대상



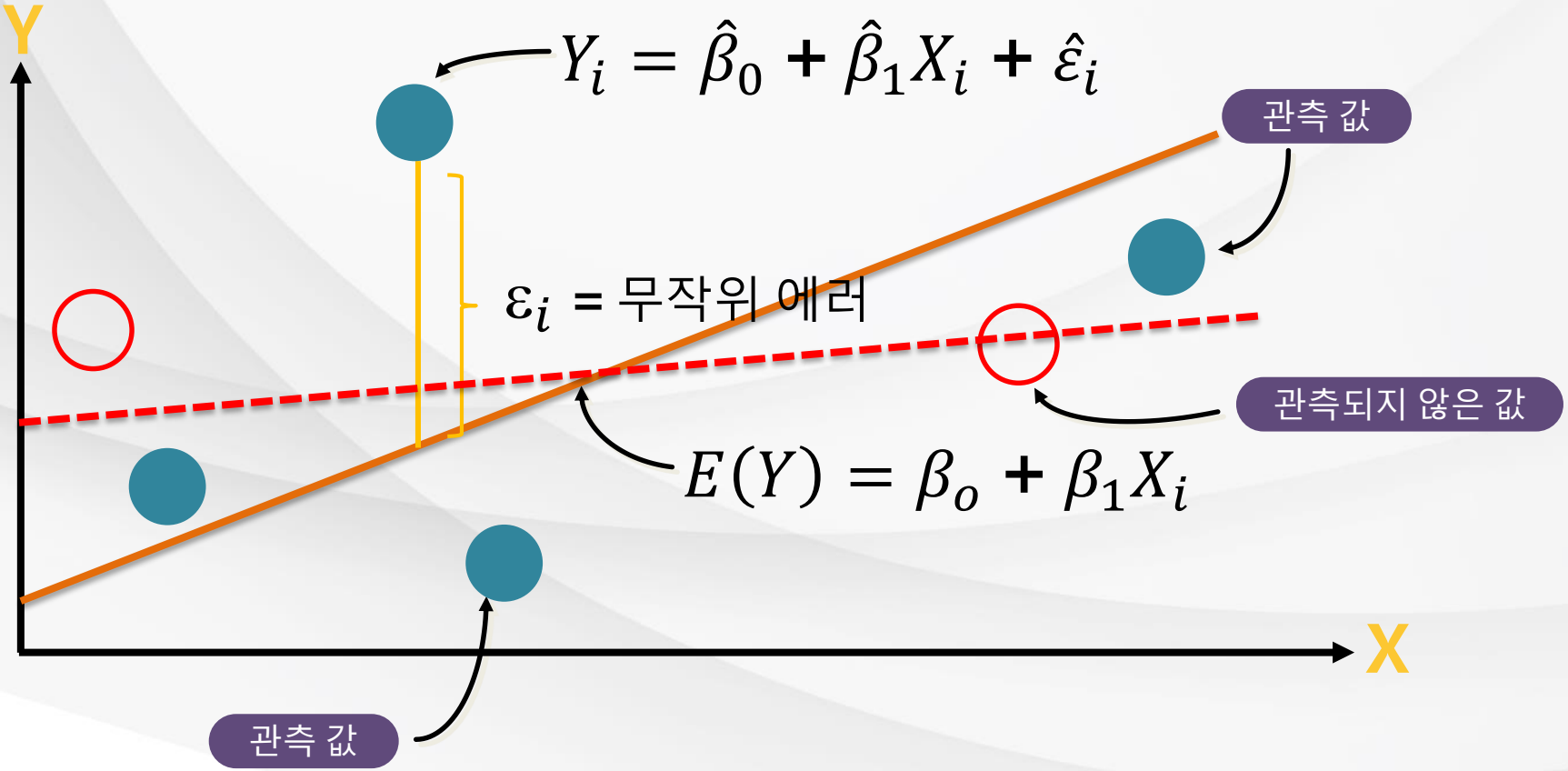
회귀 모델의 예측 대상


모집단 (참값)
무작위 샘플 (관측 값)
$Y_i = \hat{\beta}_0 + \hat{\beta}_1 X_i + \hat{\varepsilon}_i$
예측한 생성 규칙
$Y_i = \beta_o + \beta_1 X_i + \varepsilon_i$

선형 회귀 모델의 확률적 관점



선형 회귀 모델의 확률적 관점



2. 파라미터 예측: 최소 제곱 방법

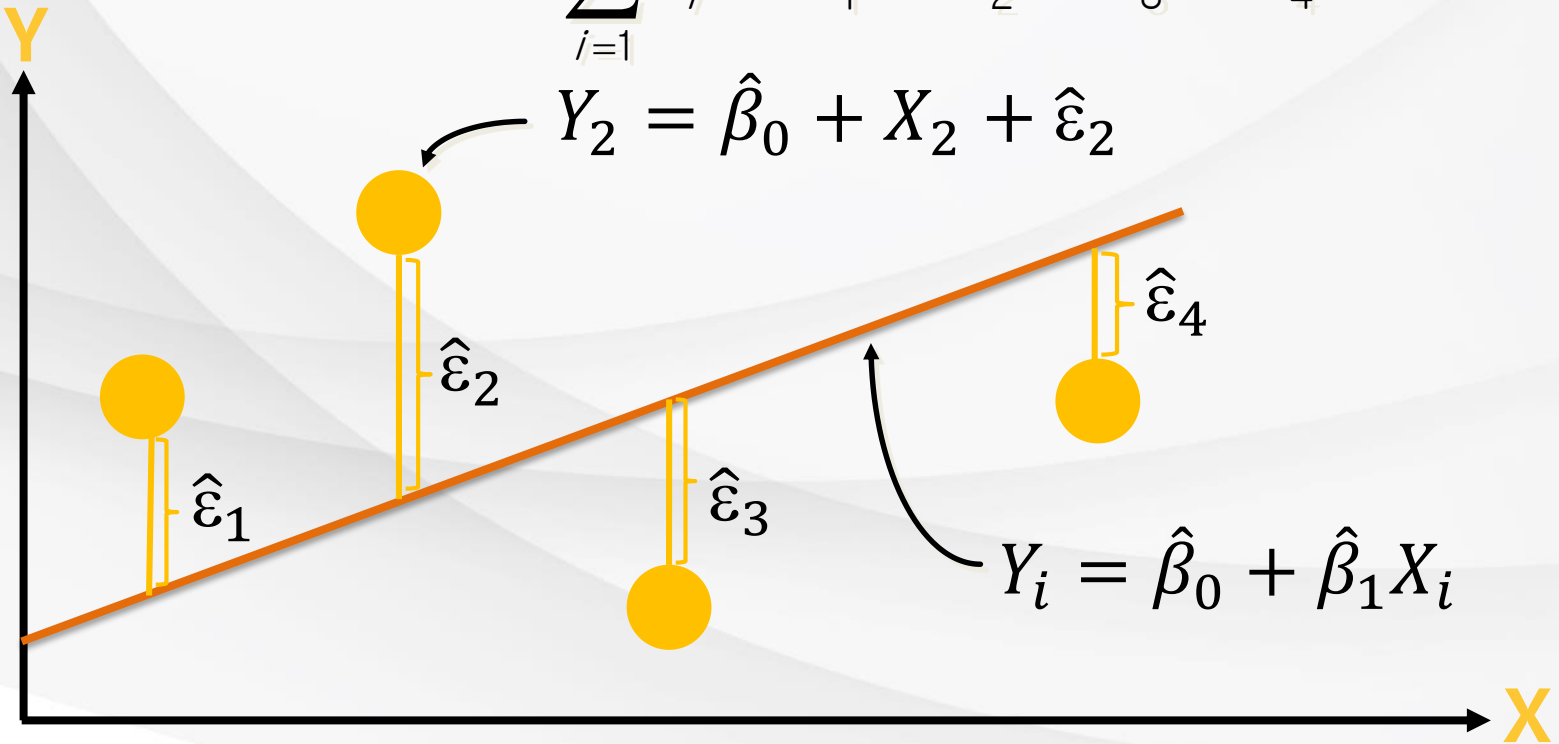
2. 파라미터 예측: 최소 제곱 방법

최소 제곱

- 최적의 모델은 실제 **Y**값과 예측된 **Y**값의 차이 (에러)가 최소가 되어야 함
- 에러의 값은 무조건 양수이어야 하므로 제곱을 시킴
- 최소 제곱 방법은 에러의 제곱의 합 (**Sum of the Squared Errors, SSE**)을 최소화 시킴

최소 제곱

$$\text{LS minimizes} \sum_{i=1}^{n} \hat{\varepsilon}_i^2 = \hat{\varepsilon}_1^2 + \hat{\varepsilon}_2^2 + \hat{\varepsilon}_3^2 + \hat{\varepsilon}_4^2$$



2. 파라미터 예측: 최소 제곱 방법

최소 제곱의 해

❖ 예측 방정식

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

❖ 기울기

$$\hat{\beta}_1 = \frac{SS_{xy}}{SS_{xx}} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

❖ Y-절편

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Y 절편 구하기

$$\sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_i \right)^2$$

$$0 = \frac{\partial \sum \varepsilon_i^2}{\partial \beta_0} = \frac{\partial \sum \left(y_i - \beta_0 - \beta_1 x_i \right)^2}{\partial \beta_0}$$

$$= -2\left(n\bar{y} - n\beta_0 - n\beta_1 \bar{x} \right)$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

2. 파라미터 예측: 최소 제곱 방법

Tacademy

- 기울기 구하기

$$0 = \frac{\partial \sum \varepsilon_i^2}{\partial \beta_1} = \frac{\partial \sum (y_i - \beta_0 - \beta_1 x_i)^2}{\partial \beta_1}$$

$$= -2 \sum x_i (y_i - \beta_0 - \beta_1 x_i)$$

$$= -2 \sum x_i (y_i - \bar{y} + \beta_1 \bar{x} - \beta_1 x_i)$$

$$\beta_1 \sum x_i (x_i - \bar{x}) = \sum x_i (y_i - \bar{y})$$

$$\beta_1 \sum (x_i - \bar{x})(x_i - \bar{x}) = \sum (x_i - \bar{x})(y_i - \bar{y})$$

$$\hat{\beta}_1 = \frac{SS_{xy}}{SS_{xx}}$$

2. 파라미터 예측: 최소 제곱 방법

기울기와 y-절편의 의미

1. 기울기 ($\hat{\beta}_1$) - 추정된 Y 는 X 가 1 단위 증가할 때마다 $\hat{\beta}_1$ 만큼 변화

- 만약 $\hat{\beta}_1 = 2$인 경우, Y 는 X 가 1 단위 증가할 때마다 2씩 증가

2. Y-절편 ($\hat{\beta}_0$) - X = 0인 경우 Y 의 평균 값

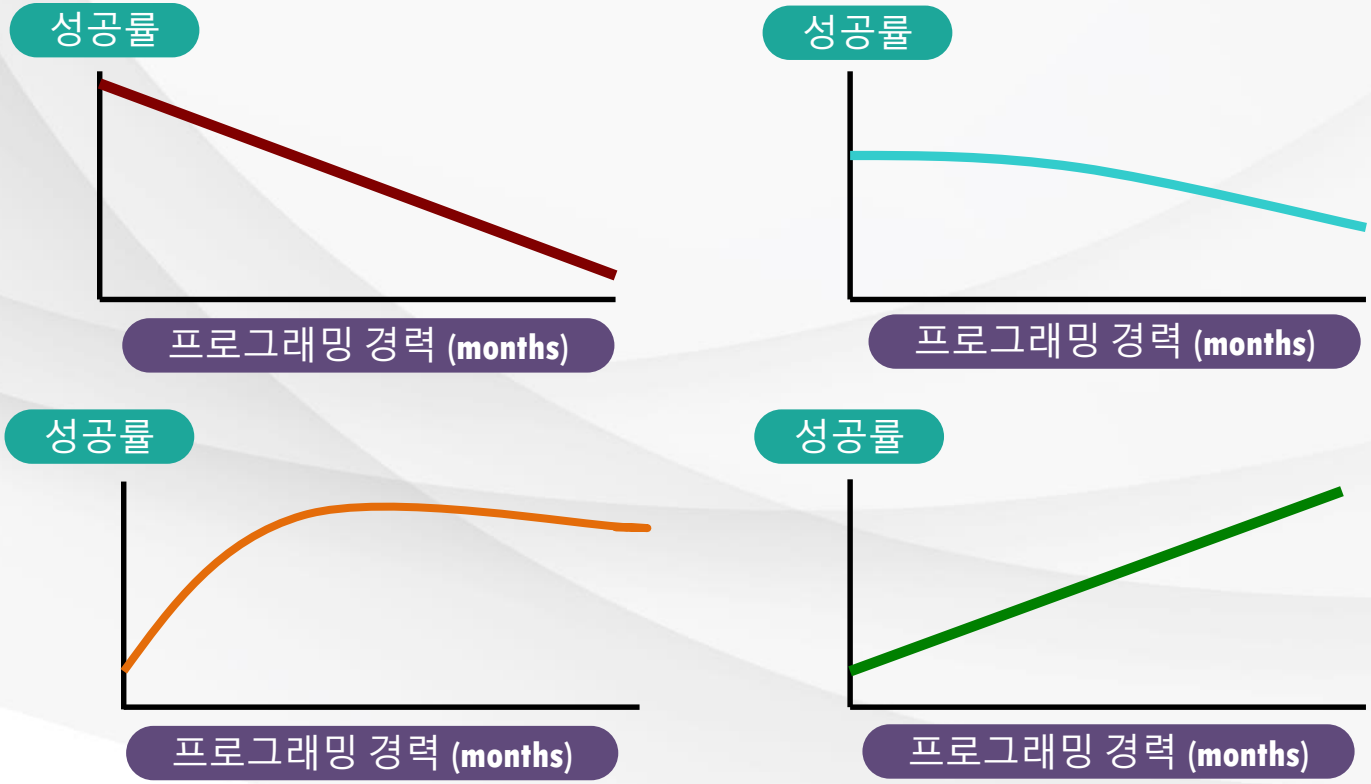
- 만약 $\hat{\beta}_0 = 4$인 경우, X 가 0일 때 Y 의 평균값은 4

3. 선형 회귀 모델로는 안 풀리는 문제들

3. 선형 회귀 모델로는 안 풀리는 문제들

Tacademy

- 무엇이 더 좋은 모델일까?
  - ❖ 예> 프로그래밍 숙제의 성공 여부 예측

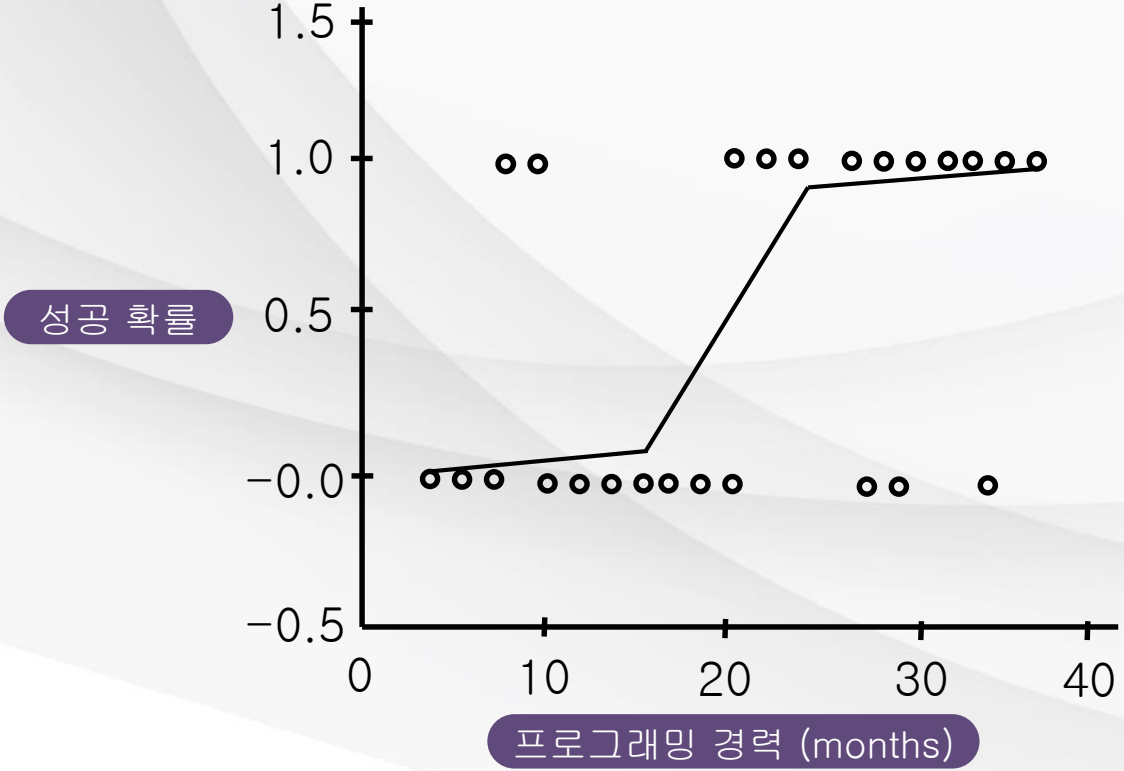


3. 선형 회귀 모델로는 안 풀리는 문제들

Tacademy

선형성의 한계

❖ 예> 프로그래밍 숙제의 성공 여부 예측



학습정리

지금까지 [선형 회귀 모델]에 대해서 살펴보았습니다.

선형 회귀 모델

한 개의 종속 변수와 다수의 설명 변수들 사이의 관계를 선형 방정식으로 모델링

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

파리미터 예측: 최소 제곱 방법

$$\hat{\beta}_1 = \frac{SS_{xy}}{SS_{xx}} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \qquad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

선형 문제로는 풀리지 않는 문제들

종속변수와 설명변수 사이 비선형 관계 존재

인공지능을 위한 머신러닝 알고리즘

3. 로지스틱 회귀 모델

Tacademy

CONTENTS

학습 목표

- 로짓(logit) 함수를 이해할 수 있다.
- 로지스틱 회귀의 분류 원리에 대해 이해할 수 있다.
- 로지스틱 회귀 모델의 파라미터를 구할 수 있다.

1. 로짓(logit) 함수

오즈 (odds)

- 어떠한 사건의 확률이 **p**일 때, 그 사건의 오즈는 다음과 같이 계산

  - **odds = p / (1–p)**

예시

		비만		
		Yes	No	Total
혈중 콜레스테롤	Normal	402	3614	4016
	High	101	345	446
		503	3959	4462

- 혈중 콜레스테롤이 정상인 그룹에서 비만인 경우의 오즈

  - 비만일 확률 / (1–비만일 확률) = (402/4016) / (1 - (402/4016)) = 0.1001 / 0.8889 = 0.111

1. 로짓(logit) 함수

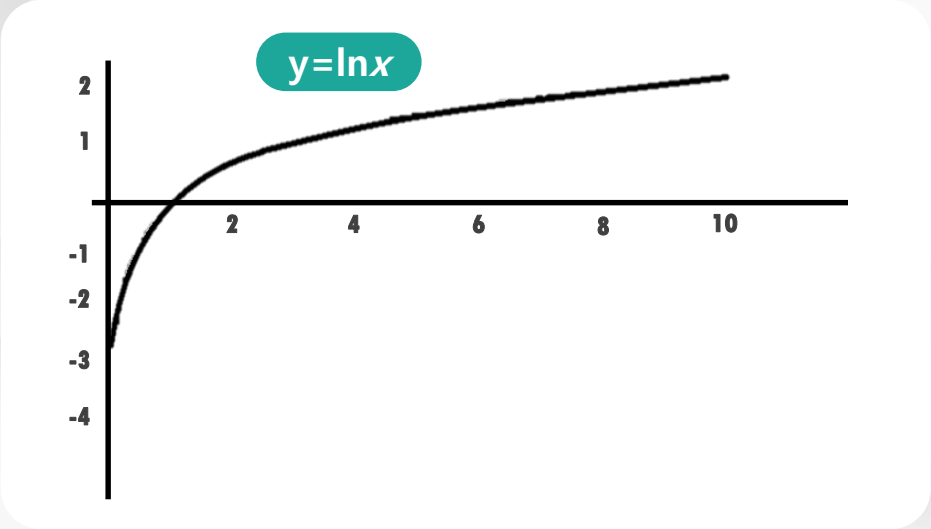
오즈 (odds)

- 혈중 콜레스테롤이 정상인 그룹에서 비만이 아닌 경우의 오즈
  - **0.8999/0.1001 = 8.99**
- 혈중 콜레스테롤이 높은 그룹
  - **odds**(비만) **= 101/345 = 0.293**
  - **odds**(비만이 아님) **= 345/101 = 3.416**
- 혈중 콜레스테롤이 정상에서 높은 수치로 갈 때,
  비만인 경우의 오즈는 약 세 배 증가
  - **odds** 비율**: 0.293/0.111 = 2.64**
  - 혈중 콜레스테롤이 높을 때**, 2.64** 배 더 비만이 되기 쉬움

1. 로짓(logit) 함수

로짓 변형

❖ 로짓은 오즈의 자연로그



$$\text{logit}(p) = \ln(\text{odds}) = \ln\left(p/(1-p)\right)$$

2. 로지스틱 회귀 모델

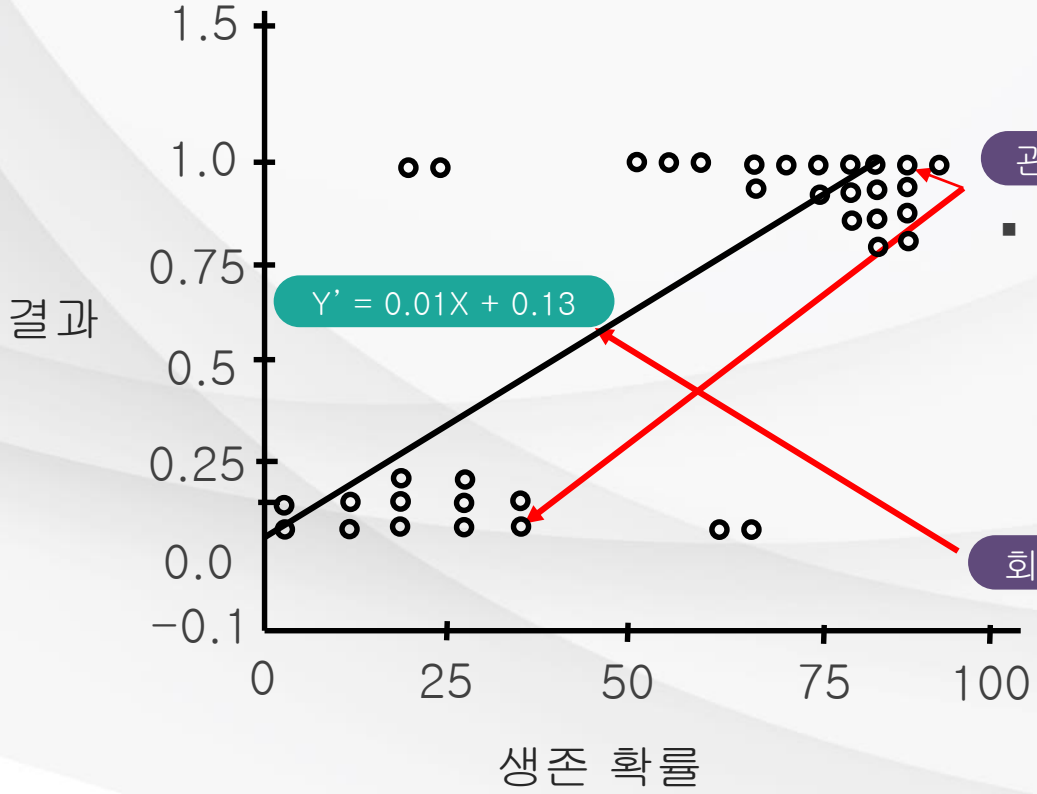
로지스틱 회귀 분석이 사용되는 예

- 종속 변수의 값을 **0**또는 **1**로 (이진 변수로) 표현할 수 있는 경우

예) 약물 치료 후 환자의 반응 예측

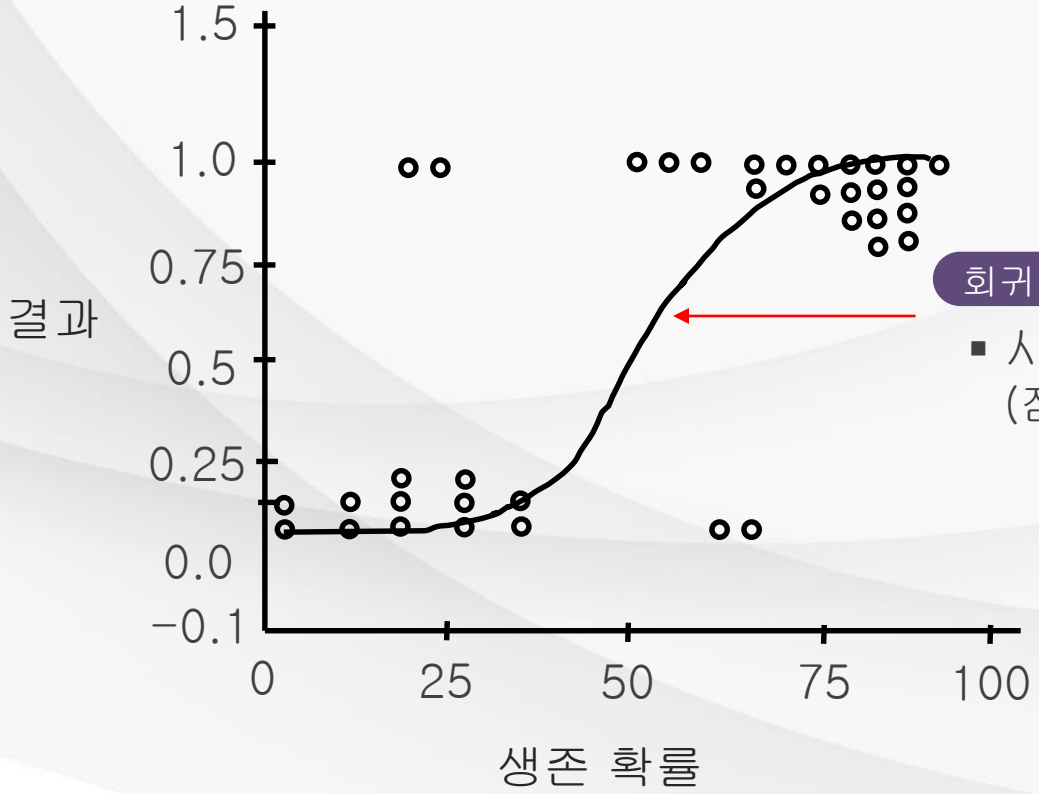
약물 치료에 대한 환자의 반응(종속 변수)을 예측하고자 할 때,
약물 치료 적용 후 환자가 살아남은 경우 **1**로,
살아남지 못한 경우를 **0**으로 표현할 수 있음

선형 회귀 모델을 사용할 경우



- 각 환자-약물 부합점수마다 점의 개수는 생존한 환자의 수를 의미

더 나은 솔루션



회귀 모델의 예측 커브

- 시그모이드 함수
  (점근적으로 y=0 과 y=1에 근사)

로지스틱 회귀 모델

- 로지스틱 회귀 모델의 방정식

$$\text{logit}(p) = \beta_0 + \beta_1 X$$

- 오즈의 로그(로짓)은 설명변수 **x**와 선형적인 관계

- 일반적 선형 회귀 문제처럼 접근 가능

종속 변수 p 값 구하기

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X$$

$$\Leftrightarrow \frac{p}{1-p} = e^{\beta_0 + \beta_1 X}$$

$$\Leftrightarrow p = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

p는 시그모이드 함수

β_1 값 해석

- **Let**
  - **odds1 = X의 odds (p/(1−p))**
  - **odds2 = X + 1의 odds**
- **Then**

$$\frac{odds2}{odds1} = \frac{e^{b_0 + b_1(X+1)}}{e^{b_0 + b_1 X}}$$

$$= \frac{e^{(b_0 + b_1 X) + b_1}}{e^{b_0 + b_1 X}} = \frac{e^{(b_0 + b_1 X)} e^{b_1}}{e^{b_0 + b_1 X}} = e^{b_1}$$

- **X**가 각 단위 값 마다 증가할 때,
  예측된 **odds**의 비율이 **e**의 기울기(β_1) 제곱만큼 증가함을 의미

Logit 변환의 의미

❖ 선형회귀에 더 적절한 함수를 도출

$$\text{Logit}(P) = \log \text{odds} = \ln\left(\frac{P}{1-P}\right)$$

확률	0	$\frac{1}{2}$	1
Odds	0	1	$+\infty$
Logit	$-\infty$	0	$+\infty$

3. 로지스틱 회귀 모델의 파라미터 추정

3. 로지스틱 회귀 모델의 파라미터 추정

최대 우도 추정법이란?

❖ 동전 던지기 문제

- 앞/뒤가 나올 확률이 공정하지 않은 (biased) 동전이 있을 경우, 동전의 앞면이 나올 확률 head(p)를 계산하고자 함
- 이 때, p는 unknown 파라미터
- 동전을 10번 던져서 앞면이 7번 나왔다고 하자. 이 때, p의 값으로 추정할 수 있는 값 중 가장 최선은 무엇일까?
- 데이터에 기반하여 0.7로 예측

최대 우도 추정법이란?

❖ 동전 던지기 문제

- 동전을 **10**번 던졌을 때 앞면이 **10**번 나온 횟수는 **N=10**이고 **p=unknown**인 **binomial** 랜덤 변수

$$\therefore P(7heads) = \binom{10}{7} p^7 (1-p)^3 = \frac{10!}{7! * 3!} p^7 (1-p)^3$$

- 알지 못하는 파라미터 **p**에 대해서 데이터를 관측할 확률을 제공

3. 로지스틱 회귀 모델의 파라미터 추정

Tacademy

최대 우도 추정법이란?

- 이 때, 데이터의 확률을 가장 높이는 **p**의 값을 찾고자 함 (또는 우도함수를 가장 높이는 **p**)
- 즉, 데이터를 가장 잘 설명할 수 있는 파라미터 **p**를 찾고자 함
- 어떻게 찾을 수 있을까?
  - 함수에 **log**를 씌움 : 곱셈을 덧셈으로 바꿈으로써 미분을 쉽게 함
  - **p**에 대해서 미분 계산 : **p**의 변화량에 대한 함수의 변화량 계산
  - 미분 값을 **0**으로 설정하고 **p**를 계산 : 함수의 최대 값이 되는 곳은 미분 값이 **0**
- 파라미터 대입해서 확률 계산해보기

$$\textbf{Likelihood} = \binom{10}{7}(.7)^7(.3)^3 = 120(.7)^7(.3)^3 = .267$$

$$\log Likelihood = \log\frac{10}{7!*3!} + 7\log p + 3\log(1-p)$$

$$\frac{d}{dp}\log Likelihood = 0 + \frac{7}{p} - \frac{3}{1-p}$$

$$\frac{7}{p} - \frac{3}{1-p} = 0$$

$$\frac{7(1-p)-3p}{p(1-p)} = 0$$

$$7(1-p) = 3p$$

$$7 - 7p = 3p$$

$$7 = 10p$$

$$p = \frac{7}{10}$$

3. 로지스틱 회귀 모델의 파라미터 추정

모수(β_i) 추정 방법

- 최대 우도(**maximum likelihood**) 추정법 이용
  - 우도함수(**likelihood function**)

$$L(x_i;\beta_i)=\prod_{i=1}^{n}\left(\frac{1}{1+\exp(-\eta)}\right)^{y_i}\left(1-\frac{1}{1+\exp(-\eta)}\right)^{1-y_i}$$

$$\eta=\left(\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_px_p\right)$$

  - 우도함수를 최대화하는 최대 우도 추정법(**MLE**)을 이용하여 $\hat{\beta}_i$을 수치적(**numerical**) 방법으로 산출

$$Max\left[L(x_i;\beta_i)\right]\ \Rightarrow\ \beta_i\,??$$

로지스틱 회귀 모델 예제

❖ 나이에 따른 동맥 심장 질환 확률 예측하기

데이터

	55세 이상	55세 미만
동맥 심장 질환 유	21	22
동맥 심장 질환 무	6	51

로지스틱 모델

$$\log\left(\frac{P(D)}{1-P(D)}\right) = \alpha + \beta_1 X_1$$

$$X_1 = \begin{cases} 1 & \text{if age} >= 55 \\ 0 & \text{if age} < 55 \end{cases}$$

우도함수 식

$$L(\alpha, \beta_1) = \left(\frac{e^{-\alpha-\beta_1}}{1+e^{-\alpha-\beta_1}}\right)^6 \left(\frac{1}{1+e^{-\alpha-\beta_1}}\right)^{21} \left(\frac{e^{-\alpha}}{1+e^{-\alpha}}\right)^{51} \left(\frac{1}{1+e^{-\alpha}}\right)^{22}$$

로지스틱 회귀 모델 예제

$$L(\alpha,\beta_1)=\left(\frac{e^{-\alpha-\beta_1}}{1+e^{-\alpha-\beta_1}}\right)^{6}\left(\frac{1}{1+e^{-\alpha-\beta_1}}\right)^{21}\left(\frac{e^{-\alpha}}{1+e^{-\alpha}}\right)^{51}x\left(\frac{1}{1+e^{-\alpha}}\right)^{22}$$

로그 우도함수

$$\therefore \log L(\alpha,\beta_1)=$$

$$6(-\alpha-\beta_1)-6\log(1+e^{-\alpha-\beta_1})+0-21\log(1+e^{-\alpha-\beta_1})-$$

$$51\alpha+51\log(1+e^{-\alpha})+0-22\log(1+e^{-\alpha})$$

로지스틱 회귀 모델 예제

로그 우도함수

$$\therefore \log L(\alpha, \beta_1) =$$

$$6(-\alpha - \beta_1) - 6\log(1 + e^{-\alpha-\beta_1}) + 0 - 21\log(1 + e^{-\alpha-\beta_1}) -$$

$$51\alpha + 51\log(1 + e^{-\alpha}) + 0 - 22\log(1 + e^{-\alpha})$$

미분식

$$\frac{d[\log L(\beta_1)]}{d\beta_1} =$$

$$-6 + \frac{6e^{-\alpha-\beta_1}}{1 + e^{-\alpha-\beta_1}} + \frac{21e^{-\alpha-\beta_1}}{1 + e^{-\alpha-\beta_1}} = 0$$

$$\frac{d[\log L(\alpha)]}{d\alpha} =$$

$$51 - \frac{51e^{-\alpha}}{1 + e^{-\alpha}} - \frac{22e^{\alpha-}}{1 + e^{-\alpha}} = 0$$

학습정리

지금까지 [로지스틱 회귀 모델]에 대해서 살펴보았습니다

로짓(logit) 함수

로짓은 오즈의 자연로그

$$\text{odds} = p / (1-p)$$

$$\text{logit}(p) = \ln(\text{odds}) = \ln(p/(1-p))$$

로지스틱 회귀 모델

로짓과 설명변수 X를 선형적인 관계로 모델링
종속변수 p와 X는 비선형 관계

$$\text{logit}(p) = b_0 + b_1 X$$

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

로지스틱 회귀 모델의 파라미터 추정

우도함수 사용

$$L(x_i; \beta_i) = \prod_{i=1}^{n} \left(\frac{1}{1+\exp(-\eta)} \right)^{y_i} \left(1 - \frac{1}{1+\exp(-\eta)} \right)^{1-y_i}$$

인공지능을 위한 머신러닝 알고리즘

4. 결정 트리

Tacademy

CONTENTS

학습 목표

- 의사 결정 트리의 분류 원리를 이해할 수 있다.
- Occam의 면도날 원칙이 결정 트리 구축 과정에 미치는 영향을 이해할 수 있다.
- Bagging 기법을 이해할 수 있다.

1. 의사 결정 트리와 분류 문제

분류 (classification)

- 훈련 데이터의 특성 값 / 클래스 정보를 학습한 뒤, 새롭게 주어지는 테스트 데이터부터는 특성 값만으로 클래스를 예측하는 것

예> 날씨 정보를 바탕으로 장대방이 데이트를 허락해줄지 말지 예측

- 필요 사항
  - 특성 값 (**attribute value**) : 고정된 집합의 특성들로 표현될 수 있어야 함

    예> 더움, 따뜻함, 추움
  - 클래스 (**class, target values**) : 이산 출력 값을 가져야 함

    예> 예/아니오, 다중클래스
  - 충분한 데이터 : 결정 트리에 주어질 충분한 훈련 예제

분류 문제의 간단한 예제

다음 주에 경기하는 국내 축구팀의 A팀과 B팀의 경기에서 A팀이 이길지 혹은 질지 예측하는 모델을 만들어보자

- 가능한 특성 값들
  - 경기 장소 : 홈 / 어웨이
  - 경기 시간 : 오후 5시 / 7시 / 9시
  - S선수의 포지션 : 공격수 / 미드필더
  - B팀 K선수의 출전 여부 : 예 / 아니오
  - …

1. 의사 결정 트리와 분류 문제

Tacademy

데이터

장소	시간	K선수 출전	S선수 포지션	R선수 출전	날씨	결과
홈	7시	예	미드필더	예	맑음	승리
홈	7시	예	공격수	아니오	비	승리
어웨이	7시	예	공격수	예	맑음	승리
홈	5시	아니오	공격수	아니오	맑음	패배
어웨이	9시	예	공격수	예	비	패배
어웨이	7시	아니오	미드필더	예	맑음	승리
홈	7시	아니오	공격수	아니오	맑음	패배
홈	7시	예	미드필더	아니오	맑음	승리
어웨이	7시	예	미드필더	아니오	비	승리
홈	9시	아니오	공격수	아니오	비	패배

1. 의사 결정 트리와 분류 문제

Tacademy

데이터

- 다음 주 경기는 어웨이, 9시 시작, S선수가 미드필더로 출전...

장소	시간	K선수 출전	S선수 포지션	R선수 출전	날씨	결과
어웨이	9시	아니오	미드필더	예	맑음	??

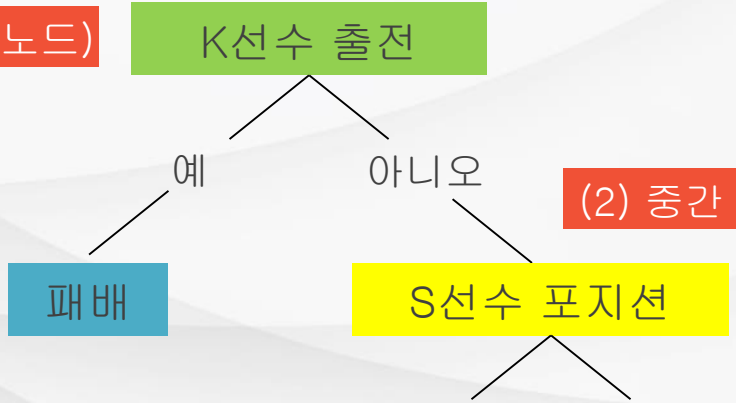
결정 트리

- 결정 트리는 트리 구조의 분류기
  - 결정 노드 : 단일 특성에 대해 데이터를 테스트
  - 말단 노드 : 클래스를 나타냄
  - 엣지 : 하나의 특성 값을 분류
  - 경로(Path) : 최종 분류 결정을 하기 위한 룰(rule)들의 논리합(disjunction)

결정 트리

- 결정 트리는 뿌리(root) 노드부터 시작해서 특성 값에 따라 적절한 말단 노드에 도착함으로써 새로운 데이터의 클래스를 분류

(1) 어느 특성부터 시작하지? (뿌리 노드)

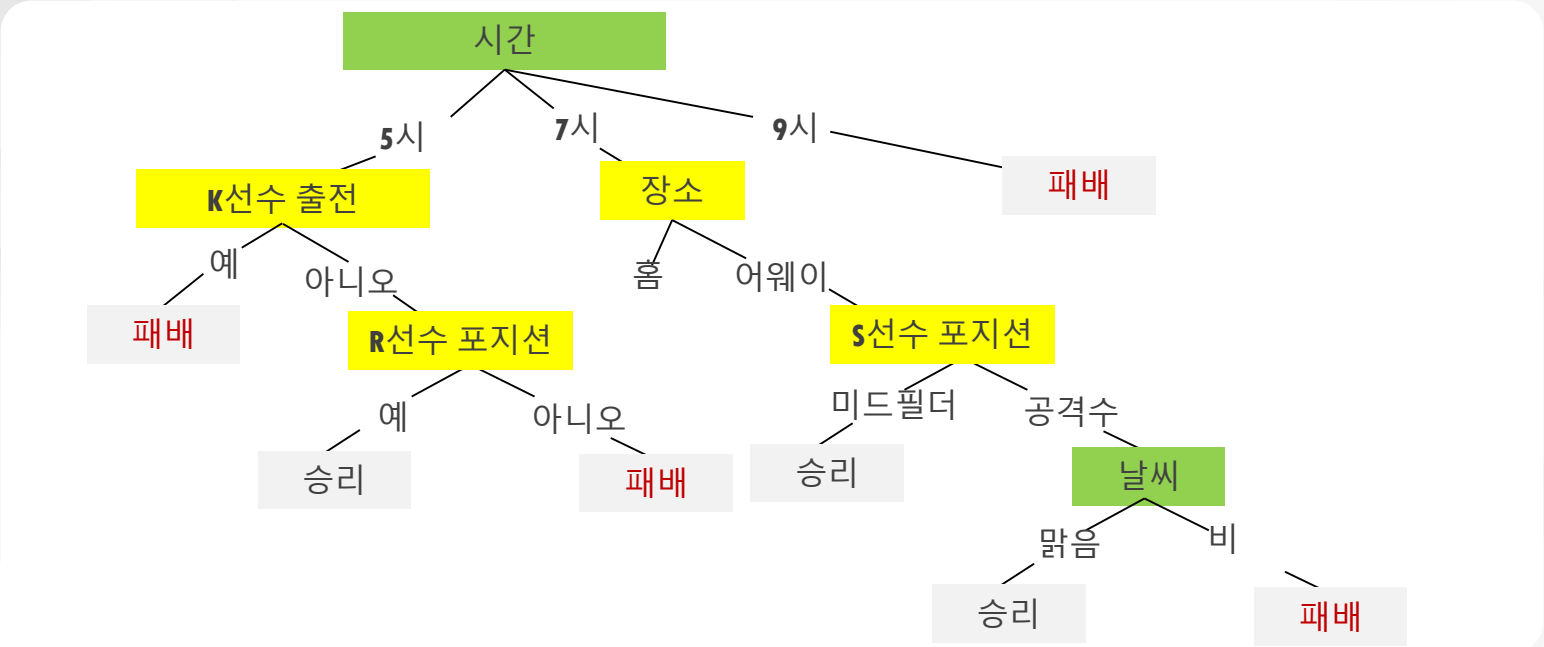


(2) 중간 노드들의 순서는?

(3) 노드를 얼마나 만들어야 할까?

랜덤 결정 트리

- 트리의 크기가 매우 커질 수 있음
- 분류의 룰을 이해하기 어려울 수 있음
- 크기가 큰 트리는 보통 크기가 작은 트리보다 정확도가 떨어짐



2. 결정 트리 구축 원칙

결정 트리 구축 원칙

❖ 각 노드에서 테스트할 특성 선택

- 분류할 때 가장 유용한 특성 순서대로 선택

❖ 정보 획득량

- 각 특성들이 훈련 예제들을 얼만큼 잘 분류할 수 있는가를 측정
- 정보 획득량은 트리 구축 과정에서 테스트할 후보 특성의 순서를 결정할 때 사용

결정 트리 구축 원칙

❖ 엔트로피

- 확률 변수의 불확실성을 수치로 나타냄
- 출력이 두 개(앞/뒤)인 확률 변수 S의 엔트로피 E(S)
  - $E(S) = -p(F)\log_2 p(F) - p(B)\log_2 p(B)$
  - 예1> 동전을 던질 때 앞면과 뒷면이 나올 확률이 같을 때 엔트로피

$$E(S) = -(1/2)\log(1/2) - (1/2)\log(1/2) = 1/2+1/2 = 1$$

  - 예2> 동전을 던질 때 앞면이 나올 확률이 1/4 뒷면이 나올 확률이 3/4 일 때 엔트로피

$$E(S) = -(1/4)\log(1/4) - (3/4)\log(3/4) = 2/4+3/4\times0.42 = 0.31$$

결정 트리 구축 원칙

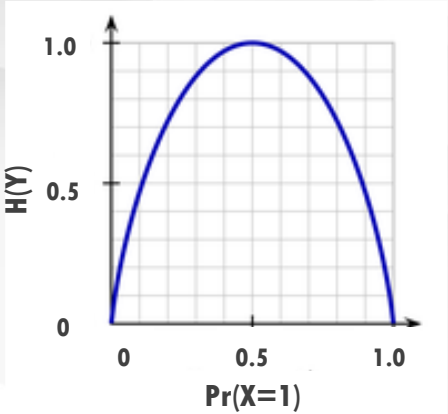
❖ 결정 트리 구축 원칙 예제

- S가 25개 예제를 가지고 있고 그 중 15개가 positive 예제, 10개가 negative 예제인 경우 [15+, 10-], 분류에 대한 S의 엔트로피는 다음과 같음
  - E(S) = - p(P)log2 p(P) – p(N)log2 p(N)
  - E(S) = - (15/25) log2(15/25) - (10/25) log2 (10/25)

엔트로피의 직관적인 이해

- 엔트로피가 0일 때, 출력은 매우 확실한 상태

- 엔트로피는 출력에 대해서 아무런 정보를 갖고 있지 않을 때 (어떠한 출력값도 동등한 확률로 나옴) 최고 값을 가짐
  - 예> 동전 던지기의 앞/뒤 확률이 같은 경우



클래스가 두 개인 문제에서
하나의 클래스의 발생 확률에 대한
엔트로피의 변화

정보 획득량

❖ 정보 획득량 = 엔트로피 (불확실성)의 감소량

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- **Values(A)**는 특성 **A**(예> 날씨)의 모든 가능한 값(예>맑음, 비)들의 집합이고, S_v는 데이터 집합 **S**에서 특성 **A**의 값이 **v**인 **S**의 부분집합 $S_v = \{s \text{ in } S \mid A(s) = v\}$
- 정보 획득량 식의 첫 번째 항은 데이터 집합 **S**의 엔트로피 값

정보 획득량

❖ 정보 획득량 = 엔트로피 (불확실성)의 감소량

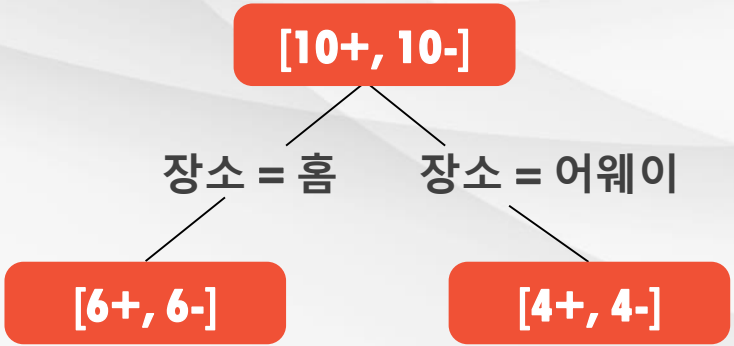
$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- 두 번째 항은 특성 **A**를 사용하여 **S**를 분리했을 때, 예상되는 **S**의 엔트로피의 값
- 정보 획득량 : 특성 **A**를 사용하여 **S**를 분리했을 때, 예상되는 엔트로피의 감소량
- 또는, 특성 **A**를 알 때, 데이터 집합 **S**의 원소를 인코딩 할 때 감소되는 비트의 크기

정보 획득량 예제

- 분리하기 전, S의 엔트로피

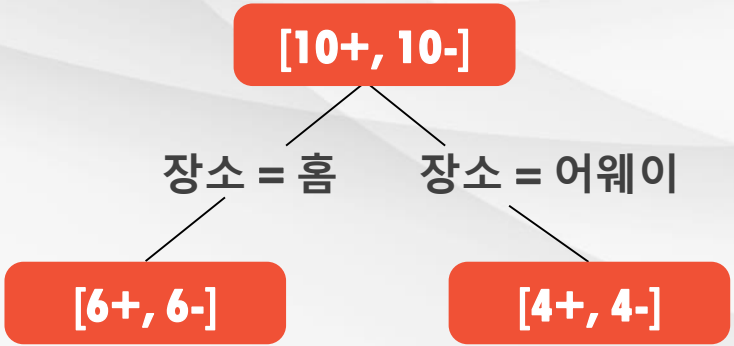
  - H(10/20, 10/20)
    = - 10/20 log(10/20) - 10/20 log(10/20) = 1



정보 획득량 예제

- 장소 특성을 사용하여 두 개의 부분집합으로 분리

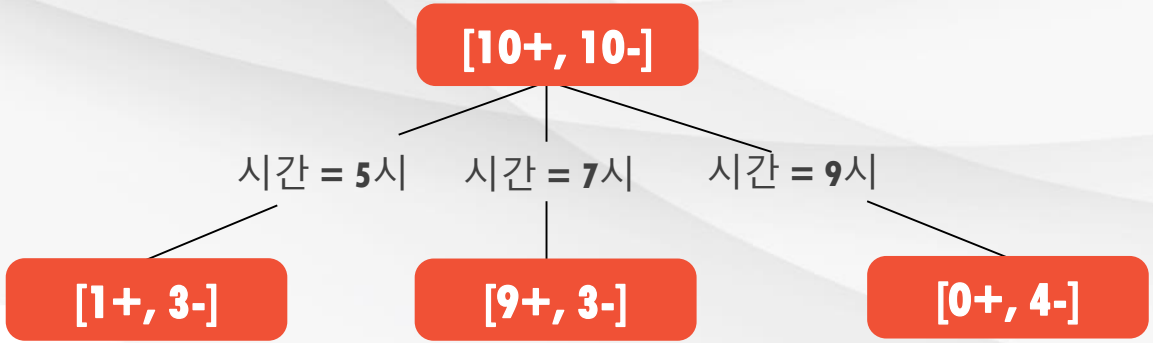
  - 첫 번째 집합의 엔트로피 : H(홈) = - 6/12 log(6/12) - 6/12 log(6/12) = 1
  - 두 번째 집합의 엔트로리 : H(어웨이) = - 4/8 log(6/8) – 4/8 log(4/8) = 1
  - 분리한 뒤 S의 엔트로피 : 12/20 * H(홈) + 8/20 * H(어웨이) = 1



정보 획득량 예제

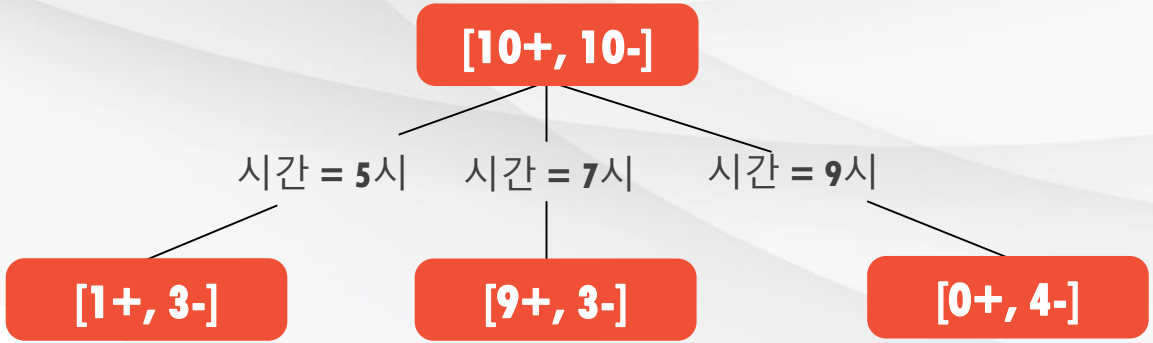
- 시간 특성을 사용하여 세 개의 부분집합으로 분리

  - 첫 번째 집합의 엔트로피 : H(5시) = - 1/4 log(1/4) - 3/4 log(3/4) = 0.31
  - 두 번째 집합의 엔트로피 : H(7시) = - 9/12 log(9/12) – 3/1 log(3/12) = 0.34
  - 세 번째 집합의 엔트로피 : H(9시) = - 0/4 log(0/4) – 4/4 log(4/4) = 0



정보 획득량 예제

- 분리한 뒤 S의 엔트로피
  - 4/20 * H(1/4, 3/4) + 12/20 * H(9/12, 3/12) + 4/20 * H(0/4, 4/4) = 0.65
- 정보획득량
  - 1-0.65 = 0.35



결정 트리 구축 원칙

❖ 어떤 노드가 먼저 선택되어야 할까?

- 시간 특성 값을 알 때가, 장소 특성 값을 알 때보다 더 많은 정보를 줌
- 시간 특성 값을 장소 특성 값보다 먼저 테스트 해야 함
- 이와 같이, 다른 특성들에 대해서 정보 획득량을 계산
- 결정 트리의 각 노드에서 가장 큰 정보 획득량을 갖는 특성을 선택

❖ 얼마나 노드를 만들어야 할까? (stopping rule)

- 모든 특성들이 트리의 경로에 모두 포함되어 있는 경우
- 말단 노드와 연관되어 있는 모든 훈련 예제들이 같은 클래스에 해당하는 경우

즉, 엔트로피가 0일 때

결정 트리 구축 원칙

❖ 가장 적합한 트리는?

- 모델을 구축하기 위해 ‘**bias**’ 가 필요

  예> 크기가 가장 작은 트리를 선호, 또는 깊이가 가장 높은 트리, 노드 개수가 가장 많은 트리

- 어떠한 트리가 새로운 데이터를 가장 잘 분류할 수 있을까?

결정 트리 구축 원칙

❖Occam의 면도날

- 면도날은 필요하지 않은 가설을 잘라내 버린다는 비유
- 데이터의 특성과 부합하는 가장 간단한 가설을 선호
- 모든 것이 똑같은 조건일 때, 가장 쉬운 설명이 가장 옳은 것
- 과학자의 가설 선호도
- 많은 데이터를 설명할 수 있으면서 가장 간단한 가설을 선택

예> F=ma

"결정 트리는 데이터의 특성과 부합하면서도 가능한 가장 작은 트리여야 함"

결정 트리 분류의 특징

❖ 장점

- 다른 분류 방법에 비해서 상대적으로 빠름 (분류 시 계산량이 적음)
- 간단하고 모델 구축 원리를 이해하기 쉬움
- 모델의 분류 룰(rule)을 사람이 직관적으로 이해하기 쉬움
  - 어느 특성이 분류에 가장 중요한지 명확히 나타냄
- 다른 모델들에 비해서 더 좋은 성능 나타낼 때가 있음

결정 트리 분류의 특징

❖ 단점

- 연속적인 특성 값을 갖는 데이터에 적합하지 않음
- 클래스의 개수가 많고 데이터가 적을 때 성능이 좋지 않음
- 훈련과정에서 계산량이 많음
  - 트리의 노드 선택 순서를 정하기 위해
    모든 특성의 정보 획득량을 계산한 뒤, 정렬해야 함

특성이 연속된 값을 가질 때 결정 트리 구축

❖ 연속된 값을 갖는 특성

- 특성 **A**의 연속된 값을 이산적인 구간의 집합으로 나눔
- 새로운 **boolean** 특성 **Ac**의 값을 임계치 **c**에 따라 설정

$$A_c = \begin{cases} true & \text{if } A_c < c \\ false & \text{otherwise} \end{cases}$$

예> 새로운 특성: 섭씨 온도

❖ 어떻게 임계치 c를 정할까?

- 데이터를 특성의 값에 따라서 정렬시킴
- 클래스가 다르면서 인접한 두 개의 데이터를 참음
  - 임계치의 집합이 생성됨
- 정보 획득량이 가장 많은 임계치를 선택

3. Bagging

Bootstrap

- 데이터의 개수가 부족할 때 사용할 수 있음
  - 예> 날씨가 맑은 날 A팀의 평균 승리 확률은?

- 원본 데이터 집합 X가 있을 때, $X=(x_1,x_2,\ldots,x_n)$, 다음 과정을 m번 반복
  - 크기가 n인 데이터 집합 X에서 복원 추출법을 사용하여 부분 집합 Xk 생성
  - X_k에 대해서 구하고자 하는 값 $\hat{\theta}*$계산

bootstrap 값 계산

$$\hat{\theta}* = (\hat{\theta}_1^*,\ldots,\hat{\theta}_B^*)$$

Bootstrap

- **Bootstrap** 값들을 사용하여 표준 편차 또는 확신 구간 등 설정 가능
- **1980**년대부터 컴퓨터가 통계적 용도로 사용되면서 쓰여지기 시작
- **Bootstrap** 분포는 원본 데이터에서 통계적 변화를 측정하기 위해 주로 사용됨

Bagging

- 1996년 Breiman에 의해 소개됨
- Bagging : "bootstrap aggregating"
- 다수의 분류기를 결합하는 앙상블 방법 중 하나
  - 주로 결정 트리 분류기에 많이 쓰임
- 원본 훈련 데이터 집합 X가 있을 때, 다음 과정을 m번 반복
  - X로부터 boostrap 샘플 X_k 를 얻음
  - X_k 로부터 분류기 C_k 를 훈련

Bagging

- **m**개의 분류기를 다음의 방식으로 결합
  - 투표
  - 평균
- 데이터가 갖는 **high variance** 문제를 해결할 수 있음

학습정리

학습정리

Tacademy

지금까지 [결정 트리]에 대해서 살펴보았습니다.

의사 결정 트리와 분류 문제

결정 트리는 트리 구조의 분류기
노드 : 단일 특성에 대해 데이터를 테스트
말단 노드 : 클래스를 나타냄
엣지: 하나의 특성 값을 분류
경로: 최종 분류 결정을 하기 위한 룰들의 논리합

결정 트리 구축 원칙

정보 획득량은 결정 트리 구축 과정에서 테스트할 후보 특성의 순서를 결정할 때 사용

Bagging

Bagging은 다수의 분류기를 결합하는 앙상블 방법 중 하나
k개의 bootstrap 샘플 $X_{1 \ldots k}$를 훈련시켜 얻은 k개의 분류기를 투표, 평균 등의 방법을 사용하여 결합

인공지능을 위한 머신러닝 알고리즘

5. 서포트 벡터 머신

Tacademy

CONTENTS

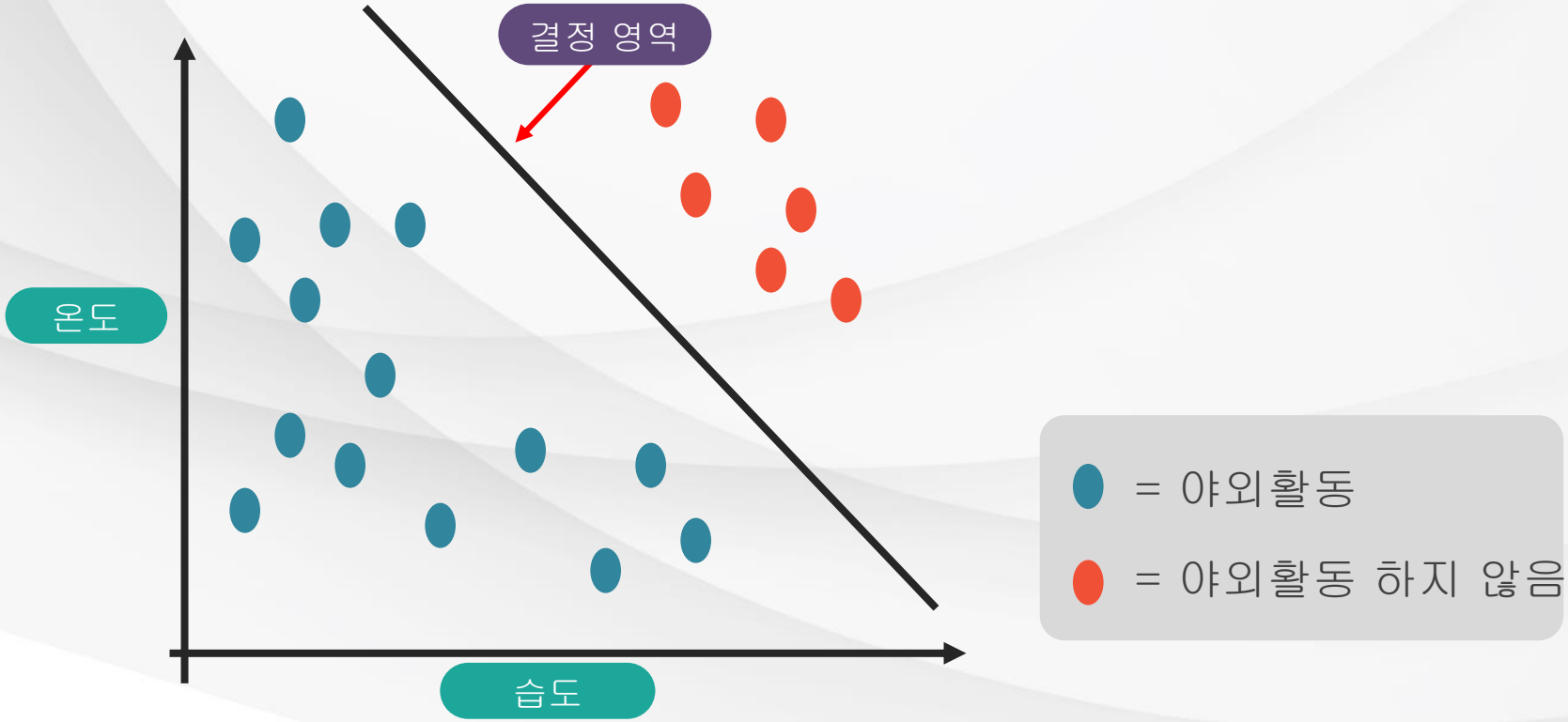
학습 목표

- 서포트 벡터 머신의 분류 원리에 대해서 이해할 수 있다.
- 서포트 벡터 머신의 학습이 최적화 문제로 변형되어 해결되는 과정을 이해할 수 있다.
- 소프트 마진 분류기와 커널을 사용한 비선형 분류기를 이해할 수 있다.

1. 좋은 선형 분류기 만들기

선형 분류의 예

❖ 예> 야외활동하기 좋은 날 분류하기

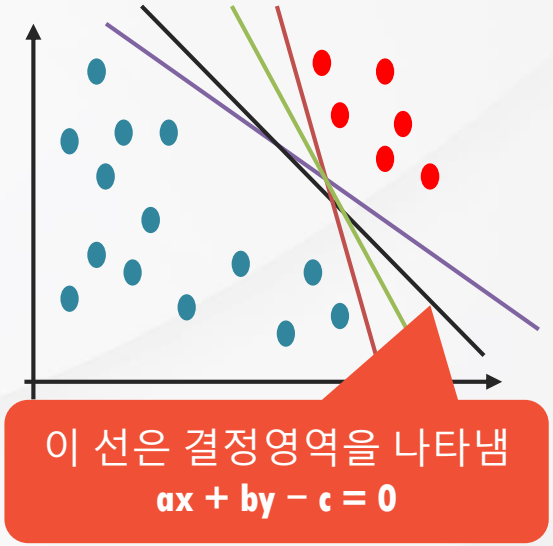


어떤 초평면을 선택해야 할까?

- 초평면이란?
  데이터 임베딩 공간에서 한 차원 낮은 부분 공간 (subspace)

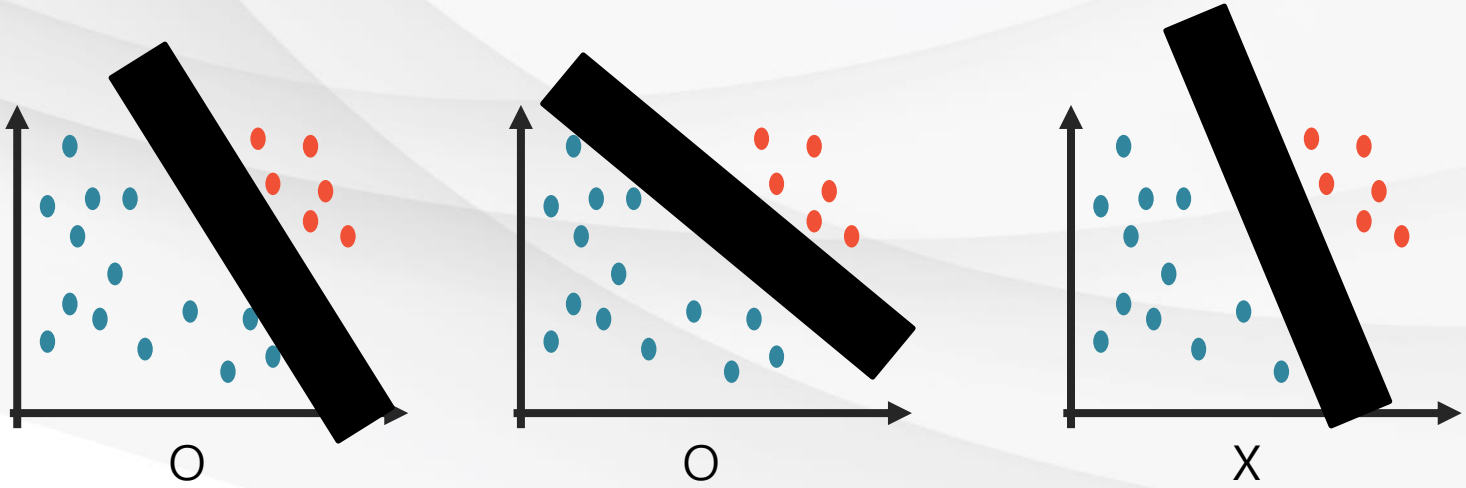
  예> 3차원 공간의 초평면: 2차원 평면

- 가능한 a, b, c에 대해 많은 해답이 존재
- 여러 선형 분류기들이 있지만
  항상 최적의 초평면을 찾지는 않음 예> 퍼셉트론
- 최적의 초평면의 조건
  - 초평면과 결정영역 근처에 있는 '분류하기 애매한' 데이터의 거리가 최대가 되어야 함
  - 직관적인 이해 : 만약 결정 영역 근처에 데이터가 없다면,
    분류기는 분류 결정을 하기 위해 불확실하고
    애매한 결정을 내리는 경우가 조금 더 드물어질 것임



또 다른 직관

- 두꺼운 결정영역을 클래스 사이에 놓는다면, 선택의 가짓수는 보다 줄어듦
- 모델의 파라미터 개수를 줄일 수 있음



2. 서포트 벡터 머신

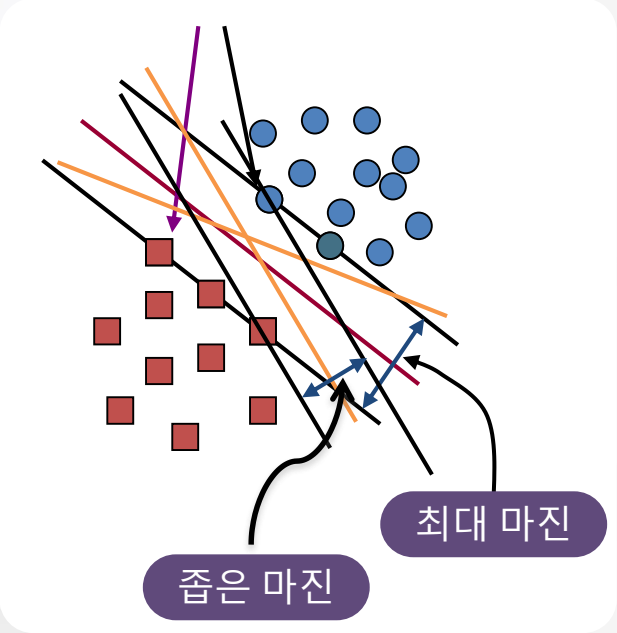
서포트 벡터 머신의 분류 원리

- 서포트 벡터 머신은 결정 영역의 초평면을 둘러싸고 있는 마진(margin)을 최대화시킴

예> 3차원 공간의 초평면 : 2차원 평면

- 클래스 결정 함수는 훈련 예제의 부분 집합(서포트 벡터)만으로 완전히 설명 가능
- 서포트 벡터 머신의 결정 함수를 구하는 것은 함수 최적화 문제
- 딥러닝이 나오기 전까지 가장 성공적인 분류기 역할을 했음

서포트 벡터 (support vectors)



서포트 벡터 머신 형식

w	정규화된 결정 초평면 벡터
x_i	데이터 포인트 i
y_i	데이터 포인트 i의 클래스 (+1 또는 -1)

서포트 벡터 머신 형식

◉ 분류기 형식

$$y_i = f(x_i) = \text{sign}(w^T x_i + b)$$

◉ x_i의 기능적 마진 (functional margin)

$$y_i\ (w^T x_i + b)$$

- 주어진 데이터 포인트가 적절하게 분류되었는지 아닌지 가늠할 수 있는 테스트 함수의 역할
- 값이 클수록 적절하게 분류되었는지 확인할 수 있음
- w와 b의 크기를 키움으로써 마진의 크기도 증가시킬 수 있음

sign함수란?

- 수의 부호를 판별하는 함수

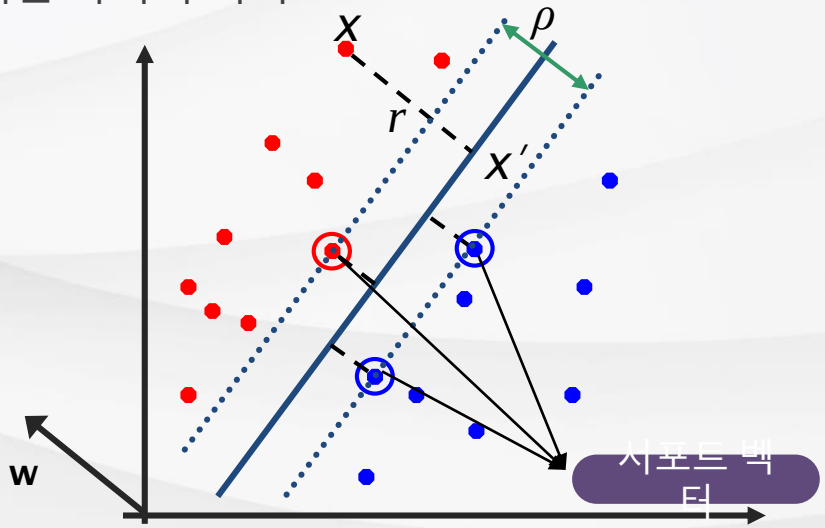
$$y_i = +1 \text{ when } w^T x_i + b \geq +1$$
$$y_i = -1 \text{ when } w^T x_i + b \leq -1$$

기하 마진 (geometric margin)

- 데이터 포인트에서 결정 영역까지의 거리 : $$r = y\frac{\boldsymbol{w}^T\boldsymbol{x} + b}{\|w\|}$$
- 결정 영역까지 가장 가까운 데이터들을 서포트 벡터(support vectors)라고 함
- 결정 영역의 마진 ρ 는 클래스들의 서포트 벡터들 사이의 거리

r 을 계산하기 위한 과정

점선 **x'−x** 은 결정 영역에 직교하고 **w**에 평행함
유닛 벡터는 **w/|w|** 이고, 점선은 **rw/|w|** 임
x' = x − yrw/|w|
x'는 **w^Tx'+b = 0**을 만족함
그러므로 **w^T(x −yrw/|w|) + b = 0**
|w| = sqrt(w^Tw)이고, **w^Tx −yr|w| + b = 0**
r에 대해서 정리하면, **r = y(w^Tx + b)/|w|**



선형 서포트 벡터 머신

- 모든 데이터의 기능적 마진이 항상 1 이상이라고 가정한다면, 다음 두 개의 조건이 훈련 데이터 집합 $\{(x_i, y_i)\}$에 대해 만족함

$$w^T x_i + b \geq 1 \quad \text{if } y_i = 1$$

$$w^T x_i + b \leq -1 \quad \text{if } y_i = -1$$

- 서포트 벡터는 부등호가 등호로 바뀜
- 각 데이터 포인트로부터 결정영역까지의 거리
  - 기능적 마진을 파라미터 **w**에 따라 크기를 바꿔줌
  - 데이터 포인트가 적절하게 분류가 잘 되었는지, **|w|**로 스케일링 된 척도로 알려줌

$$r = y \frac{w^T x_i + b}{\|w\|}$$

- 마진 ρ : $\frac{2}{\|w\|}$

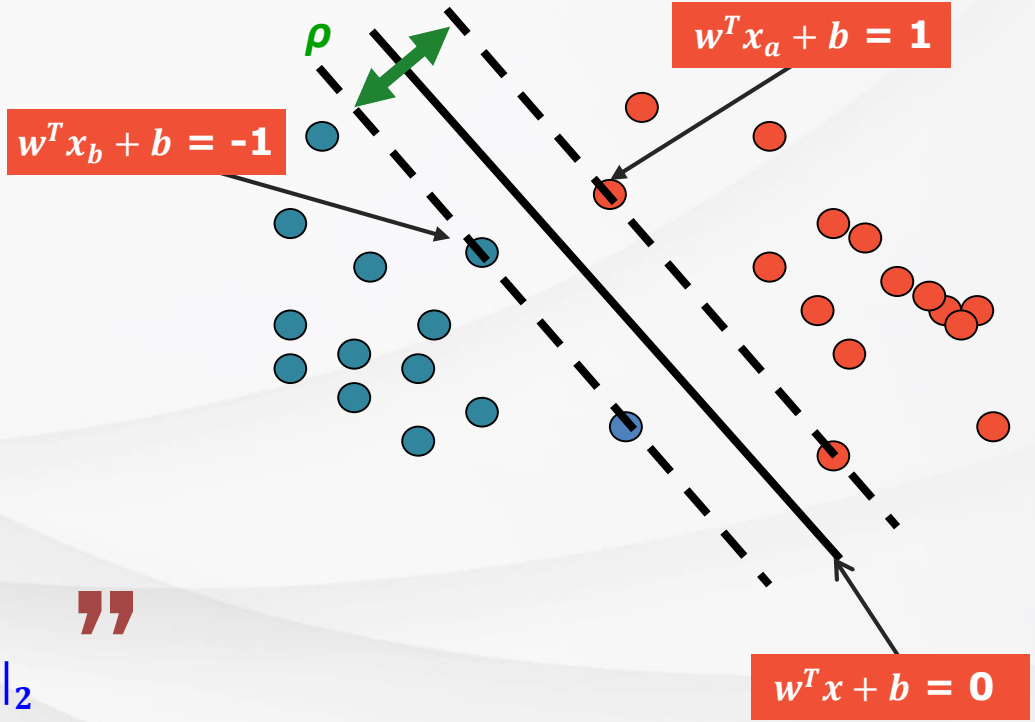
선형 서포트 벡터 머신 정리

- 초평면

$$w^T x_i + b = 0$$

- 추가 조건

$$min_{i=1\ldots n} \left| w^T x_i + b \right| = 1$$



“ 즉, $w^T(x_a - x_b) = 2$

$\rho = \|x_a - x_b\|_2 = 2/\|w\|_2$ ”

최적화 문제를 사용한 파라미터 계산 (1/3)

- 서포트 벡터 머신의 파라미터를 찾기 위해서 최적화 문제로 변형시킬 수 있음

Find w and *b* such that

1 / ||w|| is maximized; and for all $\{(x_i, y_i)\}$

$w^T x_i + b \geq 1$ if $y_i = 1$; $w^T x_i + b \leq -1$ if $y_i = -1$

- 보다 나은 형식으로 변형 (min ||w|| = max 1/ ||w||)

Find w and *b* such that

$\Phi(w) = \frac{1}{2} w^T w$ is minimized;

and for all $\{(x_i, y_i)\}$: $y_i (w^T x_i + b) \geq 1$

최적화 문제를 사용한 파라미터 계산 (2/3)

Find w and b such that
$\Phi(w) = \frac{1}{2} w^T w$ is minimized ;
and for all $\{(x_i, y_i)\} : y_i (w^T x_i + b) \geq 1$

- 선형 조건에 부합하도록 이차함수를 최적화 시키는 문제
- 이차함수의 최적화 문제는 수학적 프로그래밍 문제에서 잘 알려진 분야로, 해결할 수 있는 많은 알고리즘이 존재함
- Lagrange multiplier α_i 을 사용하여 다음의 primal과 dual problem으로 변형 가능

Maximize
$L(w,b) = 1/2 w^T w - \Sigma\alpha_i\{y_i(w^T x_i - b) - 1\}$
(1) $\alpha_i \geq 0$ for all α_i

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i^T x_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

최적화 문제를 사용한 파라미터 계산 (3/3)

솔루션은 다음과 같은 형식을 가짐

$$\mathbf{W} = \sum \alpha_i y_i X_i \quad b = y_k - w^T X_k \text{ , } k\text{는 } \alpha_k \neq 0 \text{ 을 만족}$$

- 0이 아닌 α_i 는 해당하는 x_i가 서포트 벡터임을 의미

그러므로 분류함수는 다음과 같은 형식임

$$f(\mathbf{X}) = \sum \alpha_i y_i X_i^{\mathrm{T}} \mathbf{X} + b$$

- 분류는 새로운 테스트 데이터 **x**와 서포트 벡터 x_i의 내적에 의해 계산됨

“하지만, 모델의 훈련 과정 때는
모든 훈련 데이터 쌍 (x_i, x_j)에 대해 내적 $x_i^T x_j$을 계산”

3. 비선형 분류기 만들기

소프트 마진 분류 (soft margin classification)

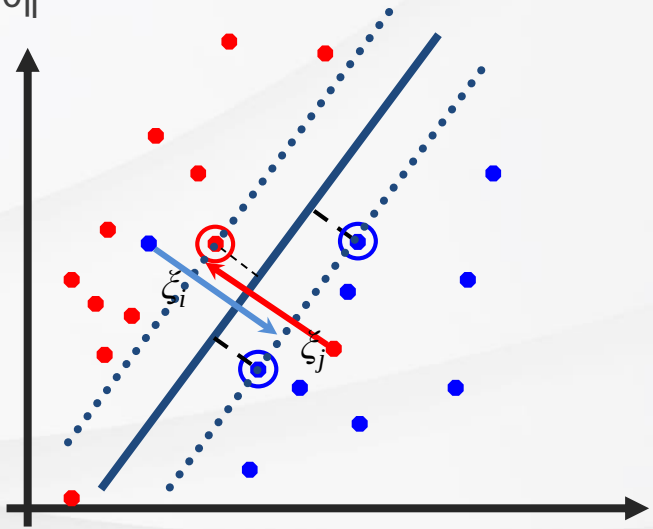
- 만약 훈련 데이터가 선형으로 분리되지 않을 경우, 슬랙 변수 ξ_i 가 잘못 분류되거나 노이즈가 포함된 데이터에 추가됨
- 잘못 분류된 데이터 포인트를 본래 속하는 클래스로 비용을 들여 이동시켜줌

$$y_i(w^T x_i + b) \geq 1 \qquad min\|w\|$$

$$\Rightarrow$$

$$y_i(w^T x_i + b) \geq 1 - \xi_i \qquad min\|w\| + C\|\xi\|$$

- 모델의 학습 방법은 여전히 결정 영역을 각 클래스로부터 가장 멀리 위치하는 것임 (large margin)



소프트 마진 분류의 파라미터 계산

- 예전 문제 형식

Find w and *b* such that
$\Phi(w) = ½\, w^T w$ is minimized and for all $\{(x_i, y_i)\}$
$y_i(w^T x_i + b) \geq 1$

- 새로운 형식은 슬랙 변수를 포함하고 있음

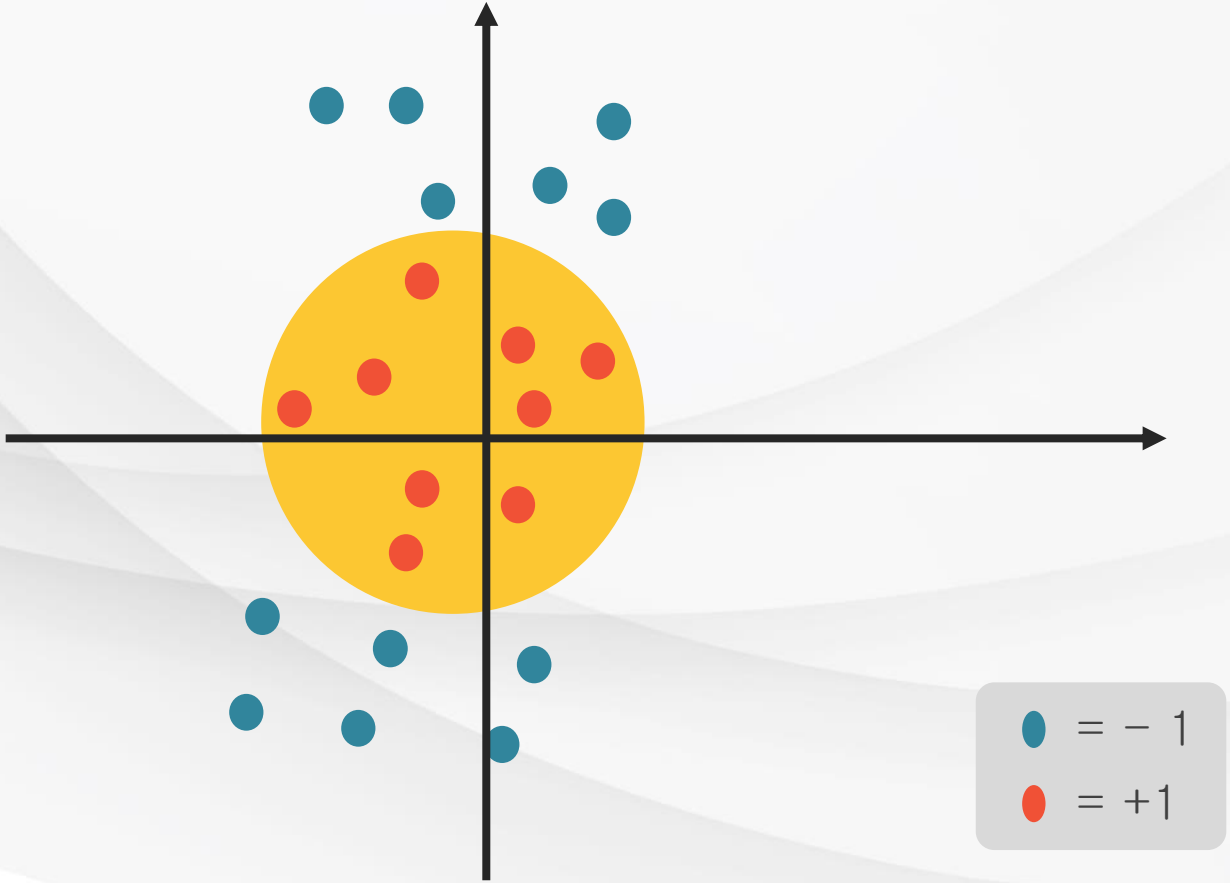
Find w and b such that
$\Phi(w) = ½\, w^T w + C\Sigma\xi_i$ is minimized and for all $\{(x_i, y_i)\}$
$y_i(w^T x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- 솔루션은 다음과 같은 형식을 가짐 (일반 서포트 벡터 머신과 유사)

$$W = \sum \alpha_i y_i X_i$$
$$b = y_k(1 - \xi_k) - w^T X_k \text{ where } k = \text{argmax } \alpha_k,$$

3. 비선형 분류기 만들기

- 결정 영역이 선형이 아닌 경우

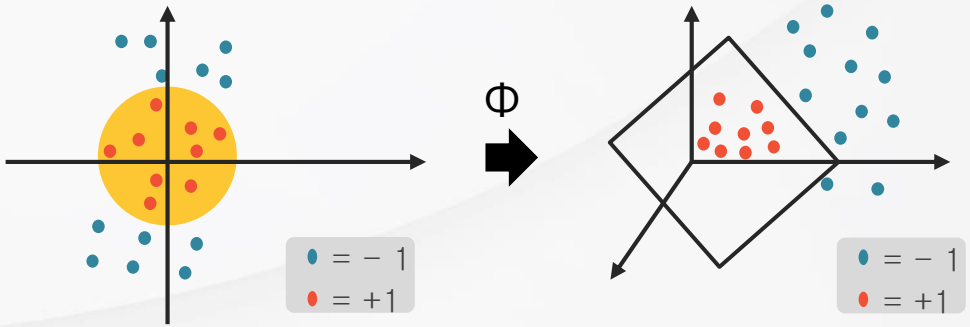


결정 영역이 선형이 아닌 경우

- 데이터 포인트를 선형으로 분류하기 위해 차원을 더 생성

$$(x_1, x_2) \Rightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

  - Φ : x → Φ(x)
  - f(x) = w^TΦ(x) + b를 학습
  - Φ(x)는 피처 맵이라고 부름



- 파라미터 결정

x대신 Φ(x)을 사용하여 다음 식을 계산

$$\text{maximize} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \langle \phi(\mathrm{x}_i) \cdot \phi(\mathrm{x}_j) \rangle$$

  - 커널 $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \langle \phi(\boldsymbol{x_i}) \cdot \phi(\boldsymbol{x_j}) \rangle$ 이며 이 경우, $\Phi(\boldsymbol{x_i})^T\Phi(\boldsymbol{x_j}) = (\boldsymbol{x_i}^T\boldsymbol{x_j})^2$ 임

분류기로써 서포트 벡터 머신의 성능

- 서포트 벡터 머신은 실세계 데이터에 좋은 성능을 보여줌
  - 프로그래머는 커널 함수를 설계 해야 함
  - 나머지 파라미터는 자동으로 계산
- 데이터 집합의 크기가 클 수록 시간 소모가 큼
  - 초평면의 최대 마진을 구하기 위해서
    훈련 데이터 개수의 제곱에 해당하는 계산량 필요
  - 모든 서포트 벡터를 저장 해야 함

"만약, 문제에 어떠한 알고리즘을 사용할지 모르겠다면,"
서포트 벡터 머신은 좋은 출발선이 될 수 있음

학습정리

학습정리

Tacademy

지금까지 [서포트 벡터 머신]에 대해서 살펴보았습니다.

좋은 선형 분류기 만들기

서포트 벡터 x_i가 결정 영역의 초평면 w를 결정

$$y = f(\mathbf{x}) = \text{sign}(\boldsymbol{w}^T\boldsymbol{x} + b) = \text{sign}(\sum \alpha_i y_i X_i^T \boldsymbol{X}) + b$$

서포트 벡터 머신

이차함수의 최적화 문제를 통해 서포트 벡터와,
Lagrangian multiplier α_i를 계산할 수 있음

비선형 분류기 만들기

슬랙 변수 ξ_i 를 잘못 분류되거나 노이즈가 포함된 데이터에 추가하여
본래 속하는 클래스로 비용을 들여 이동시켜줌

$$y_i(\boldsymbol{w}^T\boldsymbol{x} + b) \geq 1 - \xi_i$$

인공지능을 위한 머신러닝 알고리즘

6. 신경망

Tacademy

CONTENTS

학습 목표

- 컴퓨터와 뇌의 동작 방식을 비교할 수 있다.
- 퍼셉트론의 구성 요소를 이해할 수 있다.
- 다층 신경망의 분류 원리를 이해할 수 있다.

1. 뇌와 컴퓨터

뇌와 컴퓨터의 비교

1. 100억 개의 뉴런
2. 60조 개의 시냅스
3. 분산 처리 방식
4. 비선형 연산
5. 병렬 처리

1. 뉴런보다 빠른 연산(10^{-9}초)

   뉴런: 10^{-3}초
2. 중앙 처리 방식
3. 산술 연산 (선형)
4. 순차 처리

2. 퍼셉트론: 뇌를 모사한 신경망 알고리즘

신경회로망에 대한 연구

1940년대 ~ 1960년대

- 뇌의 구조에 대한 모방이 두드러진 시기
- **McCulloch and Pitts,** 뉴런 모델화**(1943)**
- **Weiner,** 사이버네틱스**(1948)**
- **Rosenblatt -** 퍼셉트론**(1957)**

생물학적 뉴런의 구조 vs. 인공 뉴런

- 생물학적인 뉴런과의 유사성

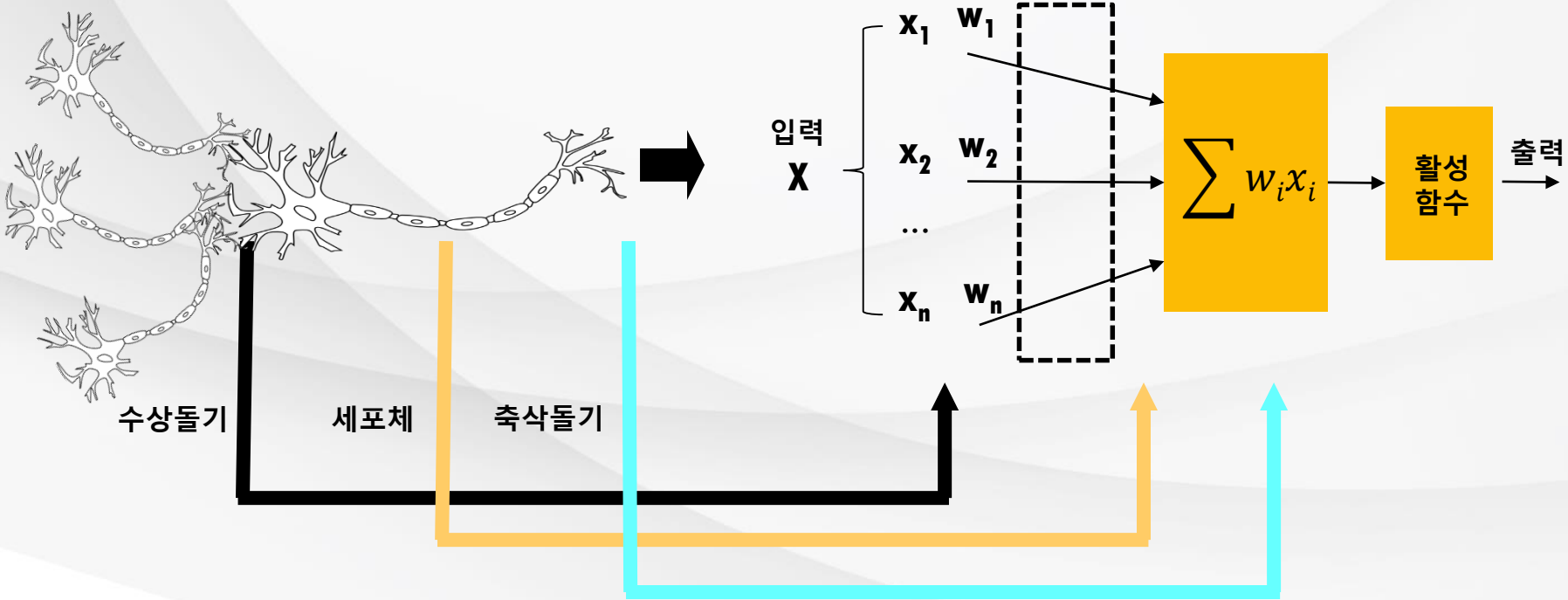
병렬 계산(parallel computing)

분산 표현(distributed representation)

2. 퍼셉트론: 뇌를 모사한 신경망 알고리즘

Tacademy

생물학적 뉴런에서 인공 뉴런으로

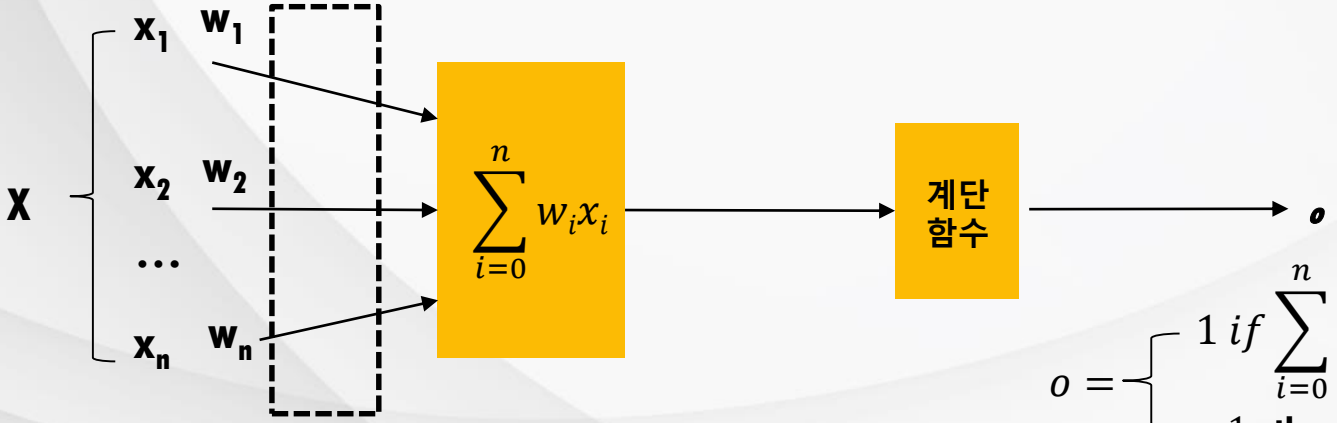


2. 퍼셉트론: 뇌를 모사한 신경망 알고리즘

신경망(인공 뉴런) 학습에 적합한 문제

- 학습해야 하는 현상이 여러 가지 속성에 의해 표현되는 경우
- 학습 예제에 에러(noise)가 존재할 가능성
- 긴 학습 시간
- 학습된 결과를 사람이 이해하는 것이 필요 없는 경우

퍼셉트론 (1957)



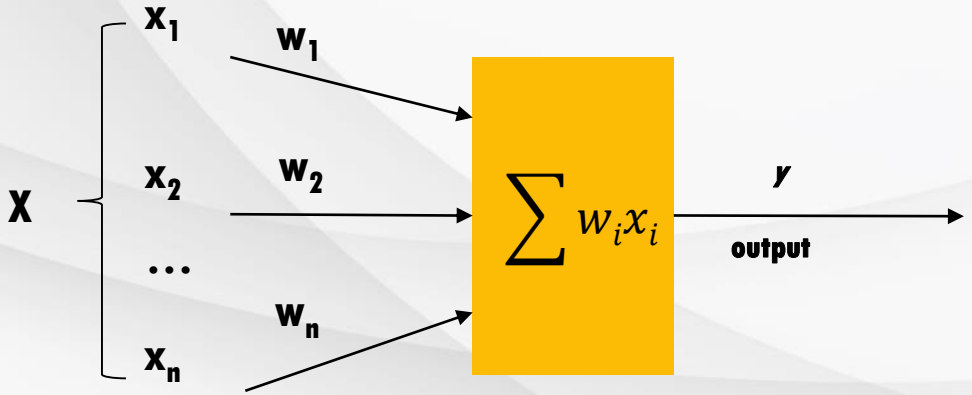
$$o = \begin{cases} 1 \; if \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \; \textbf{otherwise} \end{cases}$$

$$o(x1, \ldots, xn) = \begin{cases} 1 \; if \; w_0 + w_1 x_1 + \; \ldots + wnxn > 0 \\ -1 \; \textbf{otherwise} \end{cases}$$

- 입력: 실수 값을 갖는 벡터
- 모델 구성: 연결 가중치 & 임계치
- 출력: 1 또는 -1
- 학습: 알맞은 연결 가중치를 탐색

퍼셉트론에서 비선형 함수

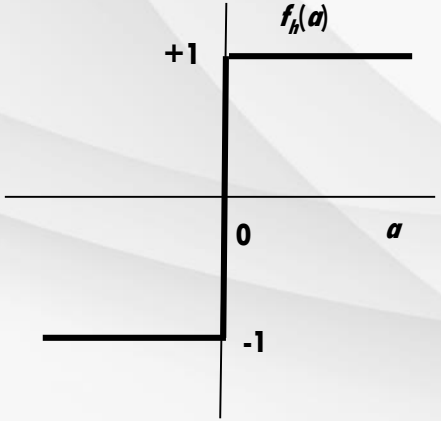
기본적인 노드에서의 입출력



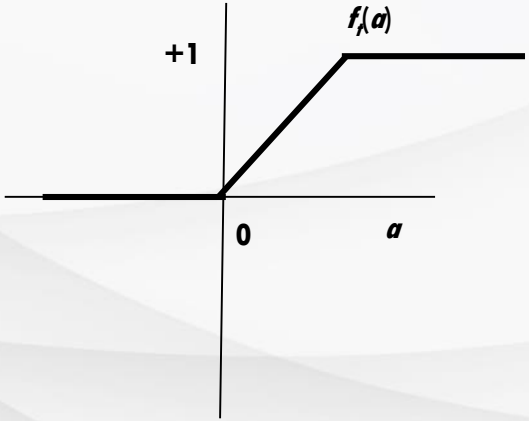
$$y = f\left(\sum_{i=0}^{n} w_i x_i\right)$$

퍼셉트론에서 비선형 함수

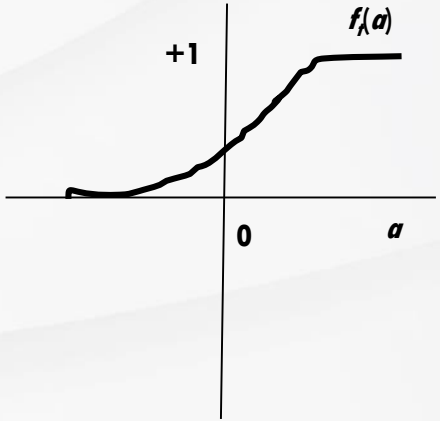
3가지 대표적인 비선형함수



계단 함수



임계논리 함수



시그모이드 함수

퍼셉트론 학습 룰

$$w_i \longleftarrow w_i + \Delta w_i$$

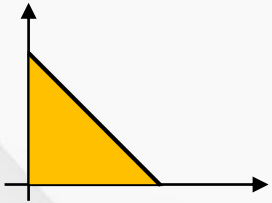
where $\Delta w_i = \eta(t - o)x_i$

- **t**는 타겟 값
- **o**는 퍼셉트론의 출력
- η는 학습률 (작은 상수)

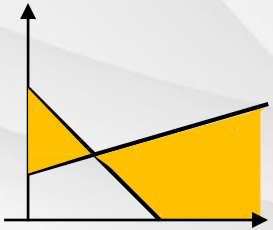
학습 후 올바른 가중치를 찾아내려면 충족되어야 할 사항

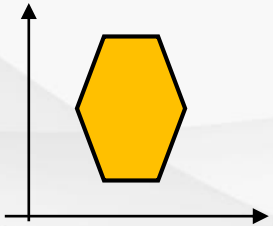
- 훈련 데이터가 선형 분리 문제이어야 함
- 충분히 작은 학습률(**learning rate**)

퍼셉트론의 한계



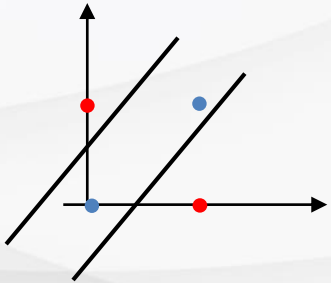
선형 분리 가능 지역





선형 분리 불가능 지역

선형 분리 가능의 일반적인 개념



입력		XOR
0	0	0
0	1	1
1	0	1
1	1	0

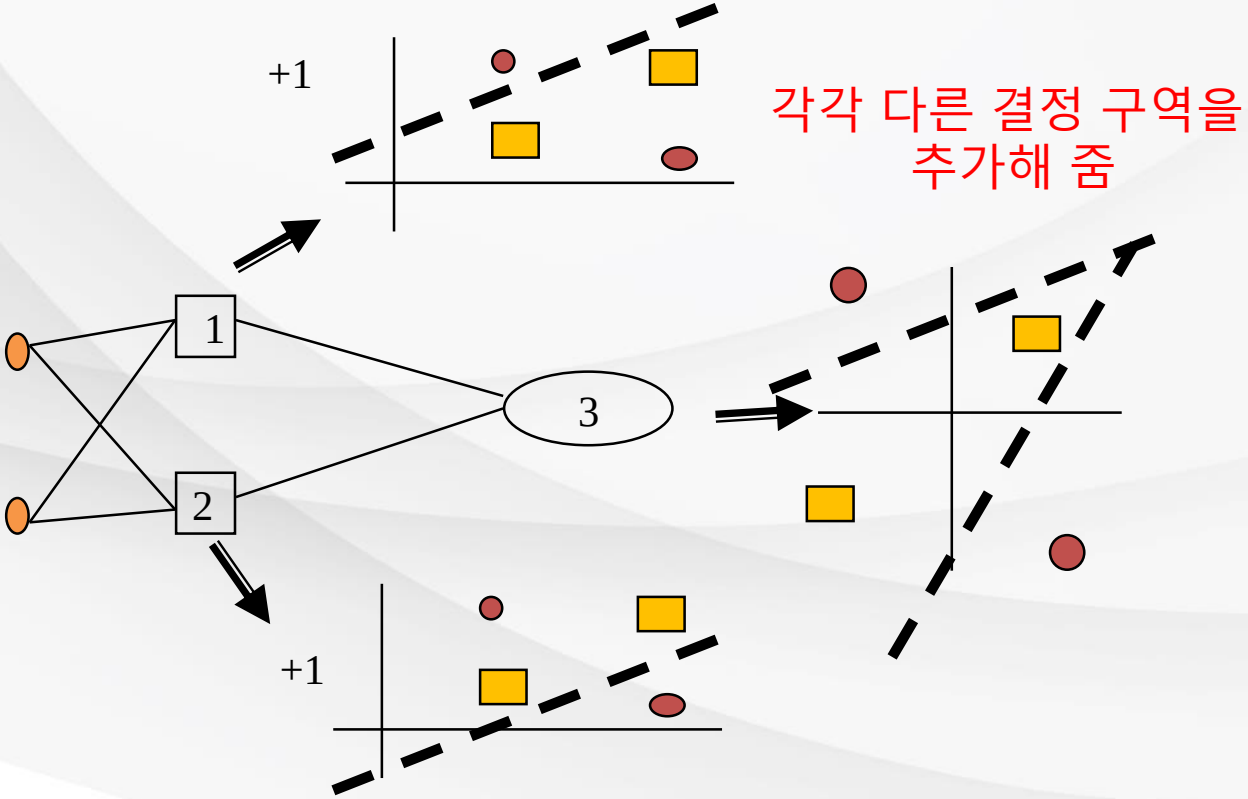
XOR 함수

문제점: XOR 문제에서 선형 분리 불가능

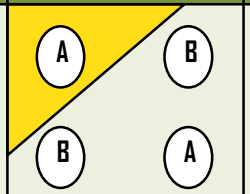
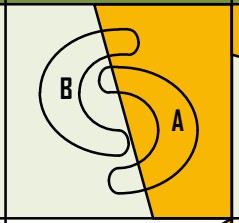
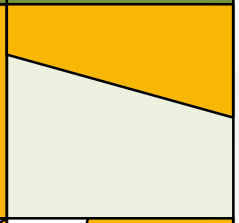
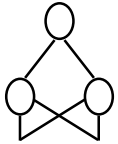
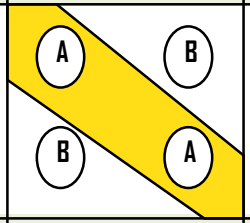
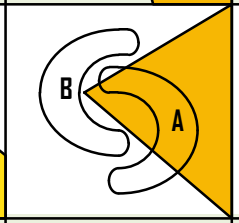
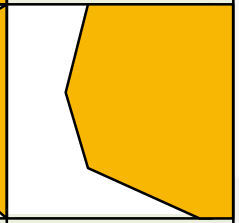
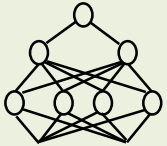
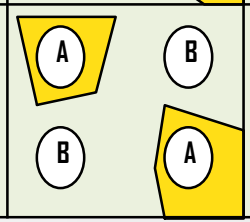
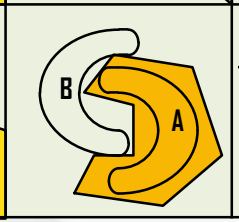
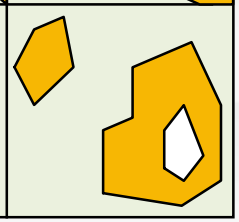
3. 다층 신경망의 분류 원리

XOR 문제의 해결

모델에 은닉층을 추가하면 분류를 위한 더 좋은 결정 영역을 얻을 수 있음

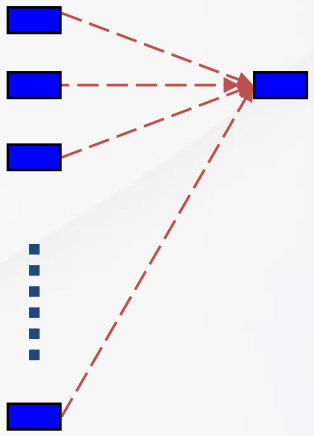


여러 패턴들의 결정 형태

구조	결정 영역 형태	Exclusive-OR 문제	얽힌 결정 영역을 갖는 클래스들	대부분의 결정 영역들
단층 	초평면에 의해 나뉘어지는 반평면 (Hyper plane)			
두 개층 	볼록한 모양 또는 닫힌 영역			
세 개층 	임의의 형태 (노드들의 개수에 따라 복잡도가 결정됨)			

다층 퍼셉트론 (다층 신경망)의 특징

- 같은 층 안에서는 연결이 존재하지 않음
- 입력층과 출력층 사이 직접적인 연결이 존재하지 않음
- 각 층 사이는 완전 연결(**Fully Connected**) 되어 있음
- 입력층 - 은닉층 – 출력층 구조를 많이 가짐
- 출력층의 유닛 개수는 입력층의 유닛 개수와 같을 필요가 없음
- 은닉층의 유닛 개수는 입력층 또는 출력층 보다 많거나 적어도 됨



*각 유닛은 퍼셉트론

$$y_i = f\left(\sum_{j=1}^{m} w_{ij} x_j + b_i\right)$$

Bias를 추가적인 가중치로 갖기도 함

다중 퍼셉트론 가중치 학습

- 오류 역전파 알고리즘(**Error Back Propagation**) 이용함
- **Generalized Delta Rule** 이용

$$o_{ip} = \sum_i w_{ij} \times i_{ip}$$

$$E_p = \frac{1}{2}\sum_j (t_{jp} - o_{jp})^2$$

$$E = \sum_p E_p$$

$$\frac{\delta E_p}{\delta w_{ij}} = \frac{\delta E_p}{\delta o_{jp}} \times \frac{\delta o_{jp}}{\delta w_{ij}}$$

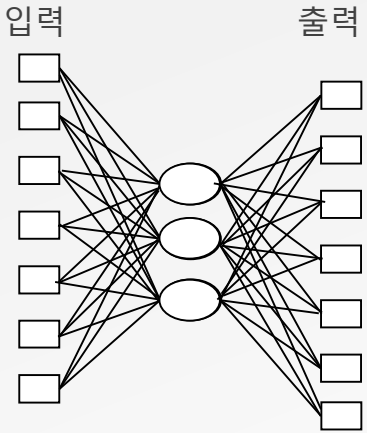
$$\frac{\delta E_p}{\delta w_{ij}} = -2\frac{(t_{jp} - o_{jp})}{2} \times i_{ip} = -(t_{jp} - o_{jp}) \times i_{ip}$$

$$w_{ij} = -\alpha\frac{\delta E_p}{\delta w_{ij}} = -\alpha(t_{pj} - o_{pj}) \times i_{ip}$$

은닉 유닛의 표현

- 입력값 들의 특성을 스스로 파악해서 **Hidden Layer**에 표현하는 능력이 있음
- 사람이 미리 정해 준 **Feature**만을 사용하는 경우보다 유연하며 미리 알 수 없는 특성을 파악하는데 유용

입력 - 학습된 은닉 유닛 - 출력 유닛의 예

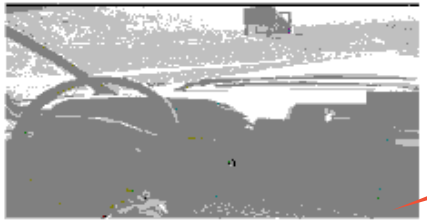


네트워크

입력		은닉값				출력
10000000	→	.89	.04	.02	→	10000000
01000000	→	.48	.38	.25	→	01000000
00100000	→	.69	.68	.82	→	00100000
00010000	→	.84	.14	.57	→	00010000
00001000	→	.70	.79	.12	→	00001000
00000100	→	.25	.45	.09	→	00000100
00000010	→	.07	.58	.96	→	00000010
00000001	→	.84	.83	.74	→	00000001

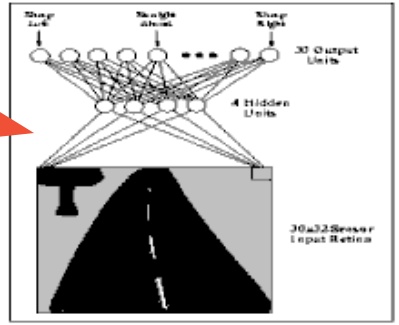
응용 예제: Autonomous Land Vehicle (ALV)

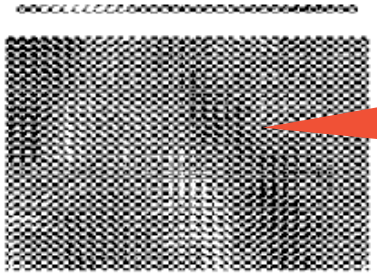
- 신경망이 무인 자동차의 운전대를 움직이는 방법을 학습
- 960개의 입력 유닛, 4개의 은닉 유닛, 30개의 출력 유닛
- 70 miles/h의 속도로 주행



CMU의
ALVINN 시스템

차량 앞에
장착된
카메라로부터
받은 이미지





한 개 은닉 유닛의
가중치 값

학습정리

학습정리

Tacademy

지금까지 [신경망]에 대해서 살펴보았습니다.

뇌와 컴퓨터

분산 처리(뇌) <-> 중앙 처리(컴퓨터)
병렬 처리(뇌) <-> 순차 처리(컴퓨터)
처리속도: 10^{-3}초 (뇌) <-> 10^{-9}초(컴퓨터)

퍼셉트론: 뇌를 모사한 신경망 알고리즘

퍼셉트론은 입력 유닛들의 가중치의 합과 임계치를 사용하여 뉴런의 발화 과정을 모사
선형 분리 문제를 해결하지 못하는 문제 (예> XOR)

다층 신경망의 분류 원리

다층 신경망은 입력층 – 은닉층 – 출력층 구조를 가지며 각 층의 유닛들은 퍼셉트론임
레이어를 쌓을수록 다양한 결정 영역이 추가로 생김으로써 선형 분리 문제 극복

인공지능을 위한 머신러닝 알고리즘

7. 역전파

Tacademy

CONTENTS

학습 목표

- 역전파 알고리즘을 이해하고 다층 퍼셉트론의 파라미터 값을 계산할 수 있다.
- 활성함수 미분값의 특징을 이해할 수 있다.

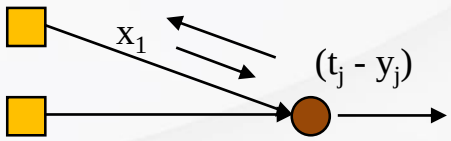
1. 역전파 학습 방법

다층 퍼셉트론

다층 퍼셉트론의 적절한 가중치를 어떻게 찾을 수 있을까?

- 단층 퍼셉트론 모델에서 적절한 가중치를 찾기 위해 경사 하강법 (**Gradient Descent**)을 사용

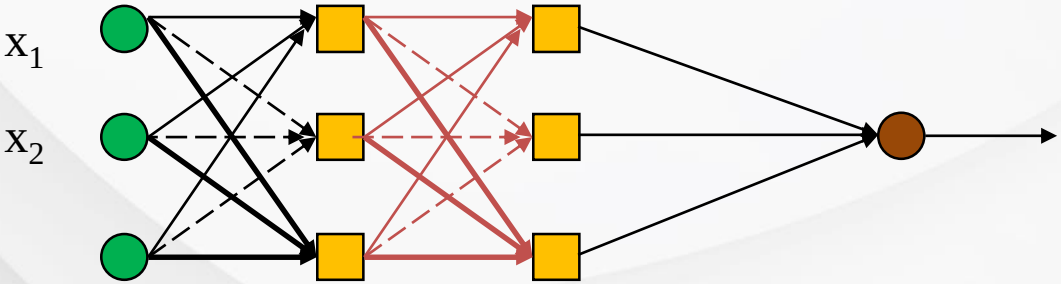
$$\Delta w_{ji} = (t_j - y_j)\, x_i$$



- 노드 **i**와 출력 **j**를 연결하는 가중치 w_{ji}는
  출력 **j**로부터 받은 에러 신호$(t_j - y_j)$와 노드의 입력(x_i)에 의해서만 영향을 받음

다층 퍼셉트론

다층 퍼셉트론의 적절한 가중치를 어떻게 찾을 수 있을까?



- 위와 같이 여러 층을 갖는 다층 퍼셉트론에서 에러가 세번째 층에서만 계산된다면 처음 두 개의 층은 어떻게 가중치를 학습할까?
- 입력층에서는 직접적인 에러 신호 $(t_j - y_j)$ 가 존재하지 않음

기여도 할당 문제 (Credit Assignment Problem)

- 전체 학습 모델을 구성하는데 관여하고 있는 모든 개별 요소들

  예> 은닉 유닛들)에 '기여도' 또는 '책임'을 할당하는 문제

- 다층 신경망에서는 어떤 가중치들을 얼마큼, 어떤 방향으로 학습시켜야 하는지 관련됨
- 앞부분 층의 가중치들이 최종 출력(또는 에러)에 얼마큼 영향을 미치는 결정하는 문제와 비슷
- 가중치 w_{ji}가 에러에 미치는 영향을 계산해야 함

$$\frac{\partial E(t)}{\partial w_{ij}(t)}$$

역전파 (Backpropagation)

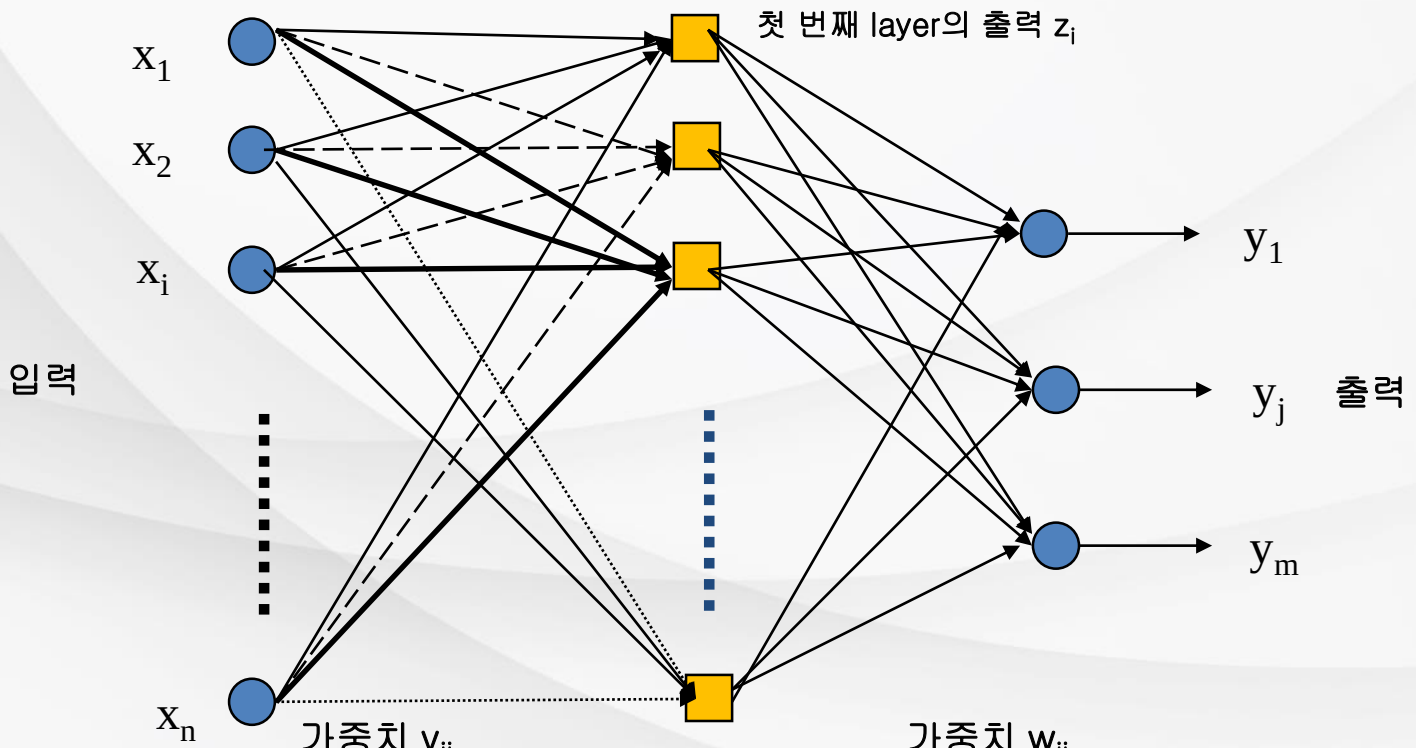
- 다층 퍼셉트론에서 기여도 할당 문제에 대한 해결책

  Rumelhart, Hinton and Williams (1986)

- 역전파는 두 단계로 나뉘어짐

1. 앞먹임 단계	입력 값들을 사용하여 다층 퍼셉트론의 최종 출력을 계산
2. 오류 후방 전파 단계	에러 값을 계산한 뒤, 최종 출력 유닛들부터 시작하여 네트워크의 후방으로 에러 값을 전파

입력층 - 은닉층 – 출력층 구조를 갖는 다층 퍼셉트론의 모습



가중치 v_{ij}
– 입력층의 j번째 노드와
은닉층의 i번째 노드의 연결 가중치

가중치 w_{ij}
– 은닉층의 j번째 노드와
출력층의 i번째 노드의 연결 가중치

다층 퍼셉트론의 노드 활성 값 계산

$$z_i(t) = g(\Sigma_j \; v_{ij}(t)\, x_j(t)) \quad \text{at time t}$$
$$= g(u_i(t))$$

$$y_i(t) = g(\Sigma_j \; w_{ij}(t)\, z_j(t)) \quad \text{at time t}$$
$$= g(a_i(t))$$

- g는 활성함수 예> 시그모이드 함수
- **Bias**는 추가 가중치로 여겨짐

역전파 – (1) 앞먹임 단계

1. 은닉 유닛들의 값 계산

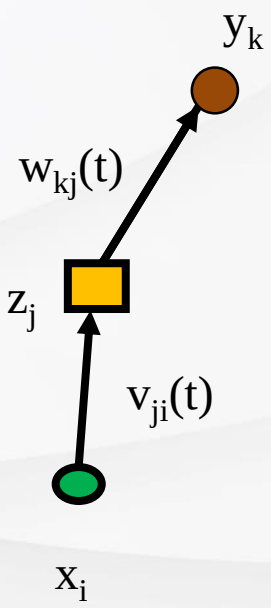
$$u_j(t) = \sum_i v_{ji}(t) x_i(t)$$

$$z_j = g(u_j(t))$$

2. 출력 유닛들의 값 계산

$$a_k(t) = \sum_j w_{kj}(t) z_j$$

$$y_k = g(a_k(t))$$



역전파 – (2) 오류 후방 전파 단계

- 에러의 제곱의 합을 사용할 경우, 다음과 같은 손실 함수 식을 얻음

$$E(t) = \frac{1}{2}\sum_{k=1}(d_k(t) - y_k(t))^2$$

- d_k는 타겟 벡터의 k차원의 값
- E를 줄이기 위해 경사 하강법을 사용하여 가중치를 변경함

$$w_{ij}(t+1) - w_{ij}(t) \propto -\frac{\partial E(t)}{\partial w_{ij}(t)}$$

- 출력 유닛과 은닉 유닛 모두 적용

역전파 – (2) 오류 후방 전파 단계

편미분 방정식은 체인룰을 사용하여 두 개 항의 곱으로 나타낼 수 있음

$$\frac{\partial E(t)}{\partial w_{ij}(t)} = \frac{\partial E(t)}{\partial a_i(t)} \bullet \frac{\partial a_i(t)}{\partial w_{ij}(t)}$$

출력 유닛과 은닉 유닛 모두 적용

은닉 유닛:

$$u_j(t) = \sum_i v_{ji}(t) x_i(t)$$

$$z_j = g(u_j(t))$$

출력 유닛:

$$a_k(t) = \sum_j w_{kj}(t) z_j$$

$$y_k = g(a_k(t))$$

Term A i 번째 출력 유닛의 $a_i(t)$ 값에 대한 에러 값의 변화량

Term B i 번째 출력 유닛에 연결되어있는 j 번째 가중치에 대한 $a_i(t)$의 변화량

역전파 – (3) 손실 함수의 미분

B항

$$\frac{\partial u_i(t)}{\partial v_{ij}(t)} = x_j(t)$$

은닉 유닛의 경우

$$\frac{\partial a_i(t)}{\partial w_{ij}(t)} = z_j(t)$$

출력 유닛의 경우

은닉 유닛:

$$u_j(t) = \sum_i v_{ji}(t) x_i(t)$$

$$z_j = g(u_j(t))$$

출력 유닛:

$$a_k(t) = \sum_j w_{kj}(t) z_j$$

$$y_k = g(a_k(t))$$

A항

$$\frac{\partial E(t)}{\partial u_i(t)}$$

은닉 유닛의 경우

$$\frac{\partial E(t)}{\partial a_i(t)}$$

출력 유닛의 경우

체인룰에 의해서 계산 가능

역전파 – (3) 손실 함수의 미분

각 출력 유닛에 대하여 아래의 식을 계산

$$\Delta_j(t) = \frac{\partial E(t)}{\partial a_j(t)} = g'(a_j(t))\frac{\partial E(t)}{\partial y_j(t)}$$

$$\Delta_j(t) = -g'(a_j(t))(d_j(t) - y_j(t))$$

은닉 유닛:

$$u_j(t) = \sum_i v_{ji}(t)x_i(t)$$

$$z_j = g(u_j(t))$$

출력 유닛:

$$a_k(t) = \sum_j w_{kj}(t)z_j$$

$$y_k = g(a_k(t))$$

손실 함수: $E(t) = \frac{1}{2}\sum_{k=1}(d_k(t) - y_k(t))^2$

역전파 – (3) 손실 함수의 미분

각 은닉 유닛에 대하여 체인룰 사용

$$a_j(t) = \sum_m w_{jm}(t) g(u_m(t))$$

$$\delta_j(t) = \frac{\partial E(t)}{\partial u_j(t)} = \sum_j \frac{\partial E(t)}{\partial a_j(t)} \frac{\partial a_j(t)}{\partial u_j(t)}$$

$$\delta_j(t) = g'(u_j(t)) \sum_j w_{ji} \Delta_j$$

은닉 유닛:

$$u_j(t) = \sum_i v_{ji}(t) x_i(t)$$

$$z_j = g(u_j(t))$$

출력 유닛:

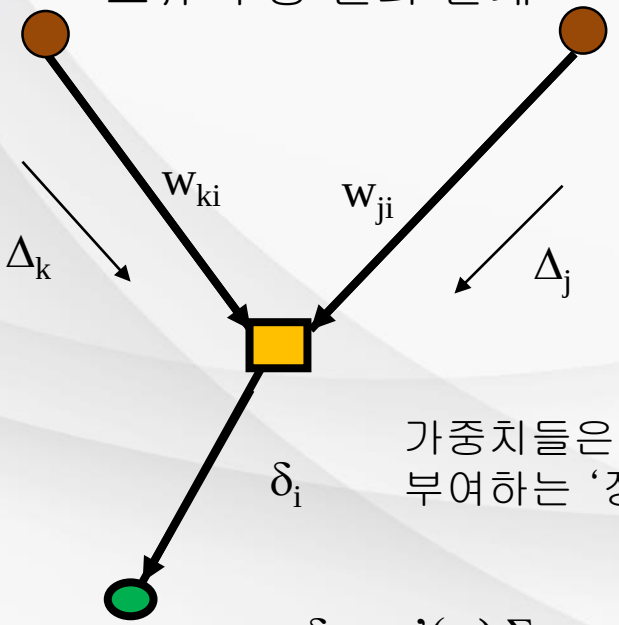
$$a_k(t) = \sum_j w_{kj}(t) z_j$$

$$y_k = g(a_k(t))$$

손실 함수: $E(t) = \frac{1}{2} \sum_{k=1} (d_k(t) - y_k(t))^2$

역전파 – (4) 기여도 할당 문제의 해결

오류 후방 전파 단계



가중치들은 은닉 유닛들에게 '기여도' 또는 '책임'을 부여하는 '정도'의 값으로 해석할 수 있음

$$\delta_i = g'(u_i)\, \Sigma_j\, w_{ji}\, \Delta_j$$

역전파 – (5) 가중치 업데이트

A와 B를 결합

$$\frac{\partial E(t)}{\partial v_{ij}(t)} = \delta_i(t) x_j(t)$$

$$\frac{\partial E(t)}{\partial w_{ij}(t)} = \Delta_i(t) z_j(t)$$

A B

$$\frac{\partial E(t)}{\partial v_{ij}(t)} = \frac{\partial E(t)}{\partial u_i(t)} \bullet \frac{\partial u_i(t)}{\partial v_{ij}(t)}$$

$$\frac{\partial E(t)}{\partial w_{ij}(t)} = \frac{\partial E(t)}{\partial a_i(t)} \bullet \frac{\partial a_i(t)}{\partial w_{ij}(t)}$$

E에 대한 경사 하강법을 하기 위해서 가중치를 다음과 같이 변경해야 함

$$v_{ij}(t+1) - v_{ij}(t) = \eta\, \delta_i(t)\, x_j(n)$$
$$w_{ij}(t+1) - w_{ij}(t) = \eta\, \Delta_i(t)\, z_j(t)$$

η은 학습률을 나타내는 파라미터 (0 < η <=1)

역전파 – (5) 가중치 업데이트

❖ 가중치 학습식

$$v_{ij}(t+1) - v_{ij}(t) = \eta \delta_i(t) x_j(t)$$

$$w_{ij}(t+1) - w_{ij}(t) = \eta \Delta_i(t) z_j(t)$$

역전파 – (5) 가중치 업데이트

❖ 출력 유닛

$$w_{ij}(t+1) - w_{ij}(t) = \eta \Delta_i(t) z_j(t)$$
$$= \underbrace{\eta(d_i(t) - y_i(t))}_{\text{에러의 크기}} \underbrace{g'(a_i(t))}_{\text{활성함수의 미분값}} \underbrace{z_j(t)}_{\text{입력의 크기}}$$

❖ 은닉 유닛

$$v_{ij}(t+1) - v_{ij}(t) = \eta \delta_i(t) x_j(t)$$
$$= \underbrace{\eta g'(u_i(t))}_{\text{활성함수의 미분값}} \underbrace{x_j(t)}_{\text{입력의 크기}} \underbrace{\sum_k \Delta_k(t) w_{ki}}_{\text{상위층 유닛들의 기여도를 가중치에 따라 평균 낸 값}}$$

2. 활성함수의 미분값

2. 활성함수의 미분값

활성함수에 대한 에러의 변화량

활성함수가 에러의 변화량에 얼만큼 영향을 미칠까?

$$\Delta_i(t) = (d_i(t) - y_i(t))g'(a_i(t))$$

$$\delta_i(t) = g'(u_i(t))\sum_k \Delta_k(t) w_{ki}$$

Where : $g'(a_i(t)) = \dfrac{dg(a)}{da}$

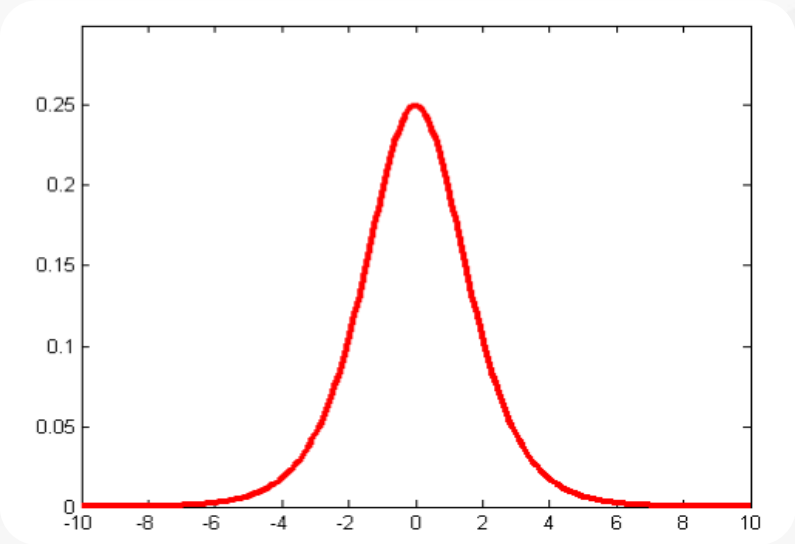
- 활성함수 **g**에 대한 미분을 계산해야 함
- 미분이 가능하기 위해서 활성함수는 '**smooth**'해야 함

2. 활성함수의 미분값

시그모이드 함수

$$g'(a_i(t)) = \frac{k\exp(-k\,a_i(t))}{[1+\exp(-k\,a_i(t))]^2} = kg(a_i(t))[1-g(a_i(t))]$$

since: $y_i(t) = g(a_i(t))$ we have: $g'(a_i(t)) = ky_i(t)(1-y_i(t))$



2. 활성함수의 미분값

활성함수 미분값과 가중치의 관계

❖ 가중치의 변화량은 활성함수 미분값의 비례

$$\Delta_i(t) = (d_i(t) - y_i(t))g'(a_i(t))$$

$$\delta_i(t) = g'(u_i(t))\sum_k \Delta_k(t) w_{ki}$$

- 가중치의 학습은 유닛의 $a_i(t)$ 또는 $u_i(t)$ 값이 너무 크거나 작지 않을 경우 잘 됨
- 값이 너무 크거나 작을 때 미분값은 **0**에 가까워짐
- 딥신경망이 학습이 잘 안되었던 이유

학습정리

학습정리

Tacademy

지금까지 [역전파]에 대해서 살펴보았습니다.

역전파 학습 방법

앞먹임 단계: 입력 값들을 사용하여 다층 퍼셉트론의 최종 출력을 계산
오류 후방 전파 단계: 에러값을 계산한 뒤, 최종 출력 유닛들부터 시작하여 네트워크의 후방으로 에러값을 전파

역전파 학습 방법

은닉 유닛의 가중치를 학습하기 위해서 체인룰을 사용,
은닉 유닛의 출력 유닛들에 대한가중치들은 해당 은닉 유닛에 대한 '기여도' 또는 '책임'을 부여하는 '정도'의 값으로 해석할 수 있음

활성함수의 미분값

시그모이드 함수의 미분 형태 $g'(a_i(t)) = k(1 - \frac{1}{1+e^{-k\,a_i(t)}})\frac{1}{1+e^{-k\,a_i(t)}}$

인공지능을 위한 머신러닝 알고리즘

8. 비지도 학습

Tacademy

CONTENTS

학습 목표

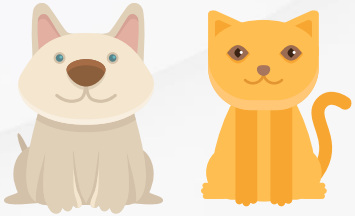
- 클러스터링과 비지도 학습의 관계를 이해할 수 있다.
- K-means 알고리즘의 클러스터링 과정을 이해할 수 있다.
- 데이터 포인트들 사이 거리 측정 알고리즘과 사용법을 이해할 수 있다.

1. 클러스터링

클러스터링이란?

cluster

- 클러스터링은 데이터에서 '클러스터(Clusters)'라는 '비슷한 그룹'을 찾는 기법을 뜻함
- 클러스터링은 서로 생김새가 비슷한 데이터끼리 하나의 클러스터로 묶고, 생김새가 매우 다른 데이터끼리 다른 클러스터로 분류 (類類相從, 가재는 게 편 등..)



클러스터링 = **비지도 학습 (Unsupervised Learning)**

- 데이터의 그룹을 묶을 수 있는 어떠한 사전 정보도 주어지지 않기 때문
- 사전 정보 (예> 레이블)이 주어지면 지도 학습임
- 이러한 이유 때문에, 클러스터링과 비지도 학습은 동의어로 여겨지기도 함

일상 속 클러스터링의 예

예제 1 : “small”, “medium”, “large” 티셔츠를 만들기 위해서 사람들을 비슷한 크기로 그룹을 지음

- 각 사람 크기에 맞는 옷을 만들려면 너무 많은 비용이 듦
- 한 사이즈로 통일하기에는 맞지 않는 경우가 많음

예제 2 : 마케팅을 하기 위해서 고객들을 비슷한 정도에 따라 여러 분류로 나눔

- 고객의 유형에 따라 마케팅 전략을 세움

일상 속 클러스터링의 예

예제 3 : 문서가 많을 때, 내용의 비슷한 정도에 따라서 하나의 파일로 묶음

- 주제에 따라서 계층적 구조를 띄기도 함

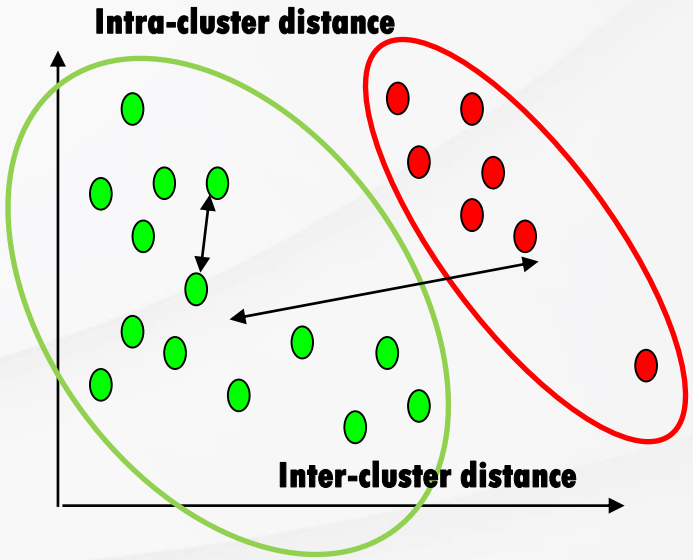
- 클러스터링은 일상생활 속에서 가장 많이 사용되는 데이터 마이닝 기법 중 하나
- 인터넷을 통한 온라인 문서들의 급속한 증가로 인해서 문서 분류가 중요한 이슈가 됨

클러스터링 이슈

- 어떻게 그룹을 나눌 것인가?
- 데이터들의 비슷한 정도(distance, similarity)를 어떻게 측정할 것인가?
- 클러스터링이 잘 되었는지 어떻게 평가할 것인가?

Inter-clusters distance → 최대화

Intra-clusters distance → 최소화



"클러스터링 결과의 질은 알고리즘, 거리 측정 방법 등에 따라 좌우됨"

2. K-means 클러스터링

K-means 클러스터링이란?

- 데이터 포인터들의 집합 **D**를 다음과 같이 정의하자
  - $\{x_1, x_2, \ldots, x_n\}$, 여기서 $x_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$는 실수값을 갖는 벡터
  - $X \subseteq R^r$, r은 데이터 속성의 개수

- **K-means** 알고리즘은 주어진 데이터를 **k**개의 클러스터로 분류
  - 각 클러스터는 **Centroid**라고 불리는 중심점 (**Center**)를 가짐
  - K의 값은 프로그래머가 정할 수 있는 가변 값임

K-means 클러스터링이란?

- K가 주어졌을 때, 알고리즘은 다음의 과정을 거침

1. 데이터 포인트들 중에 무작위로 K개를 선택하여 **Centroid**로 정함
2. 각 데이터 포인트들을 K개의 **Centroid**로 할당함
3. 각 클러스터의 구성원들을 기반으로 **Centroid**를 다시 계산
4. 수렴 조건이 만족되지 않으면 **2**번으로 감

수렴 조건

01 다른 클러스터들로 재배치되는 데이터 포인터들이 존재하지 않음

02 Centroids가 변경되지 않음

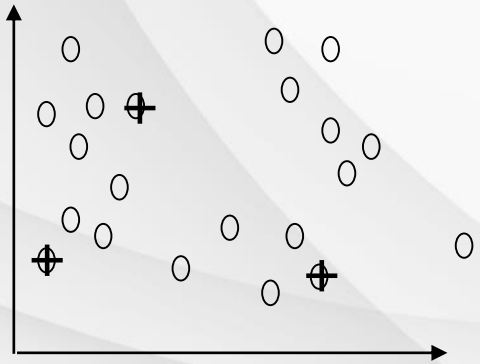
03 Sum of Squared Error (SSE)가 최저 임계치에 도달한 경우

수렴 조건

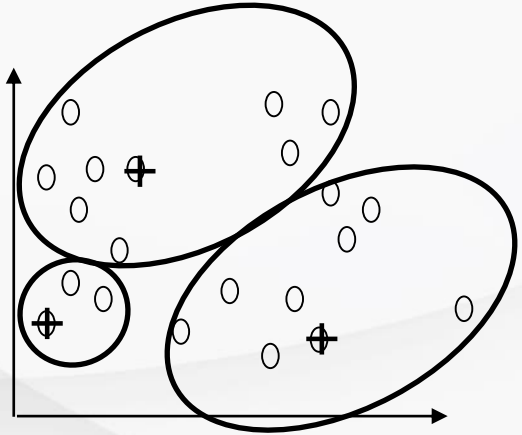
$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2$$

- C_j는 j번째 클러스터를 뜻함
- m_j는 클러스터 C_j의 centroid (클러스터 C_j에 있는 모든 데이터들의 평균 벡터)
- dist (x, m_j)는 데이터 포인트 x와 centroid m_j사이의 거리

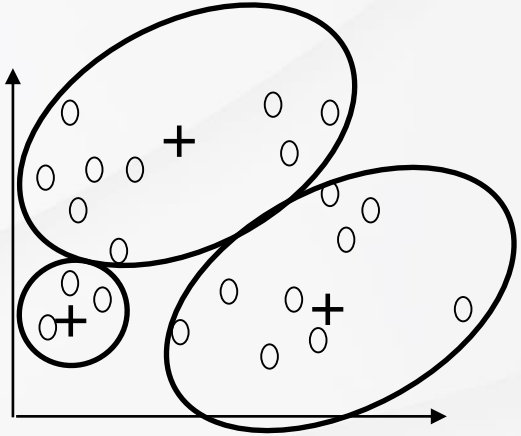
K-means의 예



1. K개의 중심을 무작위 선택

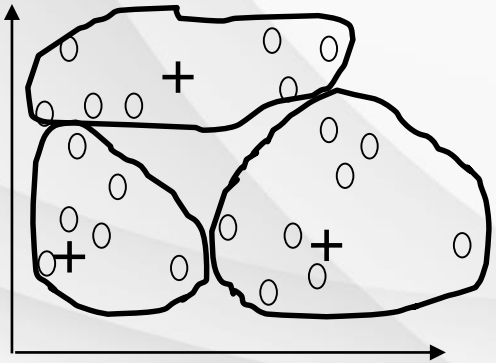


2. 클러스터 배정

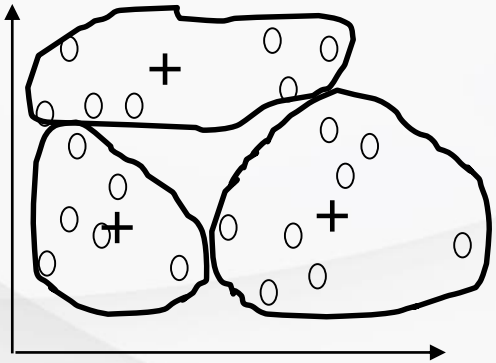


3. Centroid를 다시 계산

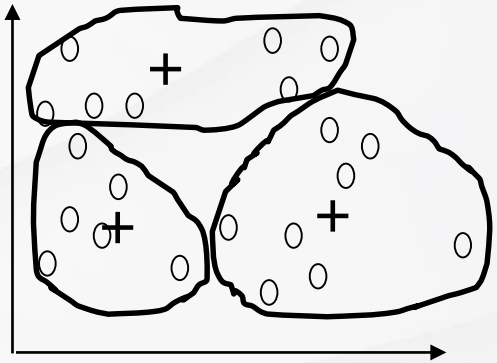
K-means의 예



4. 클러스터 재배정



5. Centroids 다시 계산



6. 클러스터 재배정

7. 클러스터링 종료

거리 측정 함수(Distance Function)의 예

- K-means 알고리즘은 데이터 집합에서 평균을 정의하고 계산할 수 있으면 사용할 수 있음
- 유클리디안 공간 (Euclidean Space)에서 클러스터의 평균은 다음과 같이 계산

$$\mathbf{m_j} = \frac{1}{|C_j|}\sum_{\mathbf{x_i}\in C_j} x_i$$

$|C_j|$는 클러스터 C_j에 존재하는 데이터 포인트 개수

- 하나의 데이터 포인터 x_i로부터 $centroid\mathbf{M_j}$ 까지의 거리는 다음과 같이 계산

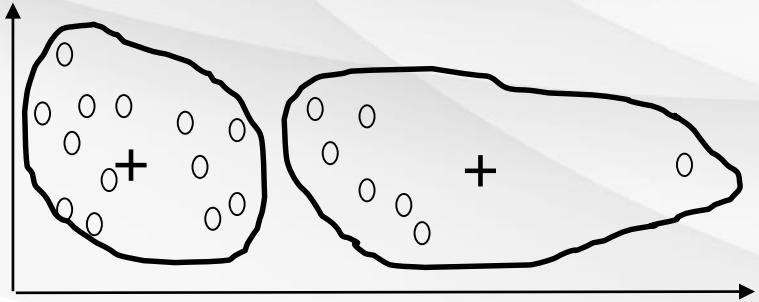
$$\begin{aligned} dist(x_i, m_j) &= |x_i, mj| \\ &= \sqrt{(x_{i1} - m_{ji})^2 + (x_{i2} - m_{j2})^2 + \ldots + (x_{ir} - m_{jr})^2} \end{aligned}$$

K-means의 장점

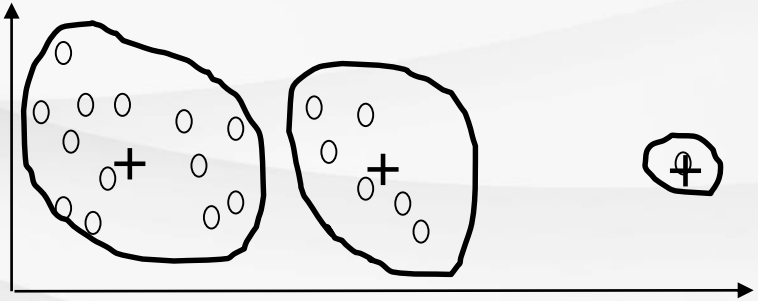
- 이해하고 구현하기 쉬움
- 효율적인 시간 복잡도를 가짐: **O(tkn)**
  - **n**은 데이터 포인터들의 개수
  - **k**는 클러스터의 개수
  - **t**는 클러스터 재배정 반복 횟수
- **k**와 **t**의 값이 작으므로 **k-means** 알고리즘은 데이터 개수에 따라 선형 복잡도를 갖는 알고리즘으로 볼 수 있음
- **K-means**는 가장 널리 사용되는 클러스터링 알고리즘
- **Sum of Squared Error**를 사용할 경우 지역적 최적화에서 종료될 수 있음, 전역적 최적점은 찾기가 어려움

K-means의 단점

- 데이터의 평균 값이 정의될 수 있는 데이터에만 사용 가능
- K의 값은 프로그래머의 몫 → 가장 최적의 K의 값을 찾기 어려움
- 아웃라이어에 매우 민감함
  - 아웃라이어 데이터란 다른 데이터 포인트들과 매우 동떨어져 있는 데이터를 뜻함
  - 아웃라이어 데이터는 데이터 기록 과정 중 벌어지는 오류 또는 독특한 성격을 갖는 이종 데이터로 인해 발생할 수 있음



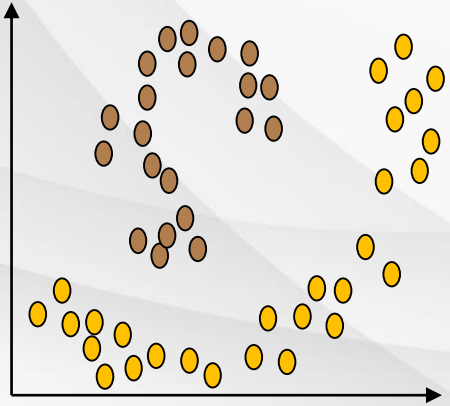
올바르지 못한 클러스터링



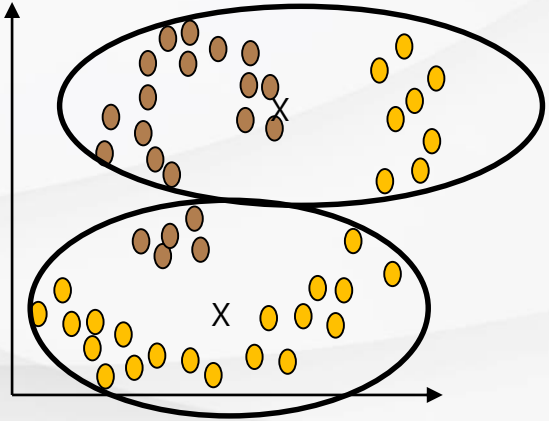
바람직한 클러스터링

K-means의 단점

K-means 알고리즘은 타원체 모양이 아닌 클러스터들을 찾는 문제에 적합하지 않음
(예> 고리 모양)



두 개의 고리 모양 클러스터



K-means 클러스터

K-means의 단점을 다루기 위한 방법

- 클러스터링 도중 다른 데이터 포인트보다 Centroid로부터 거리가 비이상적으로 먼 데이터 포인트를 제거해나감
  - 안전한 방법은 클러스터링 도중 아웃라이어가 발생하는지 모니터링 하다가 발생하면 지울지 말지 직접 결정
- 데이터로부터 무작위로 샘플링함. 샘플링은 전체 데이터 중 일부만 선택하기 때문에 아웃라이어가 선택될 확률은 낮음
  - 클러스터링이 종료되면, 샘플링 되지 않은 데이터 포인트들을 정해진 클러스터로 배정시킴

3. 거리 측정 함수들

3. 거리 측정 함수들

데이터의 특성이 수치값 (Numeric Value)을 가질 때

- 유클리디안 거리 측정 (**Euclidean Distance**)과
  맨허튼 거리 측정 (**Manhattan Distance**)이 주로 사용됨

- 두 개의 데이터 포인트(x_i와 x_j)들의 거리를 측정할 때 다음과 같이 표기 $dist(x_i, x_j)$

- 두 측정 방법은 **Minkowski Distance**의 특별한 예

$$dist(x_i, x_j) = ((x_i, x_j)^h + (x_{i2}, x_{j2})^h + \cdots + (x_{ir}, x_{jr})^h)^{\frac{1}{h}}$$

3. 거리 측정 함수들

- 데이터의 특성이 수치값 **(Numeric Value)**을 가질 때

$h = 2$인 경우 유클리디안 거리 측정

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{ir} - x_{jr})^2}$$

$h = 1$인 경우 맨허튼 거리 측정

$$dist(\mathbf{x}_i, \mathbf{x}_j) = | x_{i1} - x_{j1} | + | x_{i2} - x_{j2} | + \ldots + | x_{ir} - x_{jr} |$$

가중치 적용 유클리디안 거리 측정

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + \ldots + w_r(x_{ir} - x_{jr})^2}$$

3. 거리 측정 함수들

- 데이터의 특성이 수치값 **(Numeric Value)**을 가질 때

제곱 유클리디안(*Squared Euclidean distance*) 거리

멀리 떨어져 있는 데이터 포인터들에게 더 많은 가중치를 줄 경우

$$dist(\mathbf{x}_i, \mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{ir} - x_{jr})^2$$

Chebychev distance

데이터의 특성들 중 어느 하나라도 다를 경우, '다름'으로만 정의하고자 하는 경우

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \max(|x_{i1} - x_{j1}|, |x_{i2} - x_{j2}|, \ldots, |x_{ir} - x_{jr}|)$$

데이터의 특성이 이산값 (Binary Value)을 가질 때

- 이산적 특성: 두 개의 값 또는 상태를 가짐

  예 > 성별: 남자, 여자

- Confusion 행렬을 사용하여 거리 함수를 정의함

		데이터 포인트 i		
		1	0	
데이터 포인트 j	1	a	b	a+b
	0	c	D	c+d
		a+c	b+d	a+b+c+d

a : 두 개의 데이터 포인트들이 모두 1값을 갖는 속성의 개수
b : 데이터 포인트 j는 1값을 갖고 i는 0값을 갖는 속성의 개수
c : 데이터 포인트 j는 0값을 갖고 i는 1값을 갖는 속성의 개수
d : 두 개의 데이터 포인트들이 모두 0 값을 갖는 속성의 개수

데이터의 특성이 이산값 (Binary Value)을 가질 때

- 이산적 특성은 두 개의 상태 (0 또는 1)이 동일하게 중요할 때, 대칭적임 (동일한 가중치)
- 거리함수: 단순 계수 비교 (일치하지 않은 값의 비율)

$$dist(x_i, xj) = \frac{b+c}{a+b+c+d}$$

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

$$dist(x_i, xj) = \frac{2+1}{2+2+1+2} = 3/7$$

데이터의 특성이 이산값 (Binary Value)을 가질 때

- 이산적 특성은 두 개의 상태 (0 또는 1)이 비대칭적일 때
- 한 상태가 다른 상태보다 더 중요한 경우
- 일반적으로 상태 1이 더 중요한 경우에 사용됨 (빈도가 더 낮아 희귀한 상태)
- Jaccard 계수 측정이 사용됨

$$dist(x_i, xj) = \frac{b+c}{a+b+c}$$

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

$$dist(x_i, xj) = \frac{2+1}{2+2+1} = 3/5$$

학습정리

학습정리

Tacademy

지금까지 [비지도 학습]에 대해서 살펴보았습니다.

클러스터링

클러스터링은 데이터에서 '클러스터(Clusters)'라는 '비슷한 그룹'을 찾는 기법을 뜻함
데이터의 그룹을 묶을 수 있는 어떠한 사전 정보도 주어지지 않기 때문에
비지도 학습(Unsupervised Learning)과 동의어로 사용됨

K-means 클러스터링

데이터 포인트들을 무작위로 K개 선택하여 Centroid 계산 →
클러스터 배정과 Centroid 재계산 (수렴 조건이 만족될 때까지 반복)
아웃라이어에 약한점에도 불구하고 단순함과 효율성으로 인해 널리 사용됨

거리 측정 함수들

수치적 특성과 이산적 특성에 따라 다양한 거리 함수 사용
수치적 특성: 유클리디안, 제곱 유클리디안, 멘허튼, Chebychev 거리 측정
이산적 특성: 단순 계수 비교, jaccard 계수 비교

인공지능을 위한 머신러닝 알고리즘

9. 컨볼루션 신경망

Tacademy

CONTENTS

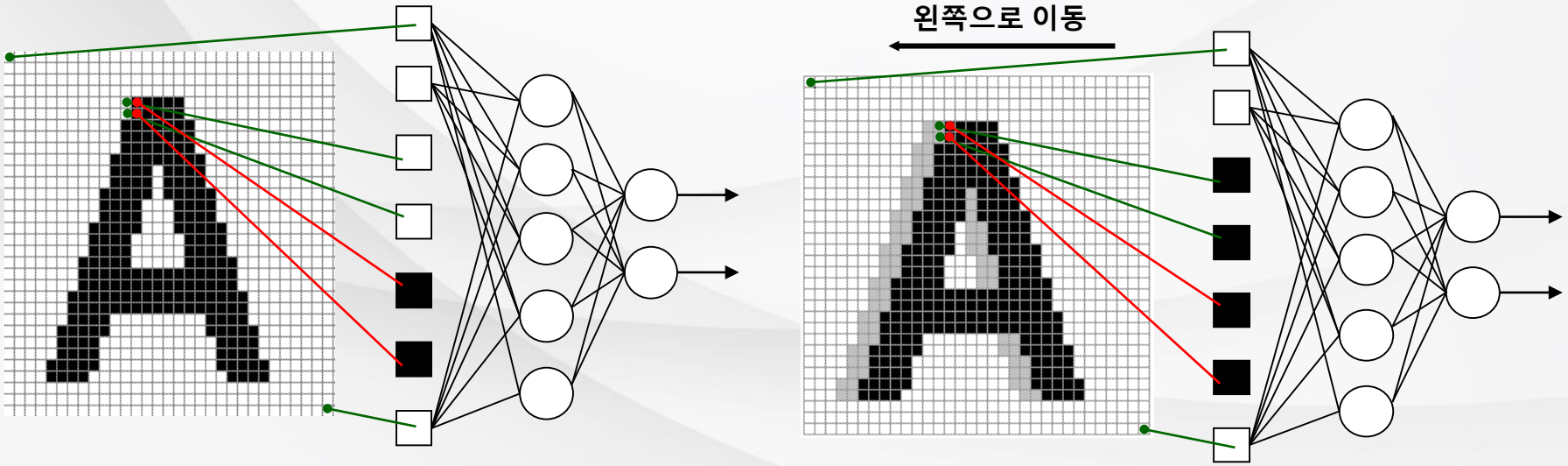
학습 목표

- 컨볼루션 신경망의 분류 원리를 이해할 수 있다.
- 기존 신경망과 컨볼루션 신경망의 구조적 차이를 이해할 수 있다.
- 컨볼루션 신경망 구조와 깊이의 진화 과정을 이해할 수 있다.

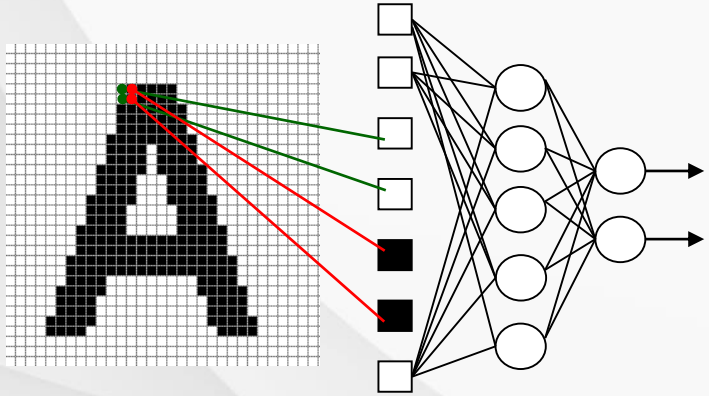
1. 컨볼루션 신경망의 원리

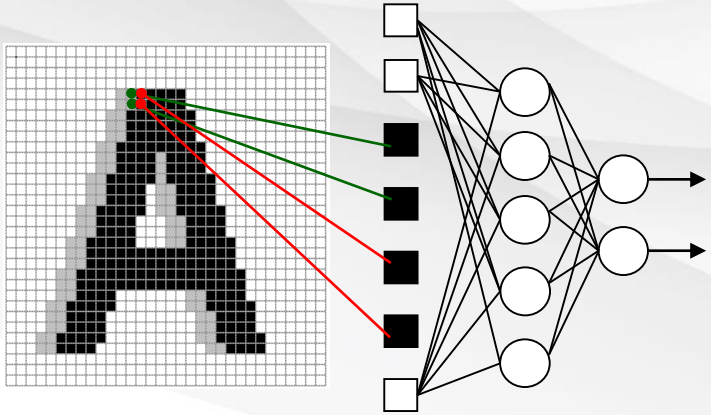
일반 신경망으로 이미지를 분류할 때 문제점

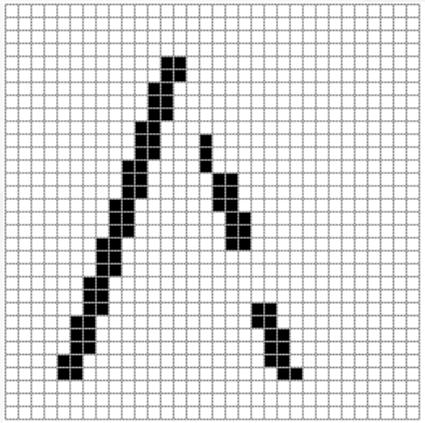
- 이미지의 위치, 크기, 각도 변화 등에 취약함



일반 신경망으로 이미지를 분류할 때 문제점





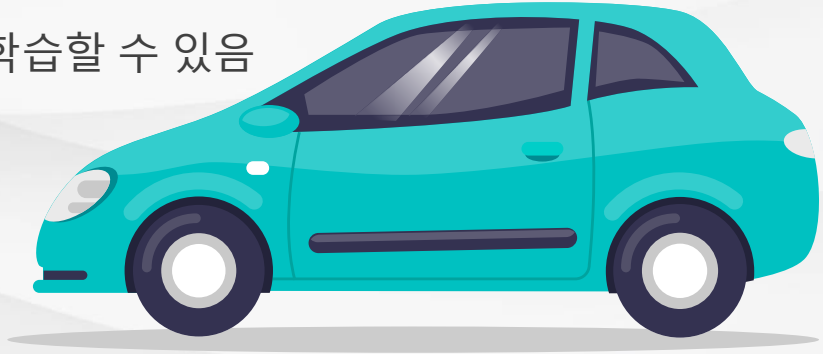


왼쪽으로 2픽셀 이동할 경우
154개의 입력이 변화함
77개 : 검은색 → 흰색
77개 : 흰색 → 검은색

- 크기 또는 모양이 변화하는 경우

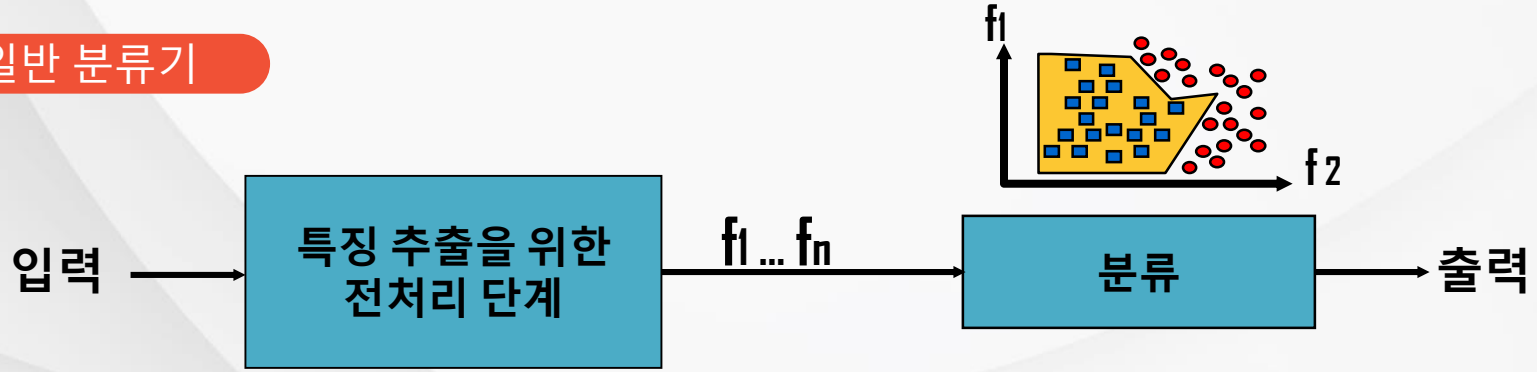
컨볼루션 신경망이란?

- 우리 뇌 속 시각 피질의 신경 세포들은 물체의 방향과 장소가 바뀌어도 별 문제 없이 인식할 수 있음
- 이러한 신경생물학적 관찰이 컨볼루션 신경망 구조 설계에 동기를 부여함
- 컨볼루션 신경망은 물체의 위치와 방향에 관계없이 물체의 고유한 특징을 학습할 수 있음
- 다층 신경망의 한 종류임
- 역전파 알고리즘을 사용하여 학습
- 이미지의 픽셀 값으로부터 직접 시각 패턴을 학습할 수 있음

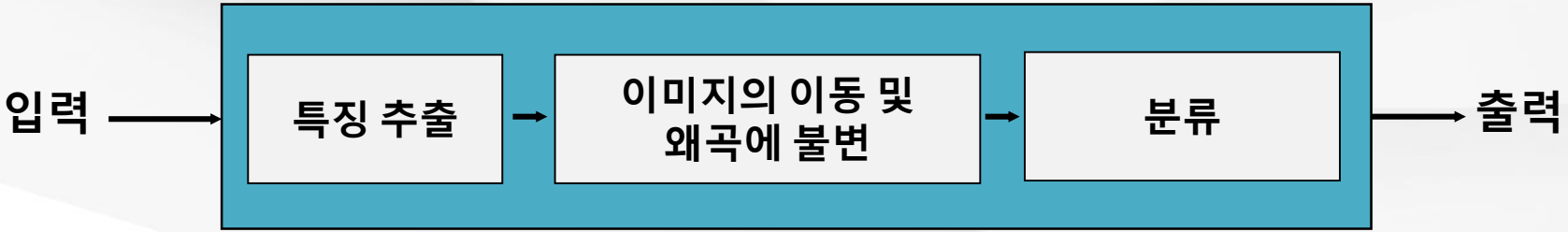


분류 과정

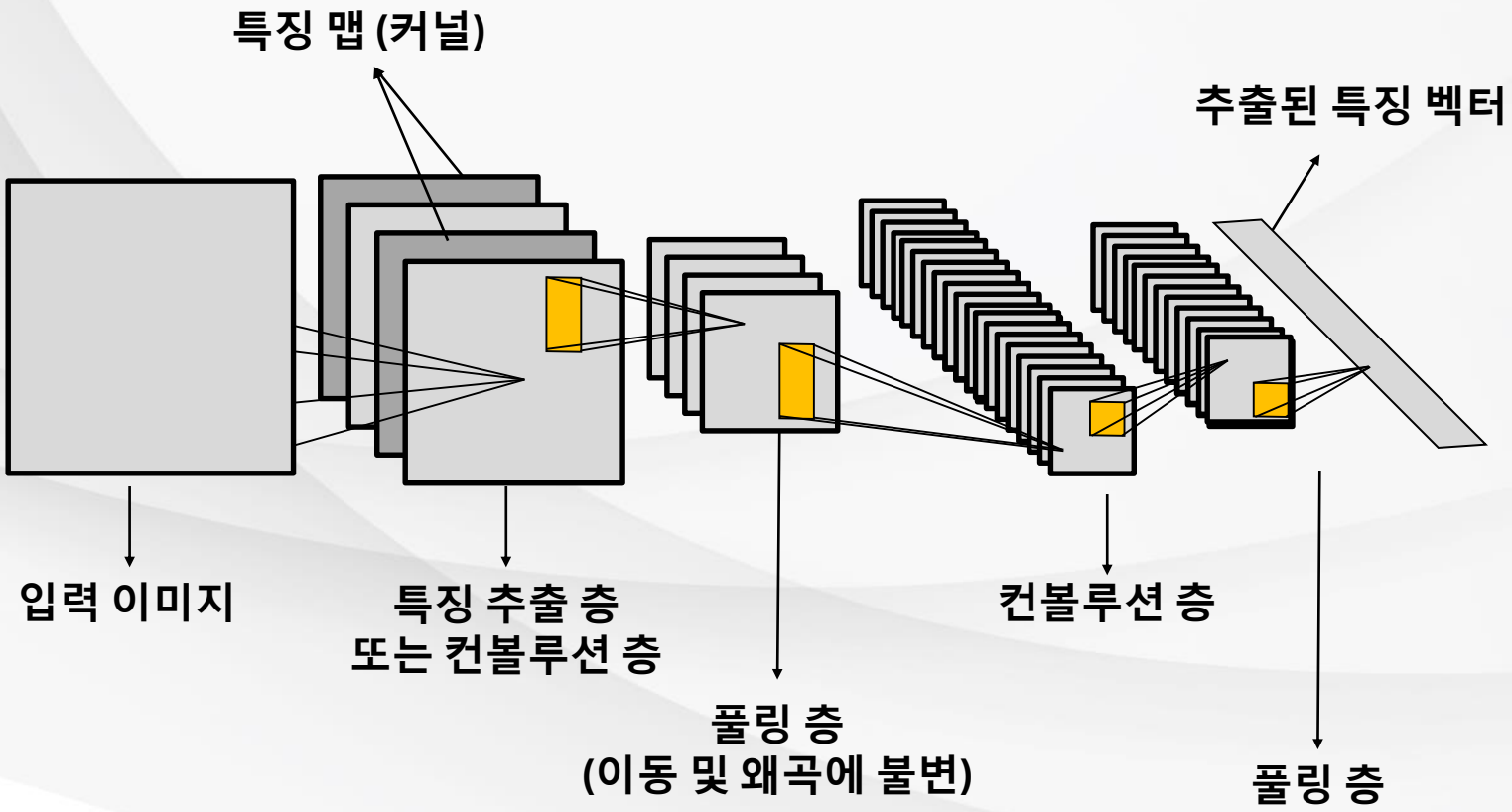
일반 분류기



컨볼루션
신경망

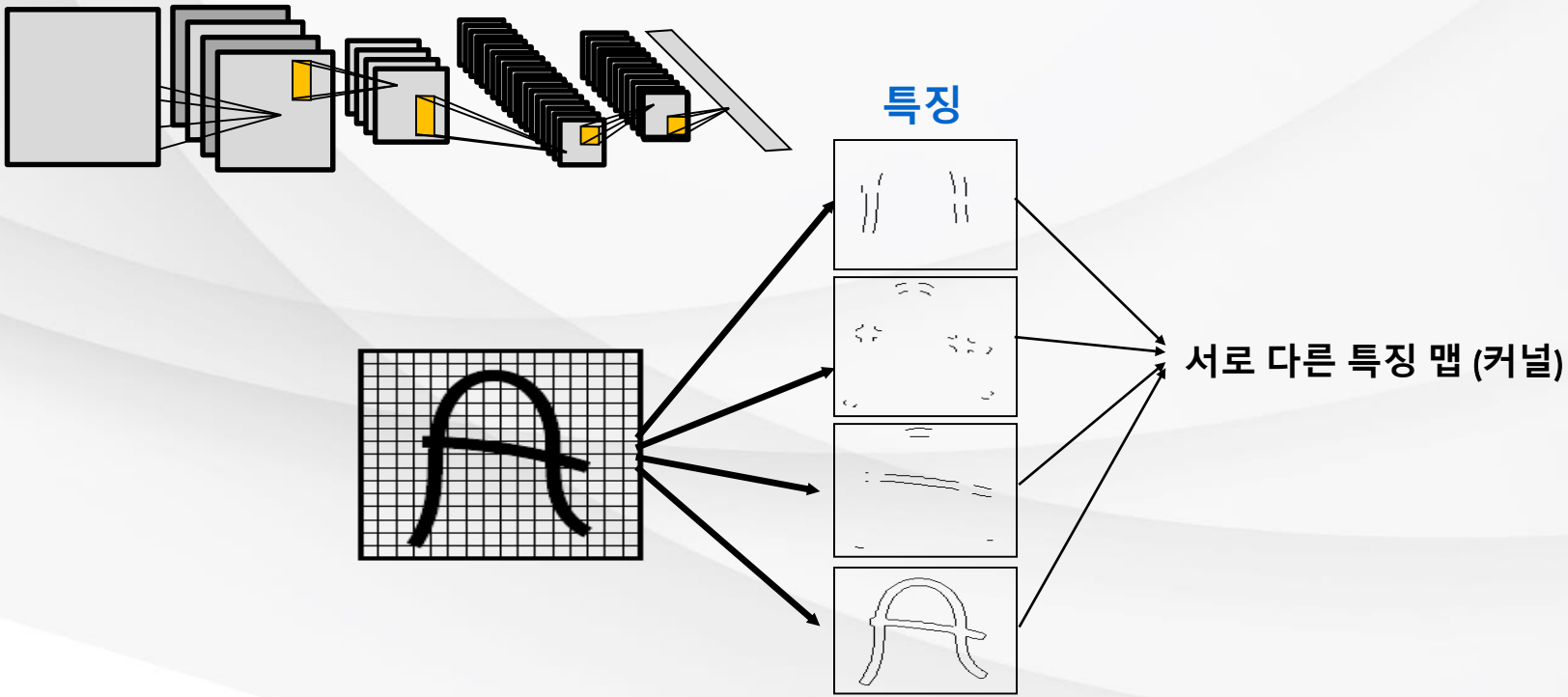


컨볼루션 신경망의 구조

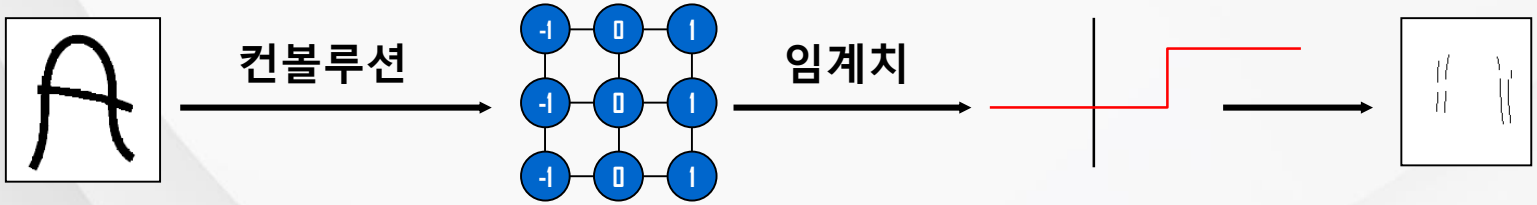


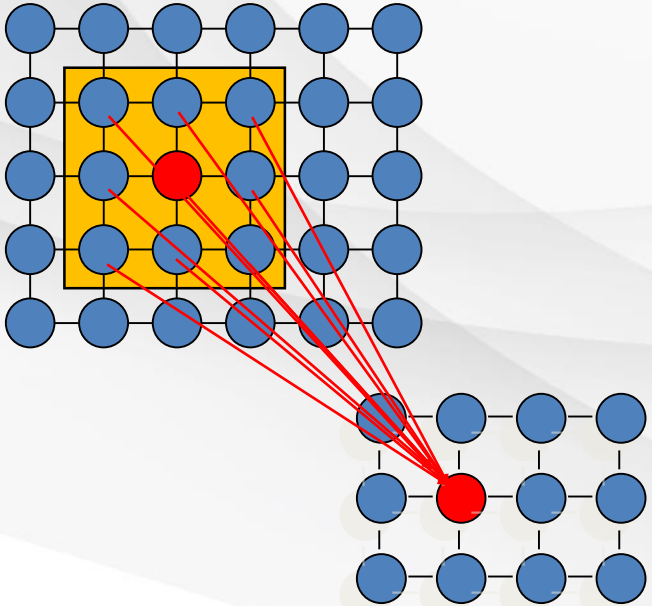
컨볼루션 층

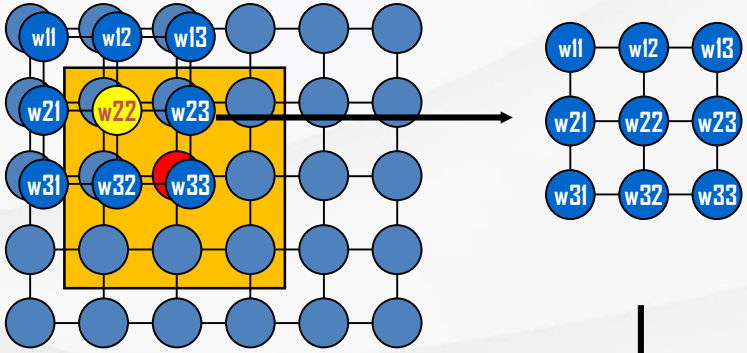
입력 이미지 속 다양한 위치에서 동일한 특징들을 탐색

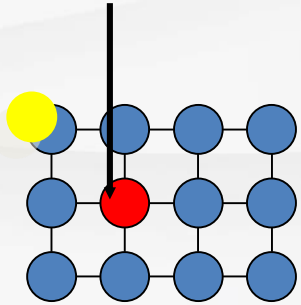


컨볼루션 과정


컨볼루션
-1 0 1
-1 0 1
-1 0 1
임계치

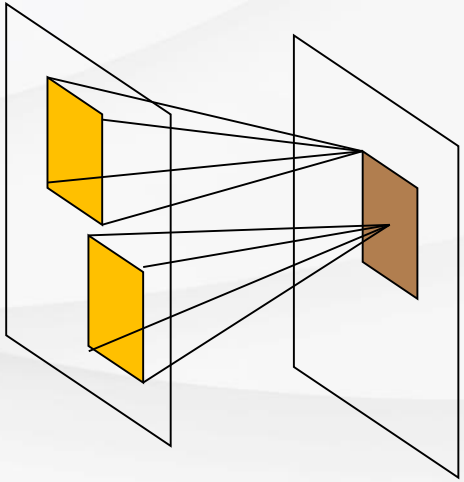



w11 w12 w13
w21 w22 w23
w31 w32 w33
w11 w12 w13
w21 w22 w23
w31 w32 w33



컨볼루션 과정

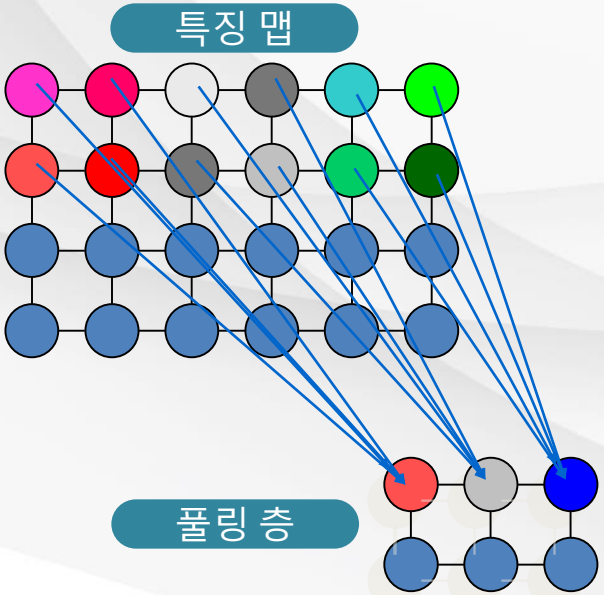
- 한 개의 특징 맵이 컨볼루션 과정을 통해 특징을 탐색할 때 특징 맵의 가중치 값은 변경되지 않음 (가중치 공유)
- 이와 같은 방식으로 특징 맵은 입력 이미지의 다양한 위치에서 동일한 특징을 탐색할 수 있음
- 모델이 갖는 파라미터의 개수를 줄여줌
- 특징 맵이 나타내고자 하는 템플릿과 이미지의 국소 부분이 일치한다면, 특징 맵의 뉴런이 발화

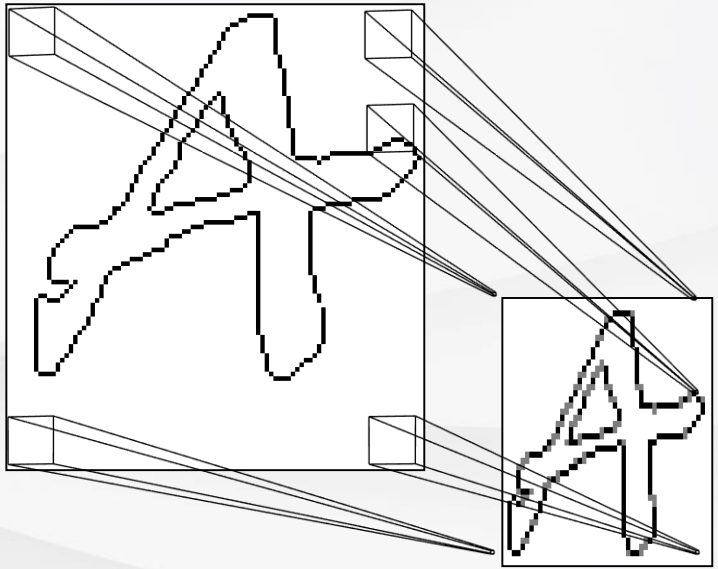


입력 이미지 컨볼루션 층

풀링 과정

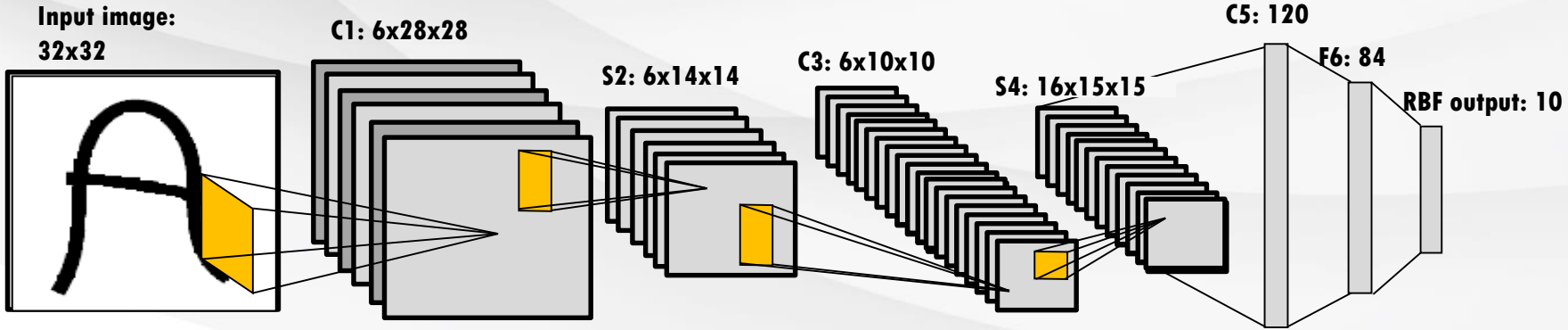
- 물체의 위치와 각도 변화에 잘 대처할 수 있게 해줌
- 각 특징 맵의 해상도를 줄여줌 → 모델의 파라미터의 개수를 줄임
- 최대 / 평균 풀링을 주로 사용



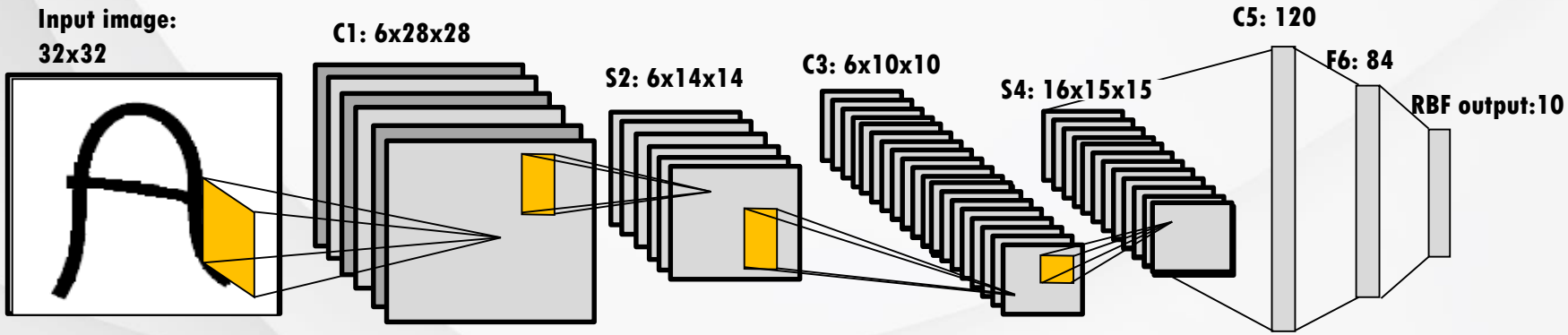


LeNet5

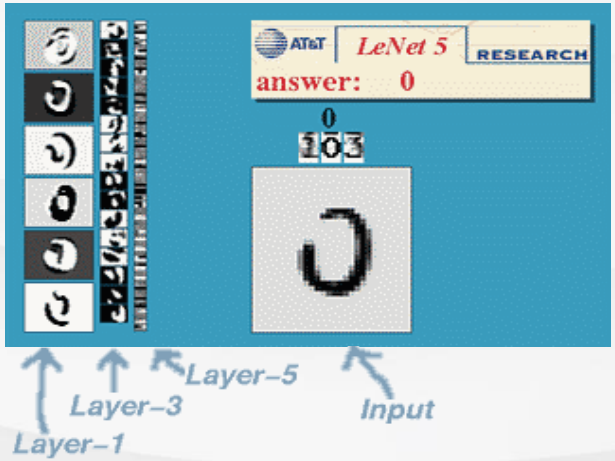
- LeCun에 의해 고안됨
- 입력으로 32x32 픽셀 크기의 이미지를 받음
- C1, C3, C5 : 컨볼루션 층 (5 × 5크기의 피처 맵)
- S2, S4 : 풀링 층 (인자 2에 의한 풀링, 피처의 크기가 절반으로 축소)
- F6 : 단층 신경망 (fully-connected)



LeNet5

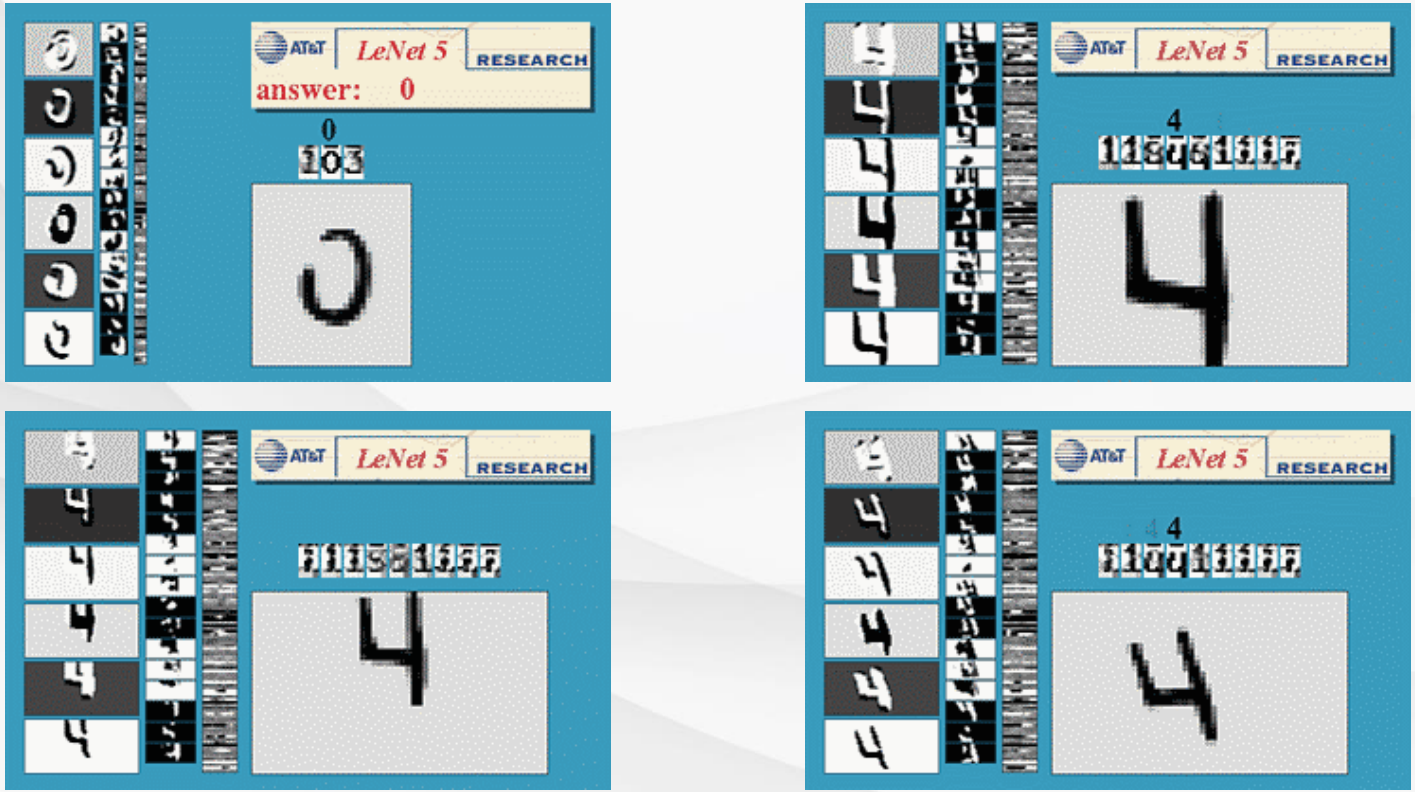


- 약 **187,000**개 뉴런들 사이 연결 존재
- 약 **14,000**개의 모델 파라미터 존재



http://yann.lecun.com/exdb/lenet/

- 장점: 물체의 위치와 각도 변화에 대한 불변성



http://yann.lecun.com/exdb/lenet/

- 장점: 노이즈와 왜곡에 대한 불변성

http://yann.lecun.com/exdb/lenet/

단점

- 메모리 관점에서 일반적인 다층 퍼셉트론보다 더 많은 용량을 차지함 (많은 수의 파라미터)
- 실행 시간 관점에서 컨볼루션 과정이 많은 계산을 필요하고 전체 실행 시간 중 약 2/3의 비중을 차지
- 같은 개수의 파라미터를 갖는 신경망보다 약 3배 가까이 실행 시간이 느림

2. ImageNet: 이미지를 자동 분류하라

2. ImageNet: 이미지를 자동 분류하라

ImageNet Challenge

- 120만 개의 고해상도 이미지
- 1,000개의 서로 다른 클래스
- 50,000 검증 이미지, 150,000개의 테스트 이미지

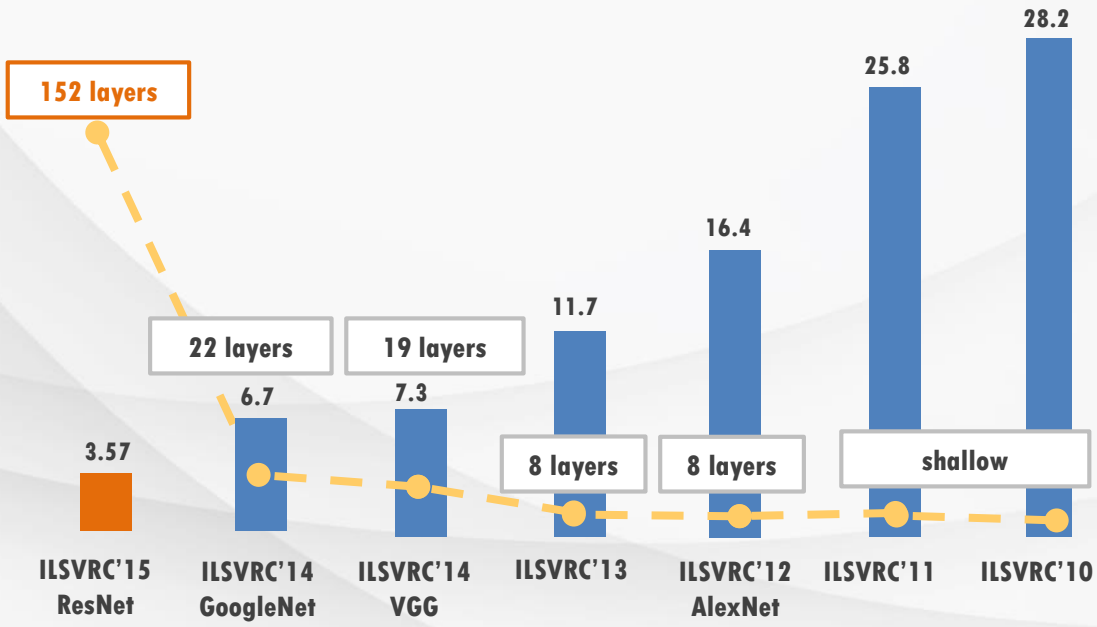
쉬운 클래스

어려운 클래스

2. ImageNet: 이미지를 자동 분류하라

Tacademy

ImageNet 데이터 집합에 대한 매년 성능 향상



이미지 분류의 $Top-5$ 에러

학습정리

학습정리

Tacademy

지금까지 [컨볼루션 신경망]에 대해서 살펴보았습니다.

컨볼루션 신경망의 원리

컨볼루션 층: 입력 이미지 속 다양한 위치에서 동일한 특징들을 탐색,
피처 맵을 입력에 대해 슬라이딩 시킴. 같은 피처 맵은 동일한 가중치 사용
풀링 층: 물체의 위치와 각도 변화에 잘 대처할 수 있게 해 줌, 최대 / 평균 풀링

컨볼루션 신경망과 일반 신경망의 차이

우리 뇌의 시각 피질이 물체를 이해하는 매커니즘 모사,
신경 세포들이 물체의 방향과 장소가 바뀌어도 별문제 없이 인식할 수 있었던 이유인 '컨볼루션' 개념을 신경망 모델에 적용

ImageNet Challenge: 이미지를 자동 분류하라

ImageNet 데이터 집합은 1000개의 이미지 클래스에 해당하는 120만 개 이미지 보유
ImageNet Challenge에서 더 좋은 성능을 보이기 위해 모델의 층이 깊어지고 있음

AlexNet(8층) → VGG(19층) → GoogleNet(22층) → ResNet(152층)

인공지능을 위한 머신러닝 알고리즘

10. 재현 신경망

Tacademy

CONTENTS

학습 목표

- 재현 신경망의 학습 원리를 이해할 수 있다.
- 재현 신경망이 가진 그라디언트 손실 문제를 이해할 수 있다.
- 언어 모델로써 재현 신경망이 어떻게 글을 생성하는지 이해할 수 있다.

1. 재현 신경망의 원리

시계열 데이터 학습 모델

❖ 재현 신경망 **(Recurrent Neural Networks)**은 시계열 데이터를 확률적으로 모델링

- 연속된 데이터들의 집합 $X = (x_1, x_2, \ldots x_T)$
- **X**가 나타날 확률

$$p(X) = p(x_1, x_2, \ldots, x_T) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)\ldots p(x_T \mid x_1, \ldots, x_{T-1}) = \prod_{t=1}^{T} p(x_T \mid x_{<T})$$

- 재현 신경망은 매 시간 단위 **T**마다 다음을 계산

$$p(x_T \mid x_{<T}) = g(h_{T-1})$$

$$h_{T-1} = \phi(x_{T-1}, h_{T-1}) \quad \phi \text{ is non-linear activation function}$$

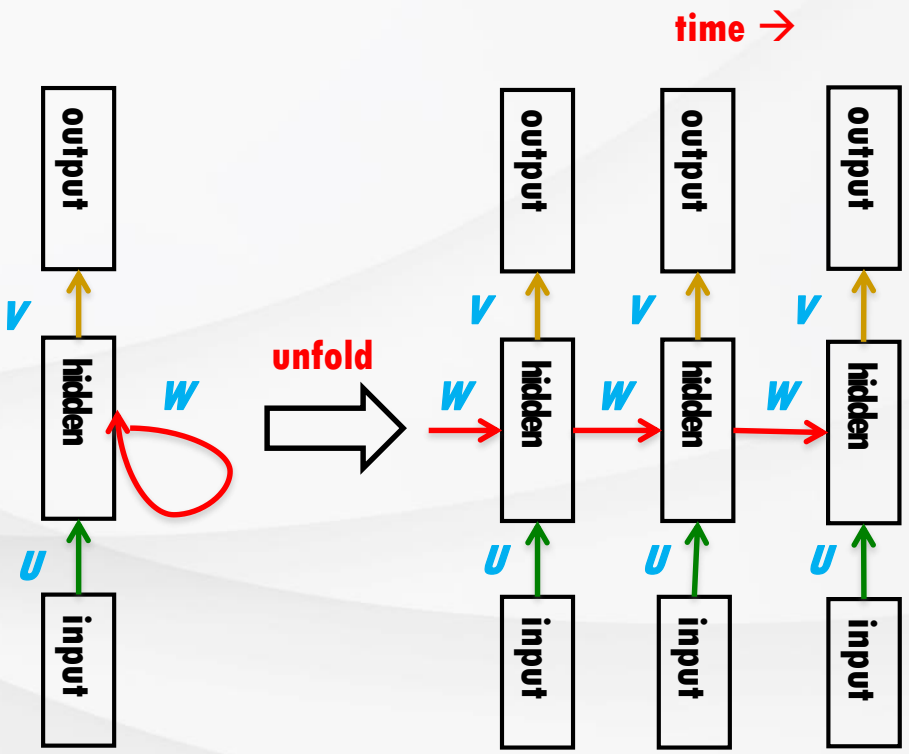
- 은닉층 = 컨텍스트 층 = 히스토리 층
- 역전파를 사용하여 모델 파라미터 학습

학습 가능한 파라미터

U, W, V

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$y_t' = \mathrm{softmax}(Vs_t)$$

- 은닉 유닛들은 연속된 벡터공간에서 오래 전 데이터 정보를 저장
- 많은 수의 뉴런과 시간이 주어진다면, 재현 신경망은 어떠한 시계열 데이터도 학습할 수 있음



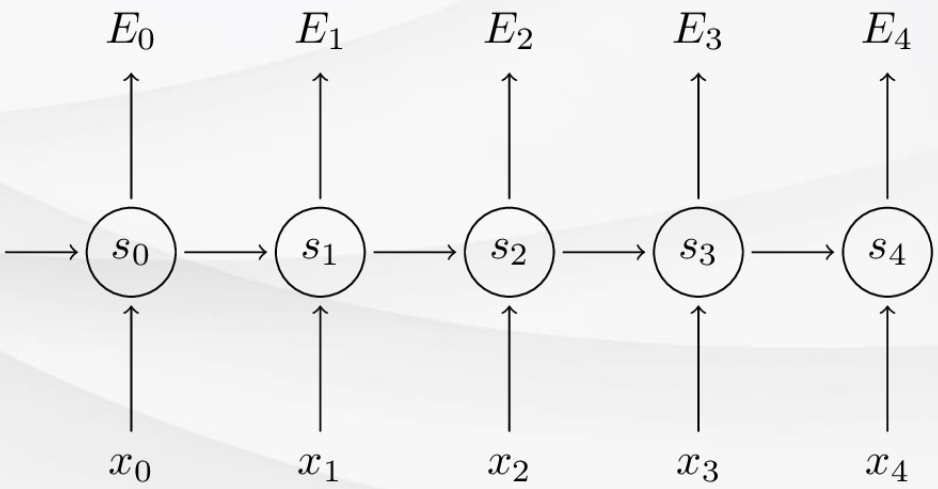
재현 신경망의 손실 함수

- y_t는 클래스 정보를 갖고 있는 이진 벡터
- t 번째 시간에서의 손실 함수는 정답 클래스의 마이너스 로그 확률
- 시계열 데이터 집합 (예> $\{x_0, x_1, x_2, x_3, x_4\}$)의 손실 함수는 각 시간 단위의 손실 함수의 총합

$$E_t(y_t, y_t') = -y_t \log y_t'$$

$$E(y, y') = \Sigma E_t(y_t, y_t')$$

$$= -\Sigma y_t \log y_t'$$



시간에 대한 오류 역전파 과정

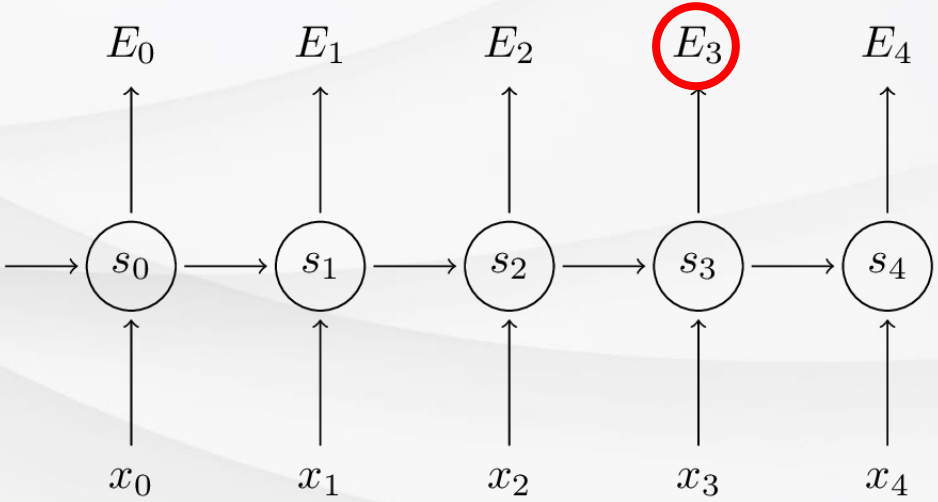
- **Back Propagation Through Time (BPTT)**
- 예> t=3에서 손실 함수에 대한 파라미터 W의 그라디언트 값 계산
- 체인룰 사용
- t=3에서 손실 함수에 영향을 미치는 변수들로 y'_3와 s_3이 존재

$$\frac{\delta E_3}{\delta W} = \frac{\delta E_3}{\delta y'_3}\frac{\delta y'_3}{\delta s_3}\frac{\delta s_3}{\delta w}$$

Hidden weight parameter

Output vector

Hidden vector



$s_t = \tanh(Ux_t + Ws_{t-1})$
$y_t' = \mathrm{softmax}(Vs_t)$

분류 과정

- 재현 신경망에서 t=3일 때, 은닉 유닛 s_3의 값 계산

$$s_3 = \tanh(Ux_t + Ws_2)$$

- s_3은 s_2, 에 의해 영향을 받고, s_2는 다시 W와 s_1에 영향을 받음
  →s_3에 대한 s_2, W, s_1의 변화량을 계산해야 함
- 하지만, s_2는 다시 s_1과 W에 의해 영향을 받으므로 상수로 볼 수 없음

$$\frac{\delta E_3}{\delta W} = \frac{E_3}{\delta y'_3}\frac{\delta y'_3}{\delta s_3}\frac{\delta s_3}{\delta W} + \frac{\delta E_3}{\delta y'_3}\frac{\delta y'_3}{\delta s_3}\frac{\delta s_3}{\delta s_2}\frac{\delta s_2}{\delta W} + \frac{\delta E_3}{\delta y'_3}\frac{\delta y'_3}{\delta s_3}\frac{\delta s_3}{\delta s_1}\frac{\delta s_1}{\delta W} + \frac{\delta E_3}{\delta y'_3}\frac{\delta y'_3}{\delta s_3}\frac{\delta s_3}{\delta s_0}\frac{\delta s_0}{\delta W}$$

$$\frac{\delta s_3}{\delta s_2}\frac{\delta s_2}{\delta s_1}\frac{\delta s_1}{\delta s_0}$$

$$\frac{\delta E_3}{\delta W} = \frac{\delta E_3}{\delta y'_3}\frac{\delta y'_3}{\delta s_3}\frac{\delta s_3}{\delta w}$$

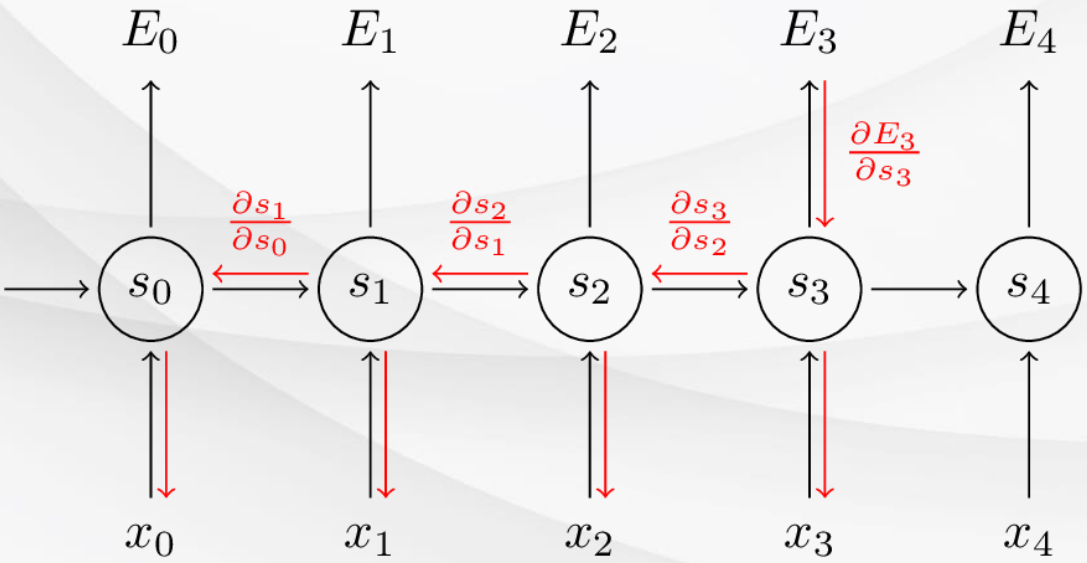
$$= \sum_{k=0}^{3} \frac{\delta E_3}{\delta y'_3}\frac{\delta y'_3}{\delta s_3}\frac{\delta s_3}{\delta s_k}\frac{\delta s_k}{\delta W}$$

$$= \sum_{k=0}^{3} \frac{\delta E_3}{\delta y'_3}\frac{\delta y'_3}{\delta s_3}\prod_{j=k+1}^{3}\frac{\delta s_j}{\delta s_{j-1}}\frac{\delta s_k}{\delta W}$$

시간에 대한 오류 역전파 과정

$$\frac{\delta E_3}{\delta W} = \sum_{k=0}^{3} \frac{\delta E_3}{\delta y'_3} \frac{\delta y'_3}{\delta s_3} \prod_{j=k+1}^{3} \frac{\delta s_j}{\delta s_{j-1}} \frac{\delta s_k}{\delta W}$$

- 그라디언트 계산 과정에서 다수의 곱셈이 필요함
- 시간에 따른 오류 역전파 과정이 생기는 이유는
  같은 파라미터 **w**를 매시간 단위마다 반복해서 사용하기 때문



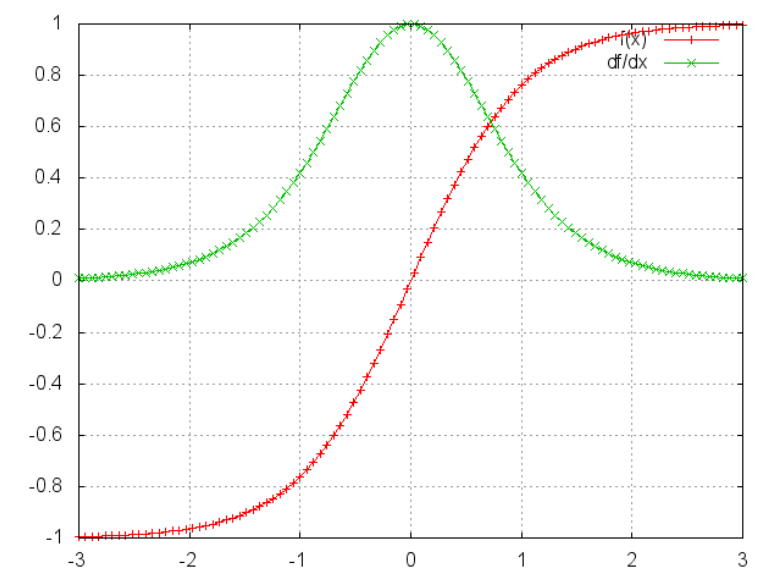
그라디언트 손실 발생

$$\frac{\delta s_3}{\delta s_2}\frac{\delta s_2}{\delta s_1}\frac{\delta s_1}{\delta s_0} = Ws'(z_3)Ws'(z_2)Ws'(z_1) \quad (* \; zt = \tanh(Uxt + Wst_{-1})$$

활성함수의 미분 값

- **tanh: 0~1**
- 시그모이드 : **0~0.25**

- 활성함수의 미분 값을 여러 번 곱해줄 경우 그라디언트의 값이 **0**으로 수렴할 수 있음
- 시간에 대한 길이가 깊어질수록 학습이 잘 되지 않음



그라디언트 폭발 발생

- 만약 파라미터의 값이 매우 클 경우 (예> **100** 이상)
- $Ws'(z_3)Ws'(z_2)Ws'(z_1)$식에서 매우 큰 값을 계산하게 됨
- 그라디언트 값이 매우 커져서 **NaN**이 되거나 프로그램 종료 발생
- 해결책: 그라디언트 클리핑 (**Gradient Clipping**)
- 그라디언트 손실이 더욱 문제가 됨
  - 그라디언트 폭발은 프로그램 종료에 의해 알아채기 쉽고 해결책이 분명하지만, 그라디언트 손실은 발생 사실을 알기 어렵고 해결책이 불분명

2. GRU, LSTM: 재현 신경망의 한계를 넘어

GRU(Gated Recurrent Units)

일반적인 재현 신경망은 매 시간마다 은닉층을 다음과 같이 직접 계산함

$$h_t = f(Ux_t + Wh_{t-1})$$

GRU는 먼저 현재 입력 데이터와 은닉층을 기반으로 업데이트 게이트(Update Gate)를 계산

$$z_t = \sigma(U^{(z)}x_t + W^{(z)}h_{t-1})$$

리셋 게이트(Reset Gate)도 같은 식이지만 다른 파라미터를 사용하여 계산

$$r_t = \sigma(U^{(r)}x_t + W^{(r)}h_{t-1})$$

잠재적 은닉 유닛 계산

업데이트 게이트

$$z_t = \sigma(U^{(z)}x_t + W^{(z)}h_{t-1})$$

리셋 게이트

$$r_t = \sigma(U^{(r)}x_t + W^{(r)}h_{t-1})$$

잠재적 은닉 유닛 계산

만약 리셋 게이트가 **0**으로 설정되면, 이전 단계까지 계산한 은닉 유닛을 고려하지 않음,
그리고 현재 시간 단계에서만 계산한 정보만 반영

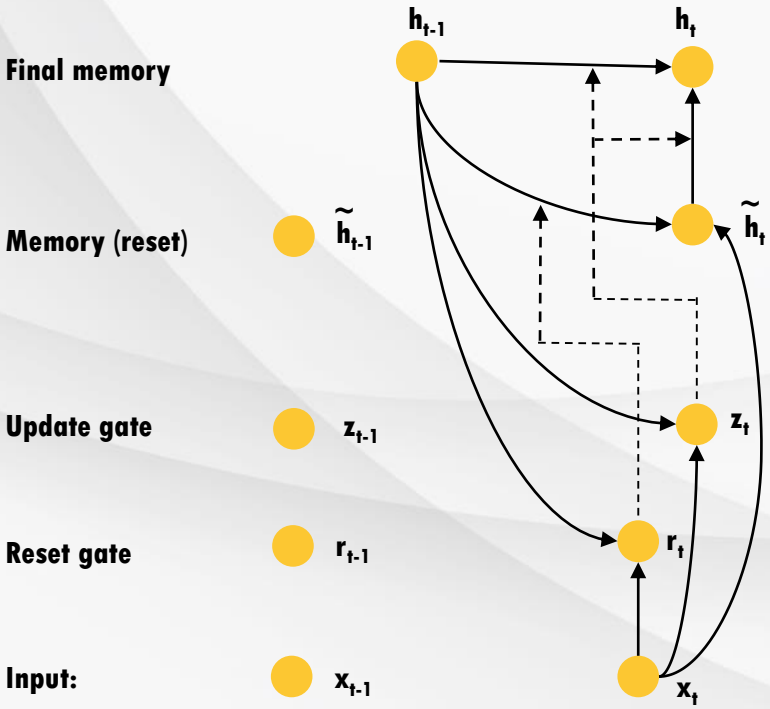
"잠재적 은닉 유닛"

$$h'_t = \tanh(Ux_t + r_t \circ Wh_{t-1})$$

최종 사용할 은닉 유닛은 현재 시간에 계산한 잠재적 은닉 유닛과
이전 시간 단계까지 계산한 은닉 유닛을 업데이트 게이트를 사용하여 조합

$$h_t = z_t \circ h_{t-1} + (1-z_t) \circ h'_t$$

GRU의 은닉 유닛 계산 과정



$$z_t = \sigma(U^{(z)}x_t + W^{(z)}h_{t-1})$$

$$r_t = \sigma(U^{(r)}x_t + W^{(r)}h_{t-1})$$

$$h'_t = \tanh(Ux_t + r_t \circ Wh_{t-1})$$

$$h_t = z_t \circ h_{t-1} + (1-z_t) \circ h'_t$$

GRU의 은닉 유닛 계산 과정

- 만약 리셋 게이트(r_t)를 **0**으로 설정하면 잠재적 은닉 유닛 계산 시 이전 시간 단계의 정보를 고려하지 않음
- 업데이트 게이트(z_t)는 최종 은닉 유닛을 계산할 때, 이전 시간의 정보를 얼마큼 반영할지 결정
  - 만약 z_t가 **1**에 가까울 경우, 이전 시간의 정보를 거의 그대로 복사하게 됨. (그라디언트 손실이 적어짐)
- 만약 리셋 게이트를 모두 **1**로 설정하고, 업데이트 게이트를 **0**으로 설정하면 일반적인 재현 신경망이 될 수 있음

$$z_t = \sigma(U^{(z)}x_t + W^{(z)}h_{t-1})$$

$$r_t = \sigma(U^{(r)}x_t + W^{(r)}h_{t-1})$$

$$h'_t = \tanh(Ux_t + r_t \circ Wh_{t-1})$$

$$h_t = z_t \circ h_{t-1} + (1-z_t) \circ h'_t$$

LSTM (Long Short-Term Memory)

- LSTM은 GRU보다 게이트가 하나 더 많음(출력 게이트 추가)
- 최종 은닉 유닛을 계산하는 과정에서 한 단계 더 추가됨
  - 두 가지 보조 메모리가 존재 (잠재 은닉 유닛 g_t, 셀 메모리 c_t)
- 데이터의 개수가 더 많을 경우 LSTM을 사용하면 더 좋은 성능을 나타낼 수 있음

GRU

$$z_t = \sigma(U^{(z)}x_t + W^{(z)}h_{t-1})$$
$$r_t = \sigma(U^{(r)}x_t + W^{(r)}h_{t-1})$$
$$h'_t = \tanh(Ux_t + r_t \circ Wh_{t-1})$$
$$h_t = z_t \circ h_{t-1} + (1-z_t) \circ h'_t$$

LSTM

$$i_t = \sigma(U^{(i)}x_t + W^{(i)}h_{t-1})$$
$$f_t = \sigma(U^{(f)}x_t + W^{(f)}h_{t-1})$$
$$o_t = \sigma(U^{(o)}x_t + W^{(o)}h_{t-1})$$
$$g_t = \tanh(U^{(g)}x_t + W^{(g)}h_{t-1})$$
$$c_t = c_{t-1} \circ f_t + g_t \circ i_t$$
$$s_t = \tanh(c_t) \circ o_t$$

Candidate hidden state

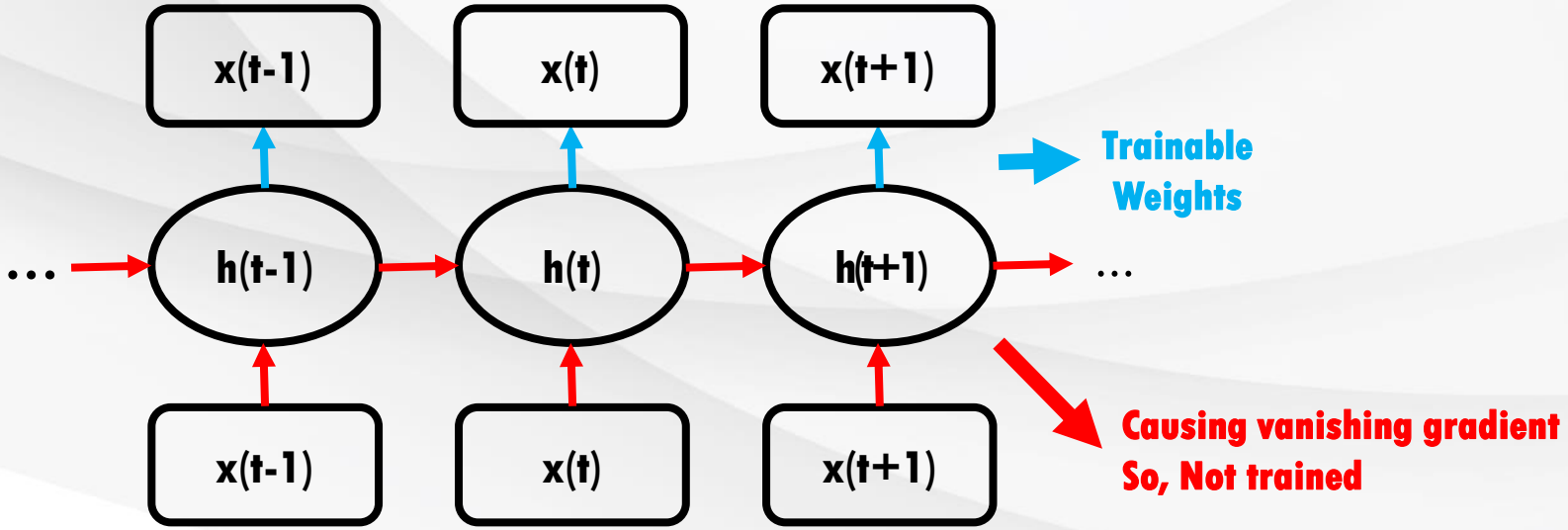
Input gate

Output gate

Forget gate

ESN (Echo State Networks)

- 파라미터 u, w는 학습하지 않음
- 많은 수의 은닉 유닛 개수가 필요하고 유닛들 사이는 매우 섬겨야 함 (Sparse)
- 길이가 긴 시계열 데이터를 효율적으로 학습할 수 있음



3. 어떻게 기계가 글을 생성할 수 있을까

언어 모델링 문제

- 문제의 목표는 문장 **W**가 나타날 확률 **P(W)**를 계산하는 것
- **P(W)**는 체인룰을 사용하여 단어들의 조건부 확률들의 곱으로 나타낼 수 있음

$$P(W) = \mathrm{P}(\mathrm{w}(1), \mathrm{w}(2), \mathrm{w}(3), \ldots, \mathrm{w}(\mathrm{M})) = \prod_{k=1}^{M+1} P(w(k)|w(1), \ldots w(k-1))$$

- 예를 들어, 언어 모델은 다음 문장이 나타날 확률을 계산할 수 있음

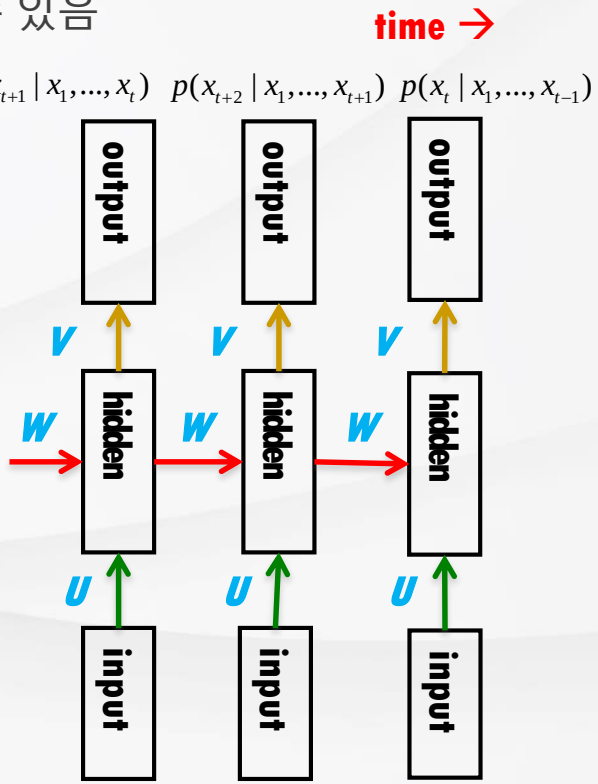
W = He went to buy some chocolate
P(W) = P(chocolate | He went to buy some) * P(He went to buy some)
= P(chocolate | He went to buy some) * P(some | He went to buy) * P(He went to buy)
= ...

- 이전 단어들의 집합이 컨텍스트로 주어졌을 때,
다음 단어가 등장할 확률의 연속적인 곱셈으로 나타남

언어 모델링 문제

- 재현 신경망을 사용하면 매시간 단위마다 단어를 생성할 수 있음
- t 번째 시간 단위에서 생성된 단어는 t+1번째 시간 단위의 입력이 됨
- **He went to buy some chocolate.**
- 매시간 단위마다 은닉 유닛은 이전 시간까지 생성한 단어들의 정보를 가지고 있음
- 길이가 매우 긴 문장의 경우, 의미 있는 문장을 생성하기 어려움

예> **The girl whom I met yesterday is very pretty**



생성된 글의 예

PANDARUS:
Alas, I think he shall be come approached and the day When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death, I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul, Breaking and strongly should be buried,
when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and my fair nues begun out of the fact, to be conveyed, Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

영어 - 셰익스피어의 글을 대신 생성

학습정리

지금까지 [재현 신경망]에 대해서 살펴보았습니다.

재현 신경망의 원리

재현 신경망은 시계열 데이터를 확률적으로 모델링
학습 가능 파라미터: U, W, V
은닉 유닛의 계산: $s_t = \tanh(Ux_t + Ws_{t-1})$ / **출력 계산:** $y_t' = \mathrm{softmax}(Vs_t)$

그라디언트 손실

활성함수의 미분값을 여러 번 곱셈하는 과정에서 발생 $Ws'(z_3)Ws'(z_2)Ws'(z_1)$
활성함수의 미분값: tanh(0~1), 시그모이드(0~0.25)
프로그래머가 알아차리기 어려워서 그라디언트 폭발보다 더 문제가 됨

언어모델로써 재현 신경망

문장 W의 확률인 P(W)는 체인룰을 사용하여 단어들의 조건부 확률들의 곱으로 나타낼 수 있음

$$P(W) = \mathrm{P}(\mathrm{w}(1), \mathrm{w}(2), \mathrm{w}(3), \ldots, \mathrm{w}(\mathrm{M})) = \prod_{k=1}^{M+1} P(w(k)|w(1), \ldots w(k-1))$$

재현 신경망의 t번째 시간 단위에서 생성된 단어는 t+1번째 시간 단위의 입력이 됨

인공지능을 위한 머신러닝 알고리즘

11. 메모리 네트워크

Tacademy

CONTENTS

학습 목표

- 메모리 네트워크의 특징과 작동 원리를 이해할 수 있다.
- 메모리 네트워크와 종단 메모리 네트워크의 차이점을 이해할 수 있다.
- bAbI task를 사용한 메모리 네트워크의 언어학습 능력을 이해할 수 있다.

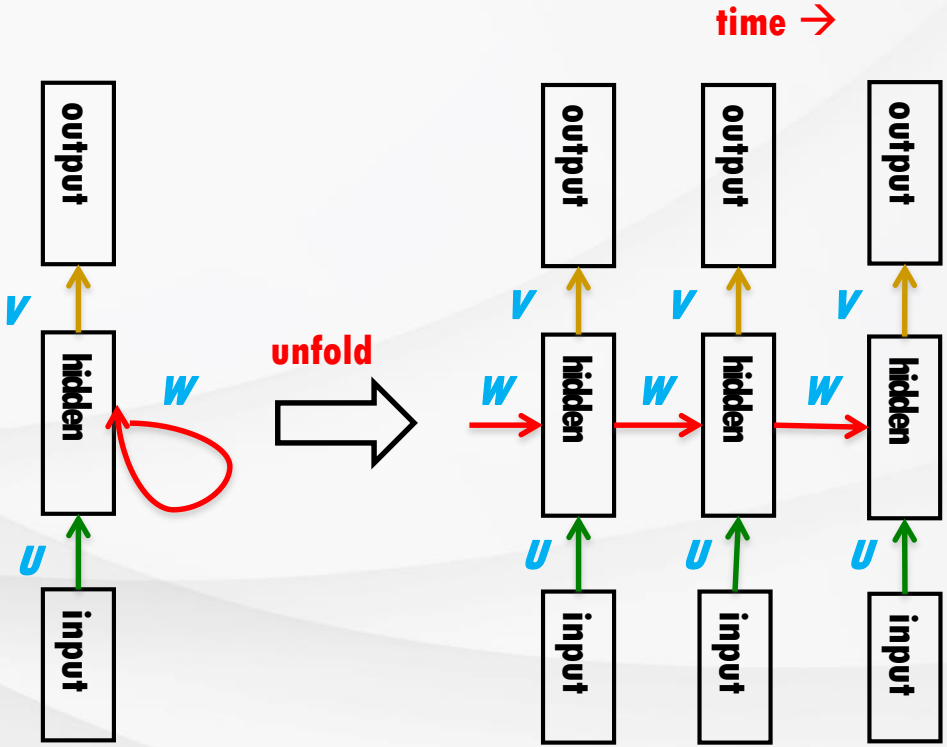
1. 메모리 네트워크

메모리가 필요한 재현 신경망

U, W, V

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$y_t' = \mathrm{softmax}(Vs_t)$$

- 은닉 유닛들은 연속된 벡터공간에서 오래전 데이터 정보를 저장
- 길이가 매우 긴 입력 데이터가 주어진다면 은닉 유닛 벡터는 제한된 저장 공간에 많은 입력 정보를 저장해야 함



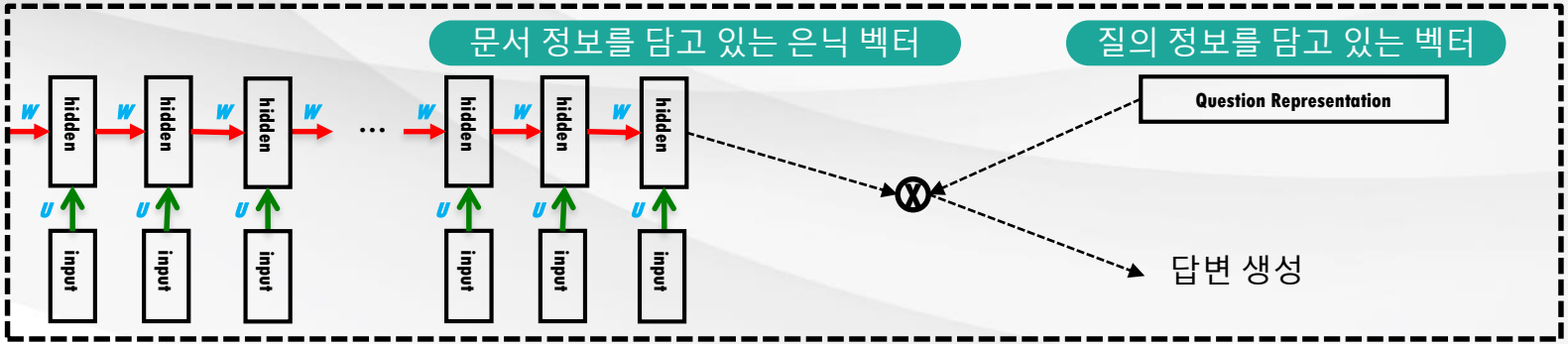
예시 : 재현 신경망으로 QA 모델 만들기

2017년은 앞으로의 산업계를 완전히 바꿔 놓을 새로운 기술이 본격 자리 잡는 한 해가 될 것이다.
앞으로 업계 변화를 주도할 기술은 무엇일까.
…
두 번째는 머신러닝(기계학습)을 통한 창조성 강한 디자인 등장이다. 머신러닝은 학습을 통해 사물을 구별하고 이를 바탕으로 새로운 것을 창조하는 단계에 이르렀다. 최근 `디자인 그래프`라 불리는 프로젝트로 구현됐다.
…

QA 모델

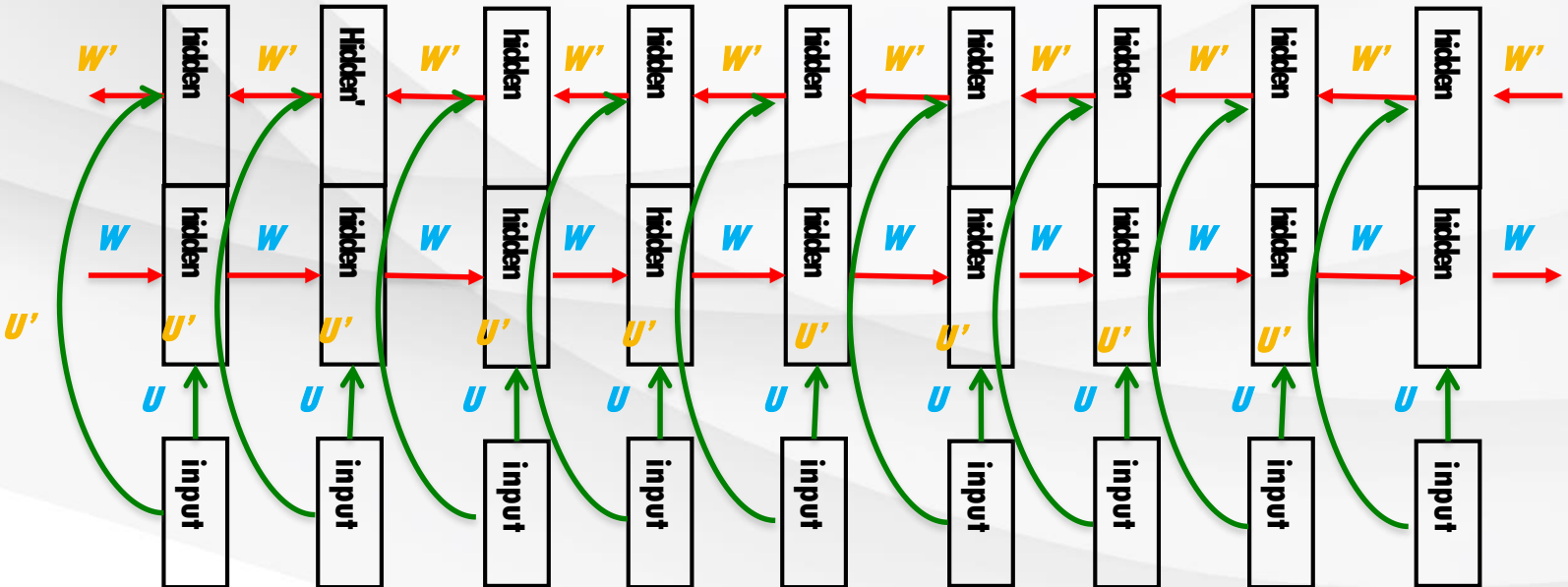
Q. 머신러닝으로 새로운 디자인을 창조하고자 하는 프로젝트 이름은 무엇인가?

A. 디자인 그래프



해결책1: 양방향 재현 신경망 (bi-directional RNN)

- 앞 방향 재현 신경망: 원본 문장의 가장 앞의 단어부터 인코더 재현 신경망의 입력으로 주어짐
- 뒤 방향 재현 신경망: 원본 문장의 가장 뒤의 단어부터 인코더 재현 신경망의 입력으로 주어짐
- 인코더 재현 신경망의 은닉 유닛은 앞 방향 / 뒤 방향 재현 신경망의 은닉 유닛을 단순 이어붙임 (Concatenation)



해결책2: 히든 유닛 벡터를 저장

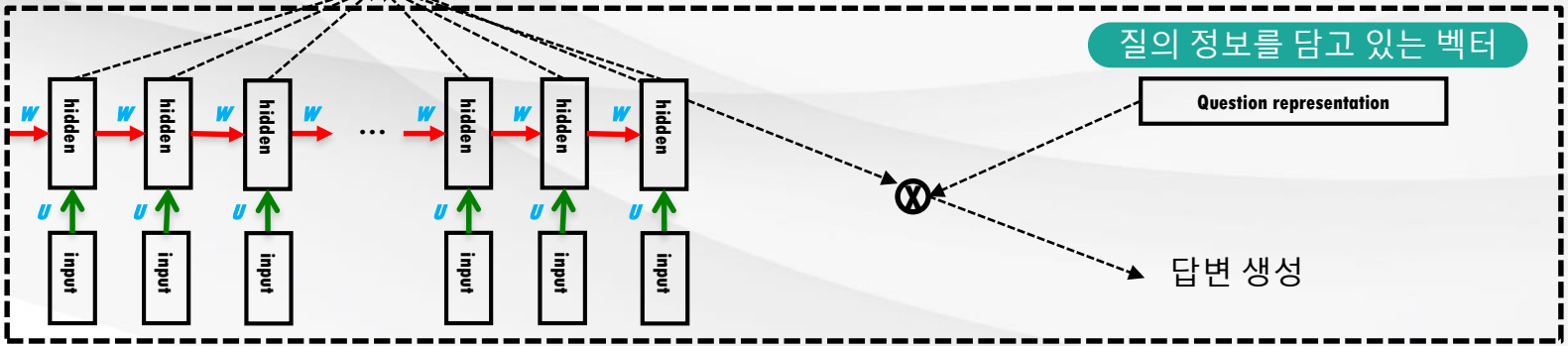
2017년은 앞으로의 산업계를 완전히 바꿔 놓을 새로운 기술이 본격 자리 잡는 한 해가 될 것이다.
앞으로 업계 변화를 주도할 기술은 무엇일까.
…
두 번째는 머신러닝(기계학습)을 통한 창조성 강한 디자인 등장이다. 머신러닝은 학습을 통해 사물을 구별하고 이를 바탕으로 새로운 것을 창조하는 단계에 이르렀다. 최근 `디자인 그래프`라 불리는 프로젝트로 구현됐다.
…

QA 모델

Q. 머신러닝으로 새로운 디자인을 창조하고자 하는 프로젝트 이름은 무엇인가?

A. 디자인 그래프

RAM 저장



메모리 네트워크

❖ 질의 응답 모듈

◉ 스토리 (예> 문장) 선택 모듈

- 질의 **q**에 대해 가장 적절한 답변을 포함하고 있는 스토리 (문장) **s***를 탐색
  >> 적절성 평가를 통해 가장 높은 점수를 받은 스토리가 선택

$$s^* = \arg\max_{s_i} G(q, s_i)$$

- 스토리 선택 모듈의 출력은 질의 **q**와 선택된 스토리 **s***의 결합
  Ex> 머신러닝으로 새로운 디자인을 창조하고자 하는 프로젝트 이름은 무엇인가**?**
  최근 `디자인 그래프`라 불리는 프로젝트로 구현됐다.

◉ 답변 선택 모듈

- 가능한 답변의 집합 $A = \{a_r\}_{1 \ldots k}$ 중 가장 적합한 답변 **a***를 선택

$$a^* = \arg\max_{a_r} H(s_a, a_r)$$

메모리 네트워크

❖ 질의 응답 모듈

◉ 적절성 평가 함수 (Scoring Function)

- $G(x,y) \in [0,1]$ (H 함수도 동일)
- $G(x,y) = \phi_x U_G^T U_G \phi_y(y)$
- $H(x,y) = \phi_x U_H^T U_H \phi_y(y)$
- U는 임베딩 행렬

메모리 네트워크의 학습: Triplet Loss

- 학습되어야 하는 함수 : **G, H**

$$s^* = \arg\max_{s_i} G(q, s_i) \quad a^* = \arg\max_{a_r} H(s_a, a_r)$$

$$G(x, y) = \phi_x U_G{}^{\mathrm{T}} U_G \phi_y(y)$$
$$H(x, y) = \phi_x U_H{}^{\mathrm{T}} U_H \phi_y(y)$$

- **G, H** 함수: 긍정 예(**s*** 또는 **a***)와 부정 예(**s_i** 또는 **a_i**) 의 점수 차이를 γ이상 차이가 나게 만드는 것이 목표
- 손실값 (에러값): $\max(0, \gamma - G(q, s*) + G(q, si))$
  - 긍정 예와 질의의 매치 점수가 부정 예와 질의의 매치 점수보다 γ이상 높을 경우 손실값은 **0**이 됨

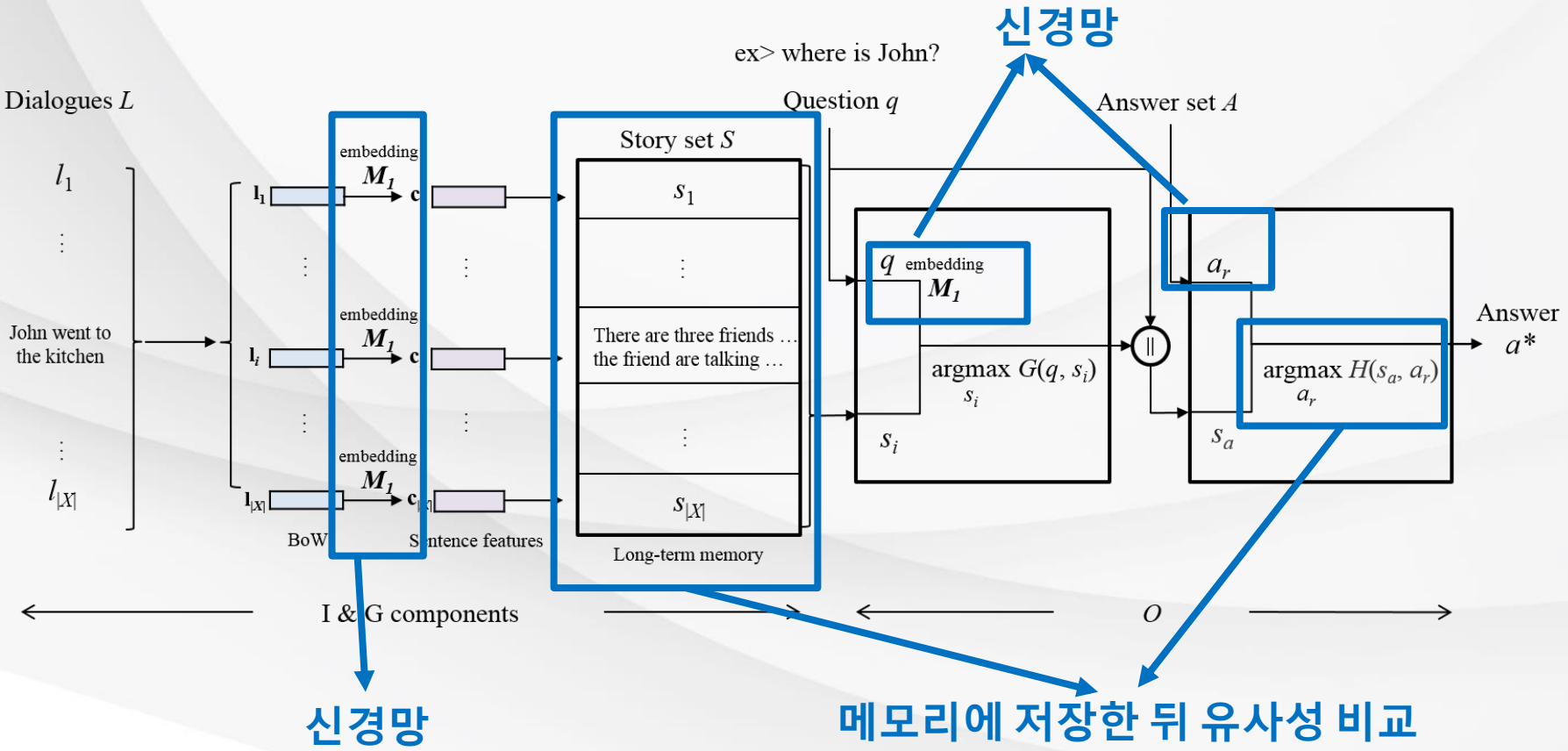
메모리 네트워크의 학습: **Triplet Loss**

- 훈련 데이터 전체에 대한 손실 값 :

$$\sum_{s_i} \max(0, \gamma - G(q, s^*) + G(q, si)) \; + \sum_{a_i} \max(0, \gamma - H([q; s^*], a^*) + H([q; s^*], ai))$$

- 경사 하강법을 사용하여 파라미터 U_G와 U_H를 학습

메모리 네트워크의 의의

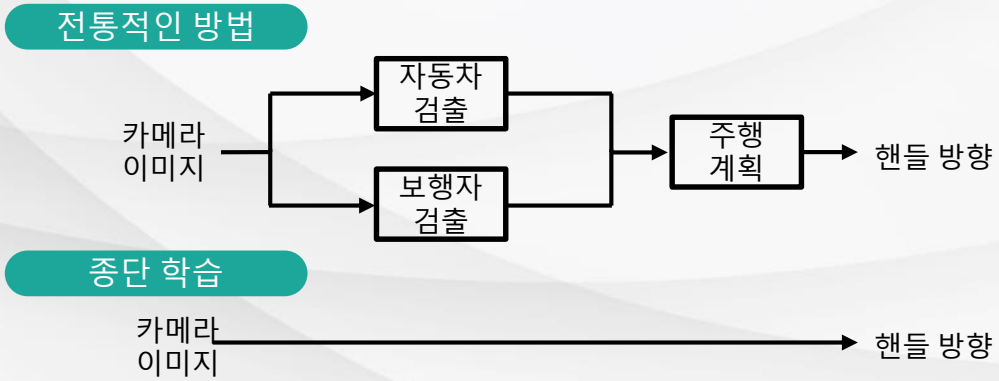

신경망
ex> where is John?
Dialogues L
Question q
Answer set A
l_1
John went to the kitchen
$l_{|X|}$
embedding M_1
$\mathbf{l}_1$
$\mathbf{l}_i$
$\mathbf{l}_{|X|}$
$\mathbf{c}$
$\mathbf{c}_{|X|}$
BoW
Sentence features
Story set S
s_1
There are three friends … the friend are talking …
$s_{|X|}$
Long-term memory
q embedding M_1
argmax $G(q, s_i)$ s_i
s_i
a_r
argmax $H(s_a, a_r)$ a_r
s_a
Answer $a*$
I & G components
O
신경망
메모리에 저장한 뒤 유사성 비교

2. 종단 메모리 네트워크

종단 학습 모델 (End-to-End Learning Model)

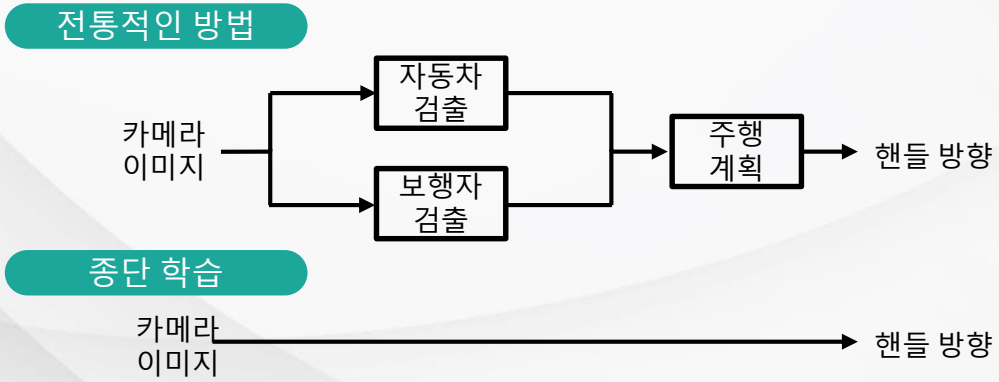
❖ 종단 학습 모델이란

주어진 문제를 풀기 위해서 유용한 내부적 특징 표현들을 스스로 학습하고
다양한 중간 단계의 과정들을 학습 알고리즘 하나로 통합하여
전체적인 작업을 종단으로 수행하는 모델



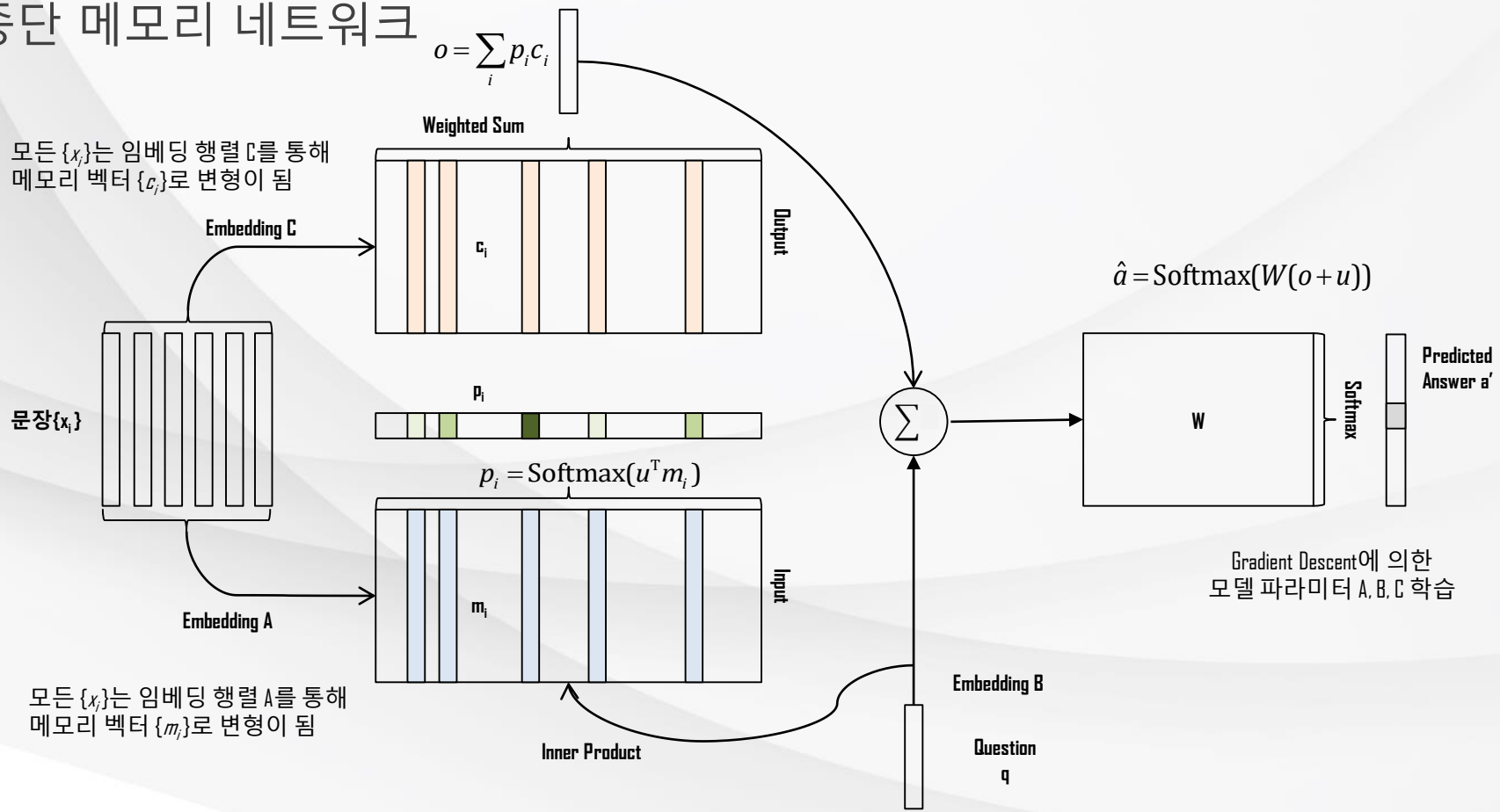
종단 학습 모델 (End-to-End Learning Model)

❖ 종단 학습 모델이란



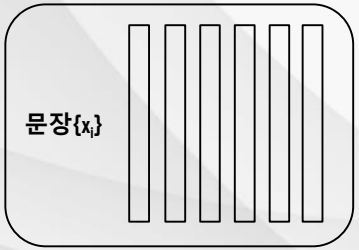
- 기존 메모리 네트워크의 스토리 선택 모듈 (함수 G)를 학습시키기 위해서 질의에 대한 적절한 스토리 정보를 사람이 추가로 작성해야 함 (too expensive!)
- 종단 학습 메모리 네트워크는(End-to-End Memory Networks) 종단 학습 방식을 사용하여 훈련 과정에서 추가적인 감독(Supervision)이 필요하지 않음

종단 메모리 네트워크

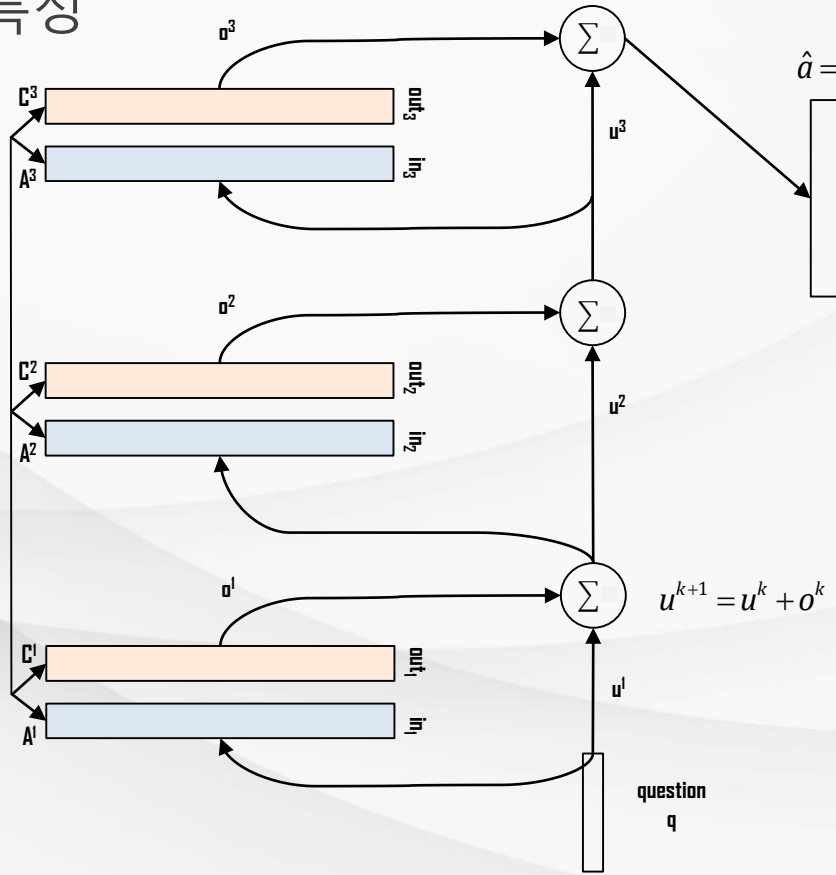

$o = \sum_i p_i c_i$
Weighted Sum
모든 $\{x_i\}$는 임베딩 행렬 C를 통해
메모리 벡터 $\{c_i\}$로 변형이 됨
Embedding C
c_i
Output
문장$\{x_i\}$
p_i
$p_i = \mathrm{Softmax}(u^T m_i)$
m_i
Input
Embedding A
모든 $\{x_i\}$는 임베딩 행렬 A를 통해
메모리 벡터 $\{m_i\}$로 변형이 됨
Inner Product
Embedding B
Question
q
$\hat{a} = \mathrm{Softmax}(W(o+u))$
W
Softmax
Predicted
Answer a'
Gradient Descent에 의한
모델 파라미터 A, B, C 학습

재현 네트워크 같은 특성

질의와 관련된 스토리가
여러 개일 경우,
K번 메모리 접근 연산

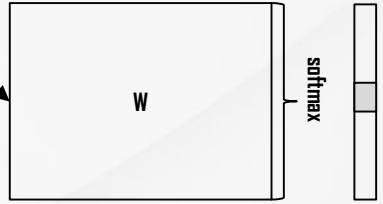


임베딩 행렬 A와 C를 모든 층에 걸쳐 파라미터를 공유함
($A_1=A_2=\ldots=A_K$, $C_1=C_2=\ldots C_K$)



마지막 메모리 층의 출력과 입력을 조합

$$\hat{a} = \mathrm{Softmax}(W(o^3 + u^3))$$



Predicted
Answer a'

3. 나와 질의응답을 하는 메모리 네트워크

bAbI Task

- 2015년 Facebook AI 연구소에서 제작
- 모델의 언어 이해 및 추론 능력을 테스트
  - 예전에는 언어 이해/추론 문제를 머신러닝으로 접근하기보다 지식 공학의 형태로 접근하였으나, 신경망을 활용한 해결 방법들이 제시되고 있음

1 John went to the kitchen.
2 John picked up the milk.
3 John travelled to the office.
4 John left the milk there.
5 John went to the bathroom.
Where is the milk now? A: office 2 3 4
Where is John? A: bathroom 5

- 20개의 테스트로 구성되며, 각 테스트는 별도의 고유한 추론 능력을 요구함
- 1,000개의 훈련 데이터 쌍 / 1,000개의 테스트 데이터 쌍으로 구성
- Supporting fact라고 불리는 답에 대한 근거 문장 정보가 표시되어 있지만, 이러한 감독 정보를 최소한으로 사용하는 모델이 선호됨

bAbI Task 종류

- **Supporting fact** 한 개
- **Supporting fact** 두 개
- **Supporting fact** 세 개
- 두 개 객체의 관계
- 세 개 객체의 관계
- 예/아니오
- 개수 세기
- 객체 집합
- 부정 표현
- 불확실한 지식
- 공동 참조 (**coreference**)
- 시간 추론
- 연역 추론
- 귀납 추론
- 장소 추론
- 크기 추론
- 길 찾기
- 의도 추론

Daniel picked up the football.
Daniel dropped the football.
Daniel got the milk.
Daniel took the apple.
How many objects is Daniel holding? A: two

개수 세기 문제의 예

Sheep are afraid of wolves.
Cats are afraid of dogs.
Mice are afraid of cats.
Gertrude is a sheep.
What is Gertrude afraid of? A:wolves

연역 추론 문제의 예

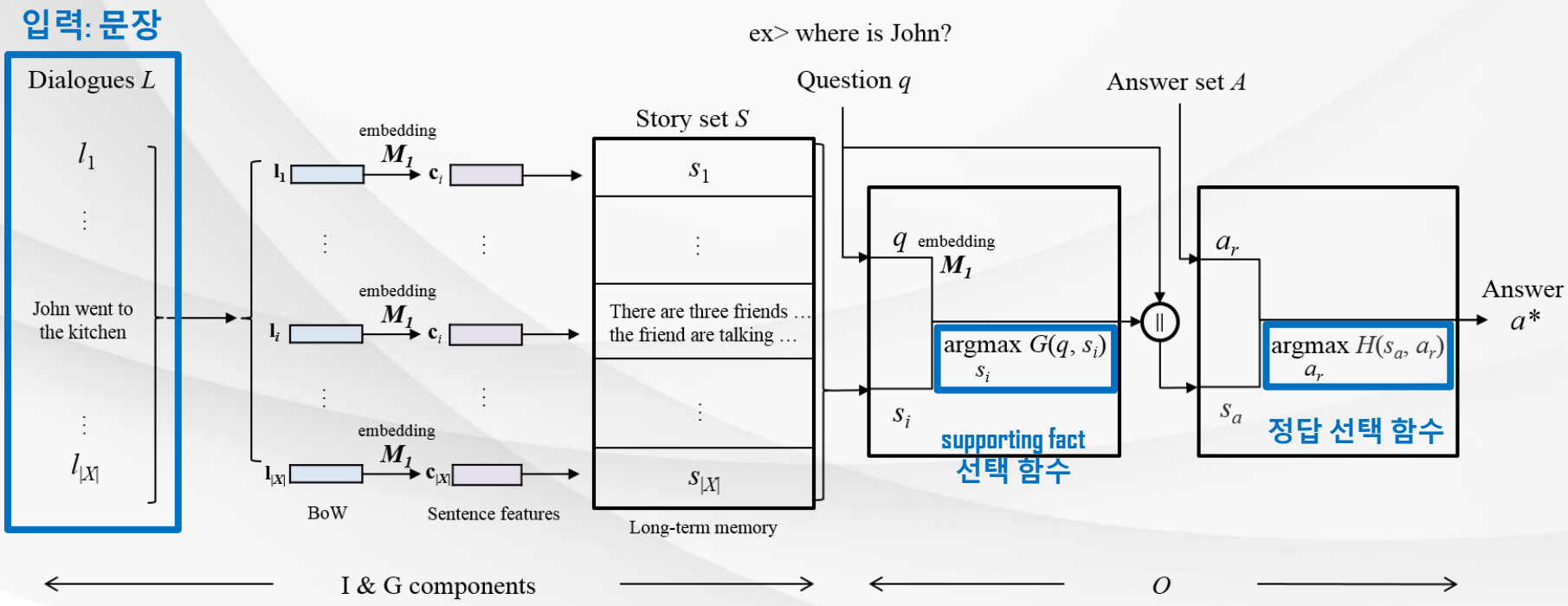
Lily is a swan.
Lily is white.
Bernhard is green.
Greg is a swan.
What color is Greg? A:white

귀납 추론 문제의 예

The kitchen is north of the hallway.
The bathroom is west of the bedroom.
The den is east of the hallway.
The office is south of the bedroom.
How do you go from den to kitchen? A: west, north
How do you go from office to bathroom? A: north, west

길 찾기 문제의 예

메모리 네트워크를 통한 질의 응답



bAbI Task에 대한 메모리 네트워크의 성능

	ngram	LSTM	SVM	MemNN	성공여부 (p>95)	멀티태스크 학습
1. Supporting fact 한 개	36	50	99	100	성공	100
2. Supporting fact 두 개	2	20	74	100	성공	100
3. Supporting fact 세 개	7	20	17	100	성공	98
4, 두 개 객체의 관계	50	61	98	100	성공	80
5. 세 개 객체의 관계	20	70	83	98	성공	99
6. 예/아니오	49	48	99	100	성공	100
7. 개수 세기	52	49	69	85	실패	86
8. 객체 집합	40	45	70	91	실패	93
9. 부정 표현	62	64	100	100	성공	100
10. 불확실한 지식	45	44	99	98	성공	98
11. 공동 참조 (Coreference)	29	72	100	100	성공	100
12. Conjunction	9	74	96	100	성공	100
13. 복합 참조	26	94	99	100	성공	100
14. 시간 추론	19	27	99	99	성공	99
15. 연역 추론	20	21	96	100	성공	100
16. 귀납 추론	43	23	24	100	성공	94
17. 장소 추론	46	51	61	65	실패	72
18. 크기 추론	52	52	62	95	성공	93
19. 길 찾기	0	8	49	36	실패	19
20. 의도 추론	76	91	95	100	성공	100
평균	34	49	79	93		92

질의 응답 + 대화 문제

1. 영화에 대한 단순 질의 응답

What movies are about open source?
Revolution OS
Ruggero Raimondi appears in which movies?
Carmen
Can you name a film directed by Stuart Ortiz? Grave Encounters

2. 영화 추천

Some movies I like are Heat, Kids, Fight Club, Shaun of the Dead, The Avengers, Skyfall, and Jurassic Park. Can you suggest something else I might like? Ocean's Eleven

3. 영화에 대한 질의 응답 + 추천

I loved Billy Madison, Blades of Glory,
Bio-Dome, Clue, and Happy Gilmore.
I'm looking for a Music movie.
School of Rock What else is that about?
Music, Musical, Jack Black, school, teacher,
Richard Linklater, rock, guitar

4. Reddit 대화

I think the Terminator movies really suck, I mean the first
one was kinda ok, but after that they got really cheesy.
Even the second one which people somehow think is great.
And after that... forgeddabotit.
C'mon the second one was still pretty cool.. Arny was still
so badass, as was Sararah Connor's character.. and the way
they blended real action and effects was perhaps
the last of its kind...

질의 응답 + 대화 데이터에 대한 메모리 네트워크의 성능

메모리를 활용한 모델이 우수한 성능을 보임

	단순 질의 응답	영화추천	질의 응답 + 추천	Reddit 대화
SVD	N/A	19.2	N/A	N/A
IR	N/A	N/A	N/A	23.7
LSTM	6.5	27.1	19.9	11.8
Supervised embeddings	50.9	29.2	65.9	27.6
MemN2N	79.3	28.6	81.7	29.2

3. 나와 질의응답을 하는 메모리 네트워크

기계학습 알고리즘의 메모리 사용에 관한 이슈들

- 입력으로 들어오는 정보 중 무엇을 저장할지 말지 어떻게 결정할 수 있을까?
- 메모리에 저장되는 지식을 어떻게 표현해야 할까?
- 메모리에는 여러 종류 (배열, 스택 구조 등)가 있는데 이들을 어떻게 활용할 수 있을까?
- 메모리에 저장되는 정보의 크기가 매우 클 때, 어떻게 효율적으로 탐색할 수 있을까?
- 계층적 메모리를 어떻게 설계할 수 있을까?
- 합성 함수의 사용과 같은 계층적 추론을 어떻게 할 수 있을까?
- 정보의 손실 (망각, unlearning)을 어떻게 구현할 수 있을까?
- 추론 모델을 평가하는 가장 좋은 방법은 무엇일까?
  어떤 문제가 가장 좋은 평가가 될 수 있을까?
- 사람, 동물의 기억 행위에서 얻을 수 있는 영감에는 무엇이 있을까?

학습정리

학습정리

Tacademy

지금까지 [메모리 네트워크]에 대해서 살펴보았습니다.

메모리 네트워크

기존 재현 신경망의 단점: 은닉 유닛 벡터가 제한된 공간에 많은 입력 정보를 저장해야 함
메모리 네트워크: 재현 신경망의 저장 공간 한계 문제를 극복하기 위해 제안되었으며 스토리 선택 모듈 G와 답 선택 모듈 H로 구성됨

$$s^* = \arg\max_{s_i} G(q, s_i) \quad a^* = \arg\max_{a_r} H(s_a, a_r)$$

종단 메모리 네트워크

종단 학습 모델의 필요성: 기존 메모리 네트워크는 스토리 선택 모듈 G를 학습시키기 위해서 질의에 대한 적절한 스토리 정보를 사람이 추가로 작성해야 함
여러 번의 메모리 접근을 위해 '파라미터 공유'라는 재현 신경망의 특징을 가짐

나와 질의응답을 하는 메모리 네트워크

bAbI Task: 모델의 언어 이해 및 추론 능력을 20가지로 나누어 테스트하려는 문제
기존 룰 기반 방식에서 메모리 네트워크를 활용하는 머신러닝 방식의 해결책이 제시됨

인공지능을 위한 머신러닝 알고리즘

12. 딥러닝 응용 사례

Tacademy

CONTENTS

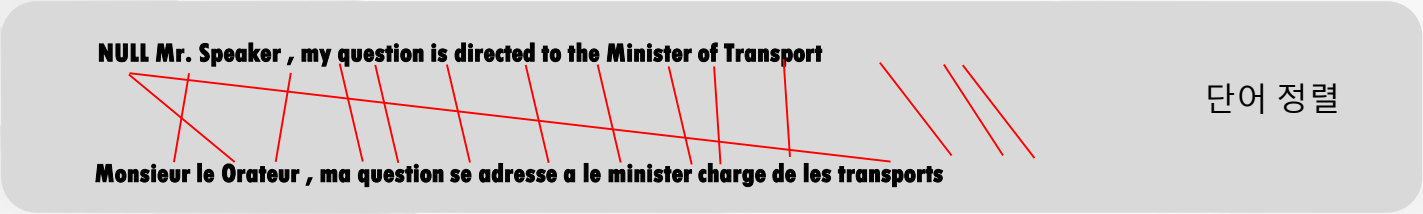
학습 목표

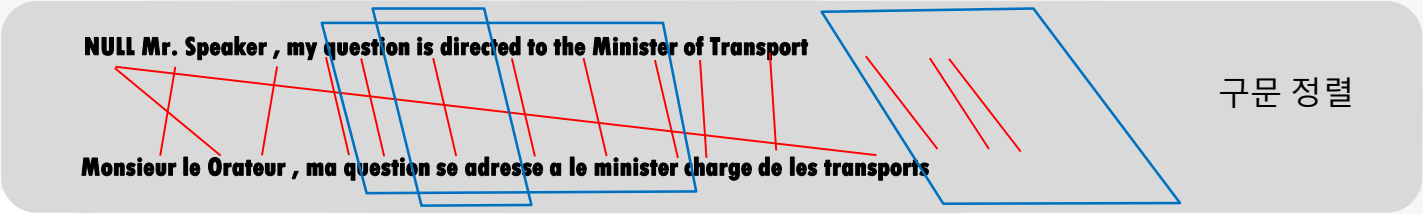
- 딥러닝을 활용한 자동 번역기의 원리를 이해할 수 있다.
- 메모리와 주의집중 기작이 어떻게 딥러닝의 문제를 해결했는지 이해할 수 있다.
- 딥러닝을 활용한 언어와 시각의 결합을 이해할 수 있다.

1. 구글 번역기는 어떤 원리일까?

기존 통계 번역의 문제점

- 과도한 메모리 사용
- 처음 정렬 정보에 따라 번역이 달라짐
- 단어/구문 번역 모델은 의미가 비슷한 단어나 구문을 고려하지 않고, 표면적인 동시 등장 횟수만 고려





한 개의 신경망을 이용한 문장 모델링

- 재현 신경망 (**Recurrent Neural Networks**)은 문장 **X**가 나타날 확률 **P(X)**을 모델링

연속된 데이터들의 집합

$$X = (x_1, x_2, \ldots x_T)$$

X가 나타날 확률

$$p(X) = p(x_1, x_2, \ldots, x_T) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)\ldots p(x_T \mid x_1, \ldots, x_{T-1}) = \prod_{t=1}^{T} p(x_t \mid x_{<t})$$

- 재현 신경망은 매시간 단위 **t**마다 다음을 계산

$$p(x_t \mid x_{<t}) = g(h_{t-1})$$

$$h_{t-1} = \phi(x_{t-1}, h_{t-2}) \quad \phi \text{ is non-linear activation function}$$

- 은닉 층 = 컨텍스트 층 = 히스토리 층
- 역전파를 사용하여 모델 파라미터 학습

두 개의 신경망을 이용한 기계 번역

- 두 개의 재현 신경망이 문장 **X**가 주어졌을 때, **Y**가 나타날 확률을 모델링

$$P(y_1, y_2, \ldots y_{T_b} \mid x_1, x_2, \ldots, x_{T_a})$$

❖ 인코더 재현 신경망

- 재현신경망의 은닉층 h는 컨텍스트(히스토리) 정보를 저장하고 있음 $h_i = \phi_\theta(h_{i-1}, x_i)$
- h_{Ta} 는 전체 입력 문장의 정보를 종합적으로 담고 있음

❖ 디코더 재현 신경망

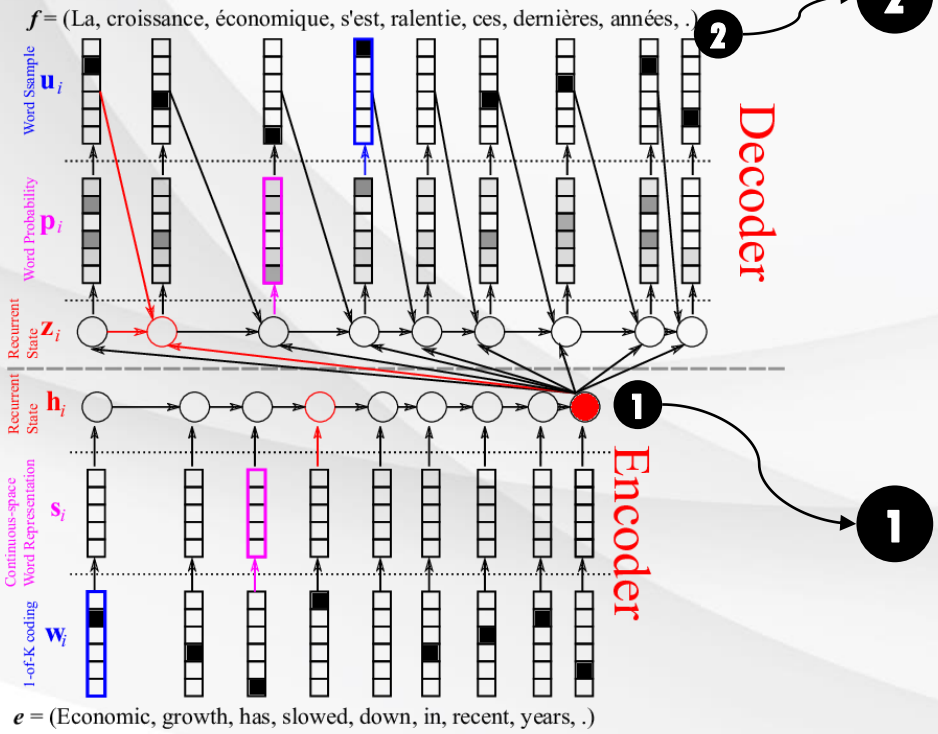
- 각 시간 단위마다, 재현 신경망은 h_{Ta} (입력 문장의 종합적 정보), y_{t-1}(이전 시간 단계에서 생성된 단어), Z_{t-1} (디코더 재현 신경망의 은닉 유닛 정보)를 기반으로 다음 단어 $\mathbf{y_t}$를 예측

$$z_t = \phi_{\theta'}(h_{T_a}, y_{t-1}, z_{t-1})$$
$$p(y_t \mid y_{<t}, X) = g(z_{t-1})$$

1. 구글 번역기는 어떤 원리일까?

두 개의 신경망을 이용한 기계 번역

$$P(y_1, y_2, \ldots y_{T_b} \mid x_1, x_2, \ldots, x_{T_a})$$



디코더 재현 신경망은 매시간 단위마다 h_{Ta}와 이전 시간까지 생성한 단어 정보를 기반으로 새로운 단어 생성

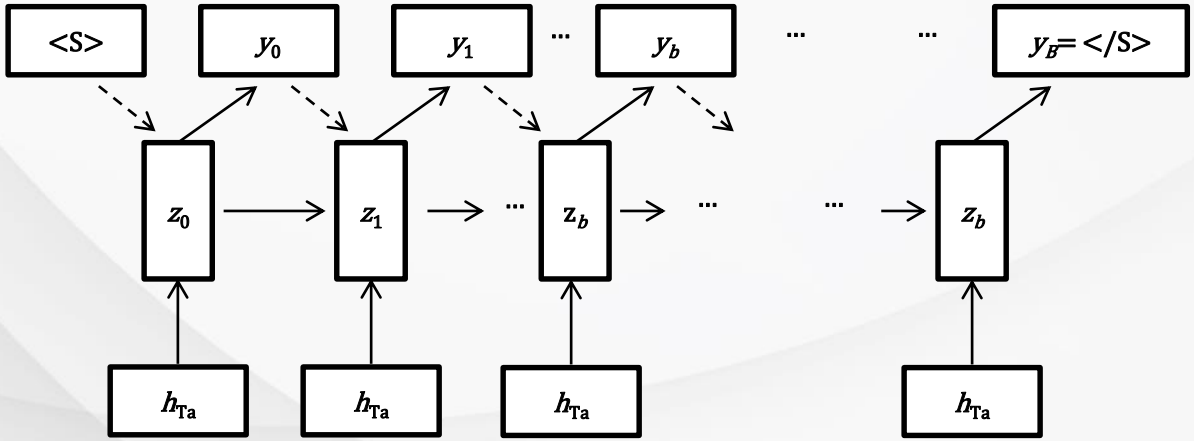
$$z_t = f(U_d y_{t-1} + W_d z_{t-1} + C h_{Ta})$$
$$y_{t-1} = \mathrm{softmax}(V z_t)$$

인코더 재현 신경망은 입력 문장의 단어들을 모두 인코딩한 은닉 벡터 h_{Ta}를 디코더 재현 신경망에 넘겨줌

$$h_t = f(U_e x_t + W_e h_{t-1})$$

https://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-2/

디코더 재현 신경망의 모습



- y_b는 사전에 있는 전체 단어 중 확률이 가장 높은 단어가 샘플링 됨
  - **Softmax** 함수에 의해 계산
  - 단어 y_b는 **b+1**번째 시간에서 입력이 됨
- 디코더 재현 신경망은 이미 다른 데이터로 학습이 완료된 **(pre-trained)** 재현 신경망의 파라미터를 초기 파라미터로 설정할 수 있음

1. 구글 번역기는 어떤 원리일까?

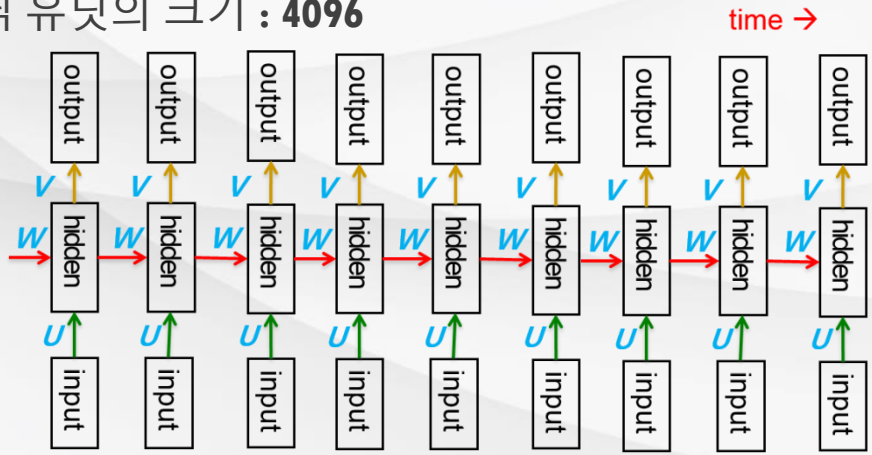
신경망을 이용한 기계 번역의 특징

- 경사하강 법을 이용하여 입력층부터 출력층까지 한꺼번에 종단 학습 가능
- 인코더와 디코더 재현 신경망은 원본 문장과 타겟 문장을 분산 표현으로 나타낼 수 있음
- 번역 문제를 의미적 공간 (은닉 유닛 공간)을 사용하여 학습할 수 있음
- 기존 통계적 기계학습의 방식과 달리 미리 정의된 단어 정렬 방식을 사용하지 않음
- 개념적으로 이해하기 쉬운 디코딩 방식 사용, 음성 인식과 비슷한 크기의 복잡도
- 통계적 기계학습에 비해 적은 모델 파라미터 개수 사용 → 더 적은 메모리 사용

2. 메모리와 주의 집중 기작

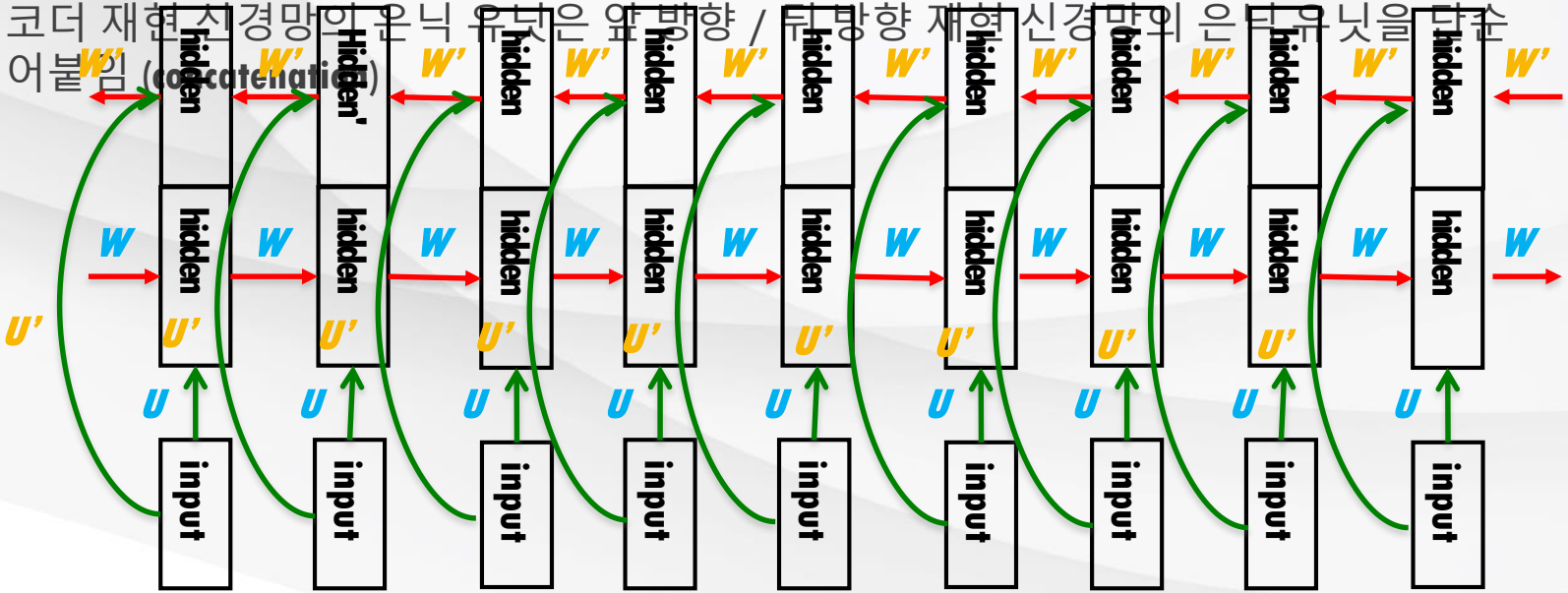
문제점: 제한된 은닉 유닛의 크기

- 인코더 재현 신경망의 가장 마지막 시간의 은닉 유닛 h_{Ta}가 입력 문장의 모든 정보를 담고 있어야함
- 입력의 길이 Ta가 길어질 경우 인코딩해야 할 정보가 많아짐
- 최근에 입력된 정보는 잘 기억하지만, 오래전에 입력된 정보는 손실될 수 있음
  → 번역에서는 큰 문제
- 주로 사용되는 은닉 유닛의 크기 : **4096**



해결책: 양방향 재현 신경망 (bi-directional RNN)

- 앞 방향 재현 신경망: 원본 문장의 가장 앞의 단어부터 인코더 재현 신경망의 입력으로 주어짐
- 뒤 방향 재현 신경망: 원본 문장의 가장 뒤의 단어부터 인코더 재현 신경망의 입력으로 주어짐
- 인코더 재현 신경망의 은닉 유닛은 앞 방향 / 뒤 방향 재현 신경망의 은닉 유닛을 단순 이어붙임 (concatenation)

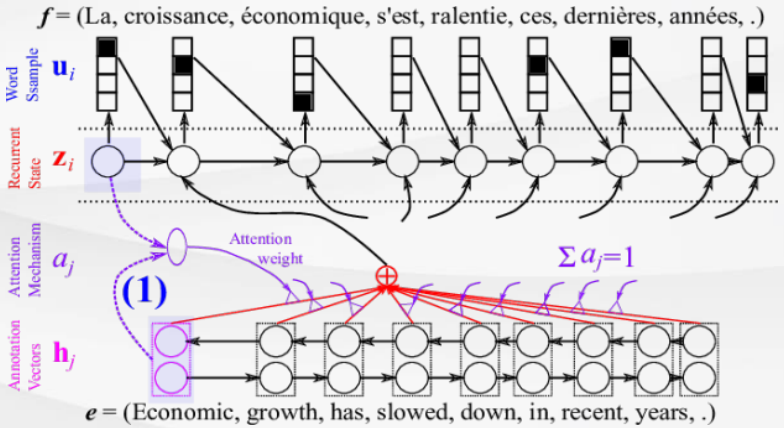


문제점: 메모리와 주의 집중 기작

- 인코더 재현 신경망에서 양방향 재현 신경망에서 계산한 은닉 유닛들을 메모리에 저장
  - 입력 문장의 전체 길이만큼의 은닉 유닛 개수가 저장됨
  - 오래전에 입력으로 주어진 단어의 정보도 갖고 있음

문제점: 메모리와 주의 집중 기작

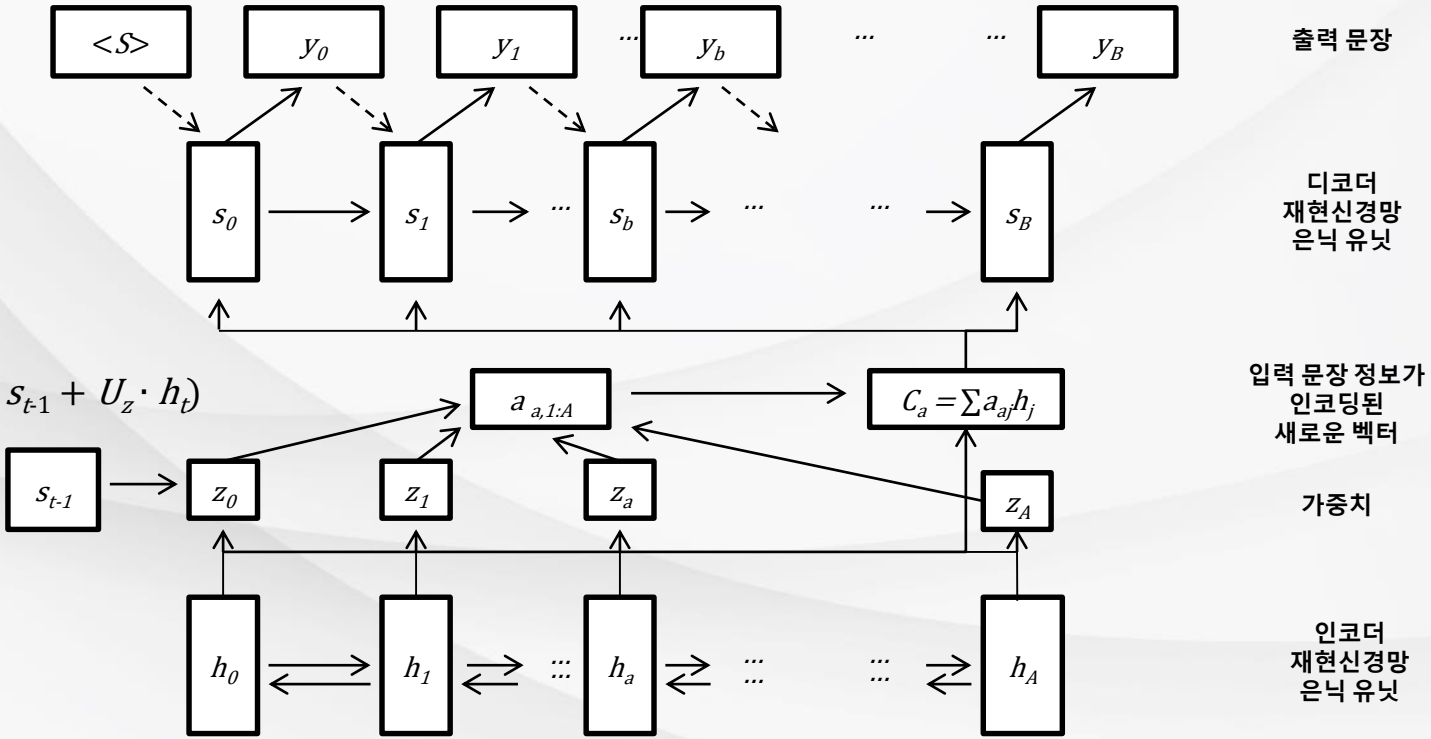
- 디코더 재현 신경망의 입력으로 인코더 재현 신경망 은닉 유닛들의 선형 조합이 사용됨
  - 인코더 신경망의 각 은닉 유닛은 입력 문장에서 한 개의 단어 정보를 인코딩
  - 은닉 유닛들의 가중치의 합: 1
  - 번역기에서 출력되는 단어들은 가중치에 따라 입력으로 주어진 단어들을 선별적으로 고려 (주의 집중)



https://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

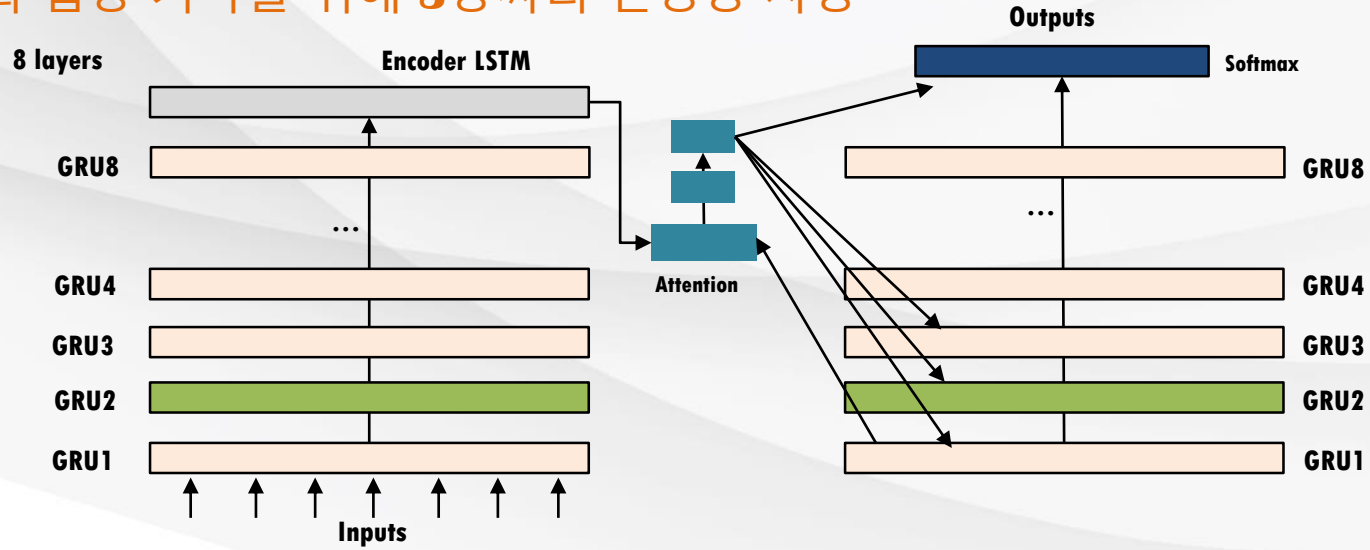
문제점: 메모리와 주의 집중 기작

$$z_t = W \cdot \tanh(V_z \cdot s_{t-1} + U_z \cdot h_t)$$

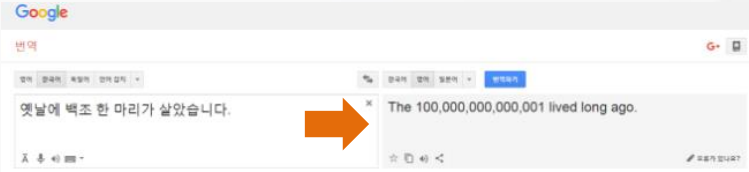


구글의 자동 번역기 (GNMT) (2016.11)

- 재현 신경망 대신 **GRU** 사용
  - 인코더용 **8**층 **GRU** (**2**층 **GRU**는 역방향)
  - 디코더용 **8**층 **GRU**
- 주의 집중 기작을 위해 **3**층짜리 신경망 사용



구글 번역기의 발전 (통계 모델 사용 시)



통계 모델 사용 시

딥러닝 사용 시

However, this invention is not only for the Chinese.

Mr.Zander also hopes to sell the charger to countries that have a poor electiricity supply.

For example, farmers in Senegal use cell phones to check on crop prices, and health workers in South Africa use their phones to check patient records.

그러나, 이 발명품은 중국인을 위한 것이 아닙니다.

mr. Zander는 또한 전기 공급이 부족한 국가에 충전기를 판매하기를 희망하고 있습니다.

예를 들어 세네갈의 농민들은 휴대 전화를 사용하여 작물 가격을 확인하고 South Arica의 의료 종사자는 전화기를 사용하여 환자 기록을 확인합니다.

자동 번역기의 성능 측정 (Precision)

- 정답 번역 문장: **The gunman was shot to death by the police.**
- 계산된 번역 문장:

  - The gunman was shot kill. (4/5)
  - Wounded police jaya of (1/4)
  - The gunman was shot dead by the police. (7/8)
  - The gunman arrested by police kill. (4/6)
  - The gunmen were killed. (1/4)
  - The gunman was shot to death by the police. (9/9)
  - The ringer is killed by the police. (4/7)
  - Police killed the gunman. (3/4)

- 초록색 = **4-gram** 이상 일치 빨강색 = 일치하지 않음

자동 번역기의 성능 측정

- **BiLingual Evaluation Understudy, BLEU (2002년 IBM의 SMT 그룹에서 제안)**
- 기계번역에서 널리 사용됨
- **BLEU** 계산 방식:
  - p_n: 수정된 **n-gram precision** (일치하는 중복 단어 무시)
  - $p_1, p_2, .., p_n$의 기하 평균
  - **BP: Brevity penalty (c=**번역된 문장의 길이, **r=**정답 문장의 길이)
  - 짧게 생성된 번역 문장일수록 높은 점수를 얻게 되는 현상 방지

$$BLEU = BP\left(\prod_{n=1}^{N} p_n\right)^{\frac{1}{N}}$$

$$BP = \begin{cases} 1 & \text{if} \quad c > r \\ r/c & \text{if} \quad c \le r \end{cases}$$

  - 주로, **N=4**를 사용

자동 번역기의 성능 측정

- 계산된 번역 문장: **The gunman was shot dead by police .**

정답 1: **The gunman was shot to death by the police .**
정답 2: **The gunman was shot to death by the police .**
정답 3: **Police killed the gunman .**
정답 4: **The gunman was shot dead by the police .**

- **Precision:** $p_1=1.0(8/8)$, $p_2=0.86(6/7)$, $p_3=0.67(4/6)$, $p_4=0.6\ (3/5)$
- **Brevity Penalty:** $c=8$, $r=10$, $BP=0.8$
- 최종 점수: $\sqrt[4]{1 * 0.86 * 0.67 * 0.6} * 0.8 = 0.613$

3. 이미지를 자동으로 설명해주는 기계

신경망을 이용한 이미지 번역

- 한 개의 컨볼루션 신경망과 재현 신경망이 이미지 **I**가 주어졌을 때, 설명문 **Y**가 나타날 확률을 모델링

$$P(y_1, y_2, \ldots y_{T_b} \mid I)$$

❖ 인코더 컨볼루션 신경망

- 이미지를 여러 등분 $I=(i_1, i_2, \ldots, i_a, \ldots, i_A)$으로 나눈 뒤 컨볼루션 신경망으로 부분 이미지 i_a를 인코딩
- 인코딩된 벡터들을 인코더 재현 신경망의 은닉 유닛처럼 사용 (매 시간 단위마다 부분 이미지 i_a 가 입력됨)

$$h_a = CNN(i_a)$$

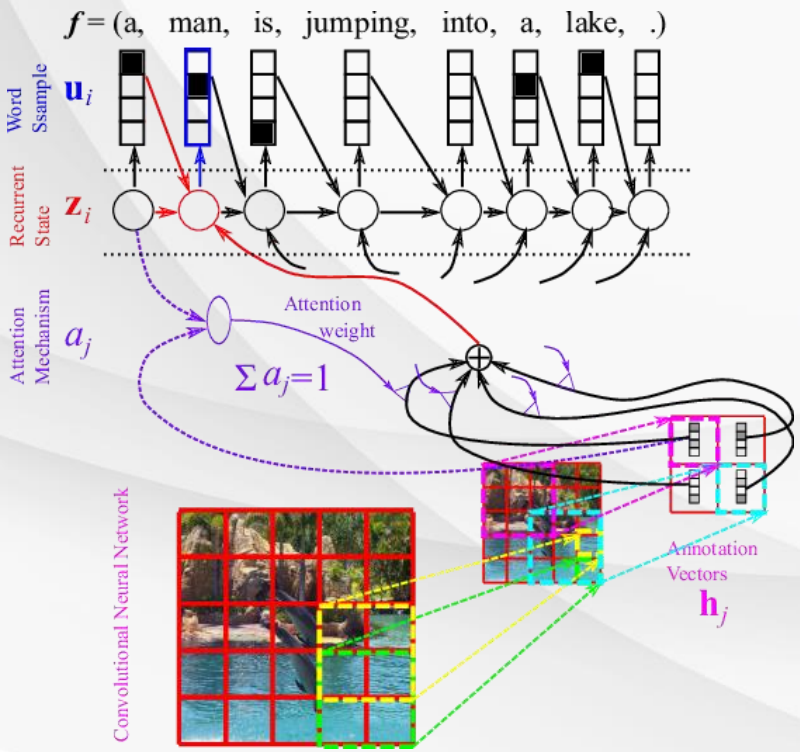
❖ 디코더 재현 신경망

- 기계 번역 세팅과 같이, 각 시간 단위마다, 재현 신경망은 h_a(입력 이미지의 정보), y_{t-1}(이전 시간 단계에서 생성된 단어), z_{t-1}(디코더 재현 신경망의 은닉 유닛 정보)를 기반으로 다음 단어 y_t를 예측

$$z_t = \phi_{\theta'}(h_a, y_{t-1}, z_{t-1})$$

$$p(y_t \mid y_{<t}, X) = g(z_{t-1})$$

신경망을 이용한 이미지 번역



설명문 출력

주의 집중 기작

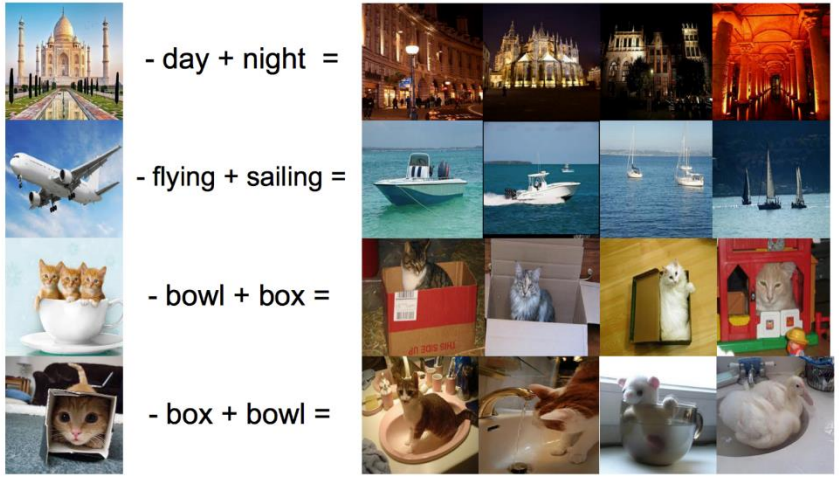
이미지 인코딩

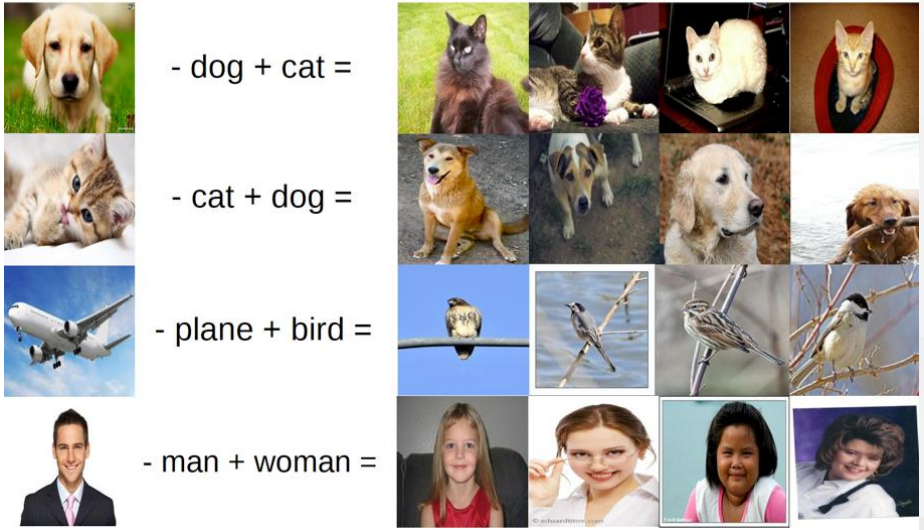
이미지 분할

https://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

이미지 번역의 예시

"little girl is eating piece of cake." "baseball player is throwing ball in game." "woman is holding bunch of bananas." "black cat is sitting on top of suitcase."





(Kiros, Salakhutdinov, Zemel, TACL 2015)

학습정리

지금까지 [딥러닝 응용 사례]에 대해서 살펴보았습니다.

구글 번역기는 어떤 원리일까?

인코더-디코더 모델 사용
인코더 재현 신경망: 원본 문장을 연속된 벡터 공간에 임베딩 시킴
디코더 재현 신경망: 원본 문장의 정보 h_{Ta}와 이전 단계의 출력 단어 y_{t-1},
디코더의 은닉 유닛 s_{t-1}을 기반으로 다음 단어 y_t 예측

메모리와 주의 집중 기작

재현 신경망의 은닉 유닛의 크기가 한정되어 있기 때문에 오래전 입력으로 들어온 단어의 정보가 손실됨
인코더 재현 신경망의 은닉 유닛들을 메모리에 저장하고 선형 조합 (주의 집중 기작)

이미지를 자동으로 설명해주는 기계

이미지를 여러 등분으로 나눈 뒤, 부분 이미지들을 컨볼루션 신경망으로 인코딩 시킴
인코딩된 벡터들은 순서대로 디코더 재현 신경망에 입력됨

인공지능을 위한 머신러닝 알고리즘

13. Weka를 이용한 머신러닝 실습

Tacademy

CONTENTS

학습 목표

- Weka를 사용하여 데이터를 전처리하고 분석할 수 있다.
- Weka로 분류를 하기 위해 알고리즘의 파라미터를 설정할 수 있다.
- Weka로부터 얻은 분류 결과를 해석할 수 있다.

1. Weka 소개

Weka란?

- **Weka(Waikato Environment for Knowledge Analysis)**
  - 뉴질랜드의 **Waikato** 대학교 컴퓨터공학부에서 제작
  - **Weka**는 뉴질랜드에서만 발견되는 새이기도 함

- 대표적인 기계학습 알고리즘 모음, 데이터 마이닝 도구

Weka의 특징

❖ Weka의 주요 기능

- 데이터 전처리, 특징값 선별(Feature Selection)
- 군집화, 데이터 가시화
- 분류, 회귀 분석, 시계열 예측

❖ 소프트웨어 특성

- 무료 및 소스 공개 소프트웨어 (free & open source GNU General Public License)
- Java로 구현, 다양한 플랫폼에서 실행 가능

Weka를 구성하는 인터페이스

Explorer : 다양한 분석 작업을 한 단계씩 분석 수행 및 결과 확인 가능,
일반적으로 가장 먼저 실행

Experimenter: 분류 및 회귀 분석을 일괄 처리. 결과 비교 분석
- 다양한 알고리즘 및 파라미터 설정
- 여러 데이터 알고리즘 조합 동시 분석
- 분석 모델 간 통계적 비교
- 대규모 통계적 실험 수행

KnowledgeFlow: 데이터 처리 과정의 주요 모듈을 그래프로 가시화하여 구성

Simple CLI: 다른 인터페이스를 컨트롤하는 스크립트 입력창
Weka의 모든 기능을 명령어로 수행 가능

2. Weka로 붓꽃(iris) 분류하기

기계학습 – 패턴 분류 절차

❖ 피처 정의 (features or attributes)

- **sepal length, sepal width, petal length, petal width**
- 클래스 (**class**) **label:** 붓꽃의 세 아종을 예측 목표 변수로 설정

setosa, versicolor, or virginica

❖ 샘플 수집 및 데이터셋 구성

- 붓꽃의 각 아종 별로 **50**개체의 피처를 측정
- 데이터셋**: 2**차원 표 형태**: 150 samples (or instances) * 5 attributes**

기계학습 – 패턴 분류 절차

❖ 패턴 분류 수행

- 기계학습 알고리즘을 활용
- 예> 결정 트리, 랜덤 포레스트, **SVM**, 다층 퍼셉트론

❖ 패턴 분류 성능 평가

- 패턴 분류 모델의 상대적 비교 과정 (알고리즘 + 파라미터 설정)
- 다양한 평가 기준을 적용: 분류 정확도, **precision** + **recall** 등

피처 정의 – 붓꽃(iris)

Iris setosa

Iris versicolor

Iris virginica

분류에 사용할 꽃의 특징

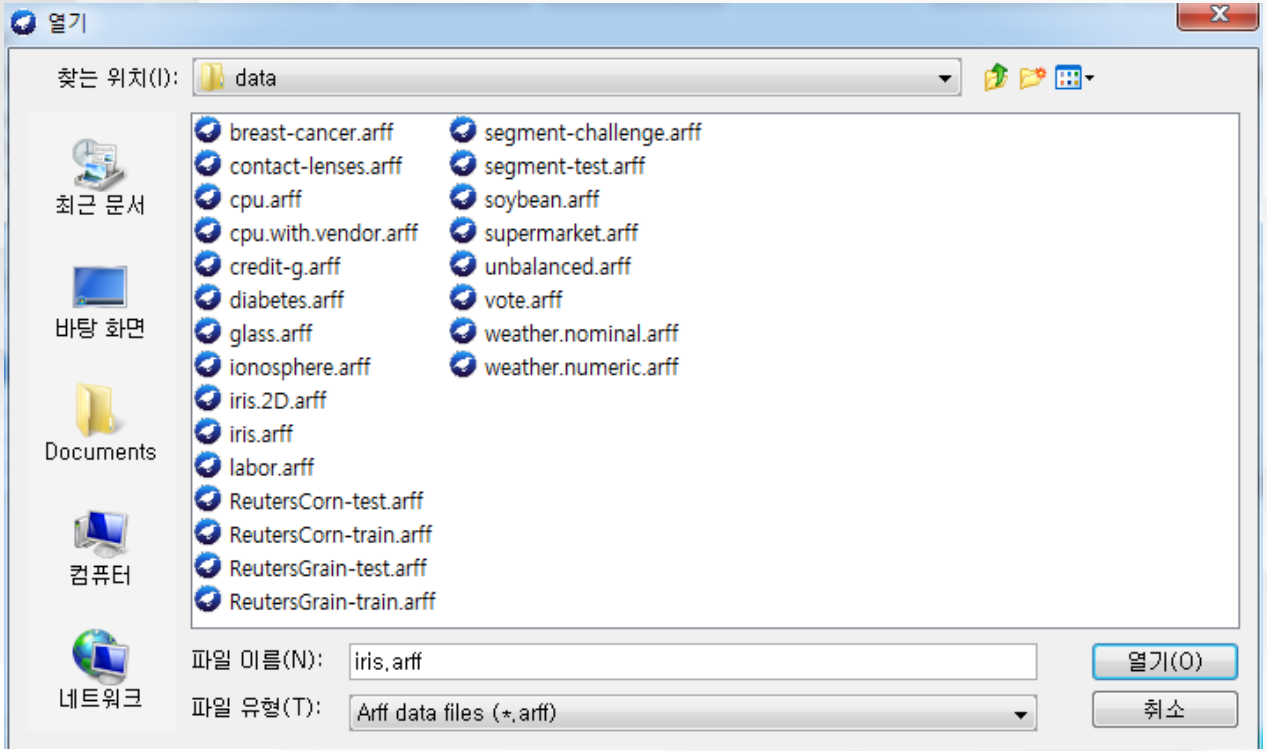
꽃받침 **(Sepal)**	길이 **(Length)**
꽃잎 **(Petal)**	너비 **(Width)**

특징 기준 판별

2.5, 9.4, 1.0, 0.5, Iris-setosa
1.2, 9.2, 1.2, 0.2, Iris-setosa
1.9, 9.0, 1.4, 0.4, Iris-setosa
…
7.2, 1.2, 3.4, 9.1, Iris-versicolor

데이터셋 – 파일 열기

Weka 폴더 → **'data'** 폴더에서 **'iris.arff'** 파일 선택



데이터셋 - Weka의 데이터 형식 (.arff)

Dataset name

Attribute name

Attribute type

헤더

```
@RELATION iris

@ATTRIBUTE sepallength REAL
@ATTRIBUTE sepalwidth REAL
@ATTRIBUTE petallength REAL
@ATTRIBUTE petalwidth REAL
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}
```

데이터
(csv format)

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
```

Excel을 이용하여 csv 파일 생성 후, 헤더만 추가하면 쉽게 arff 포맷의 파일 생성 가능

데이터셋 구성 - 데이터 전처리

- 예측 모델 학습 및 평가를 위해 준비하는 데이터 집합을 모델에 입력하기 전에 다양한 처리를 하여 데이터의 품질을 향상시키는 과정

Data cleaning

Data integration

Data transformation

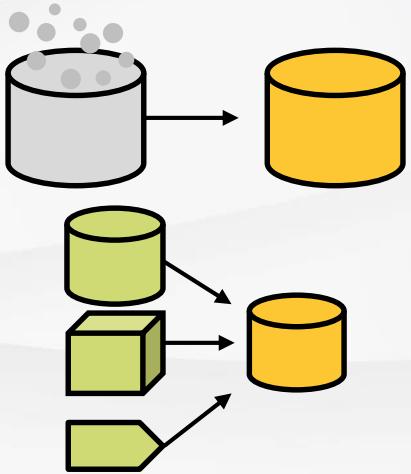
Data reduction

5.1, 3.5, 1.4, 0.2, Iris-setosa → **4.2, 1.5, 1.3, Iris-setosa**

T1						
T2						
…						
T2000						

→

T1						
T2						
…						
T450						



데이터셋 구성 – 전처리(1)

- 적용할 **filter**를 선택
- 특징 **(attribute)**
  - 데이터 차원 축소

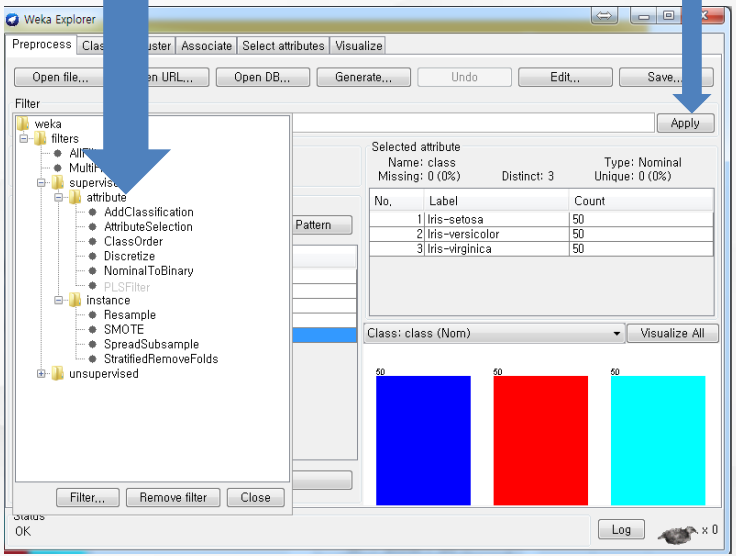
    weka.filters.supervised.attribute.AttributeSelection
  - 데이터 변형 및 데이터 이산화

    weka.filters.supervised.attribute.Discretize

    weka.filters.unsupervised.attribute.Normalize, Standardize

적용 가능한
데이터 전처리 기법

선택 후 **apply** 버튼 클릭



데이터셋 구성 – 전처리(1)

❖ 데이터 인스턴스

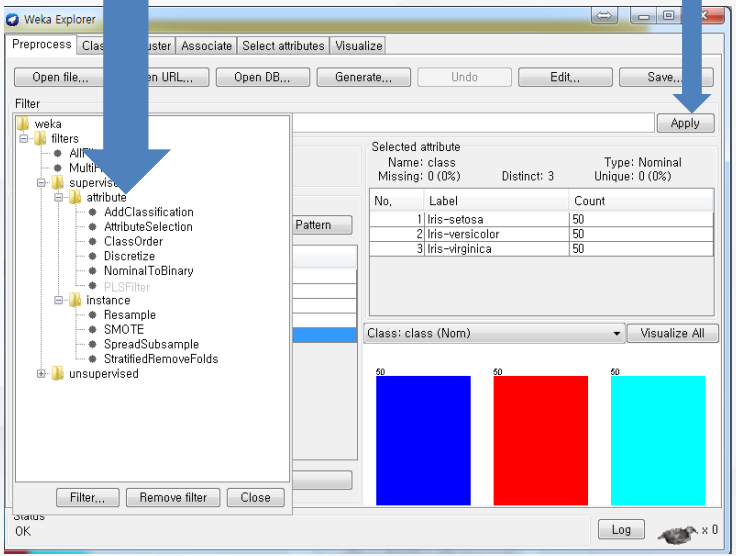
- 데이터 개수 증대

weka.filters.supervised.instance.Resample

weka.filters.supervised.instance.SMOTE

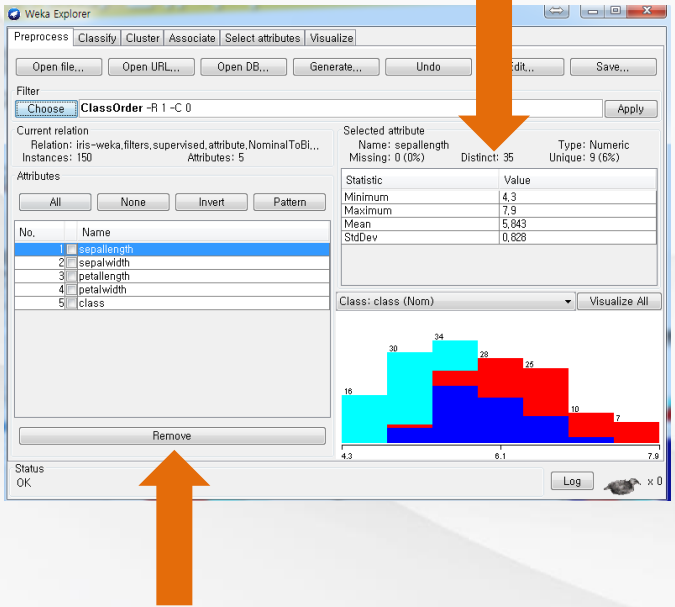
적용 가능한
데이터 전처리 기법

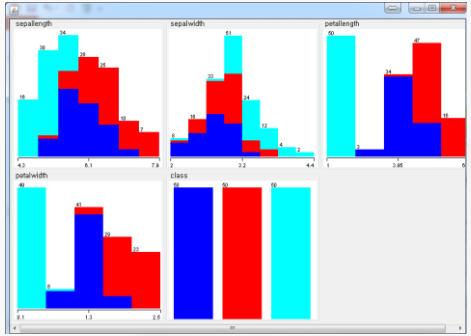
선택 후 apply 버튼 클릭



데이터셋 구성 – 전처리(2)

특징값 별 기초적 통계 분석
(Selected Attribute)





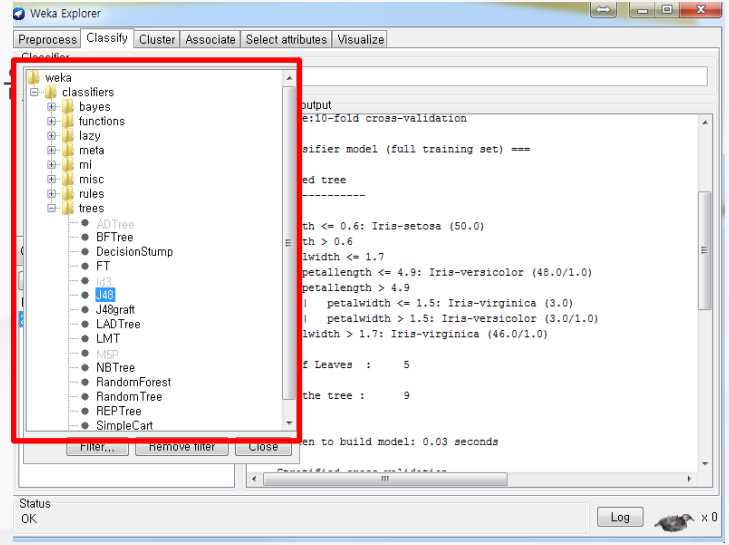
모든 특징값을 대상으로
클래스 레이블 분포 가시화

특징값 삭제 (Remove Attributes)

패턴 분류 수행 - 알고리즘 선택

❖ J48 (c4.5의 Java 구현 버전)

- 학습 결과 모델에서 분류 규칙을 '트리'형태로 얻을 수 있음
- **Weka**에서 찾아가기: **classifiers-trees-J48**



❖ 랜덤 포레스트

- 결정 트리의 앙상블 모델
- 특징들을 무작위로 선택하여 결정 트리들이 생성됨
- 분류 과정에서 각 트리는 투표를 하며 가장 많은 표를 얻은 클래스가 선택됨
- **Weka**에서 찾아가기: **classifiers-trees-RandomForest**

❖ 다층 퍼셉트론

- 실용적으로 매우 폭 넓게 쓰이는 대표적 분류 알고리즘
- **Weka**에서 찾아가기: **classifiers-functions-MultilayerPerceptron**

분류 알고리즘의 파라미터 설정

❖ 파라미터 설정 = 자동차 튜닝

- 많은 경험 또는 시행착오 필요
- 파라미터 설정에 따라 동일한 알고리즘에서도 최악에서 최고의 성능을 모두 보일 수도 있음

❖ 결정트리의 주요 파라미터 (J48, SimpleCart in Weka)

- 트리의 크기에 직접적 영향을 주는 파라미터: **confidenceFactor, pruning, minNumObj** 등

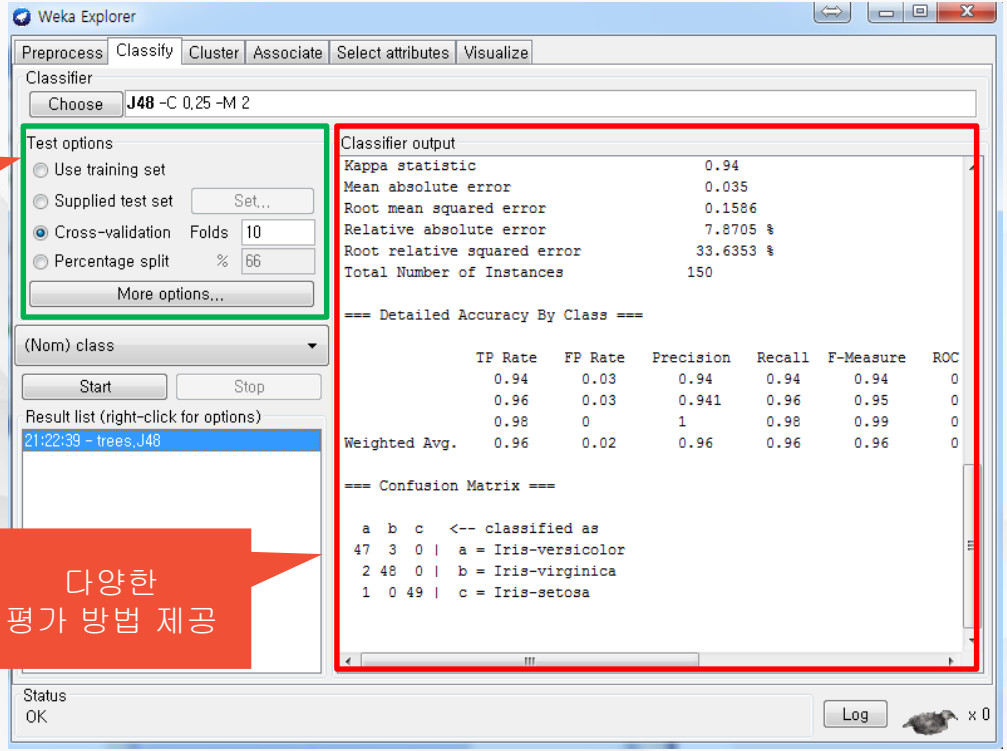
❖ Random Forest의 주요 파라미터 (RandomForest in Weka)

- **numTrees:** 학습 및 예측에 참여할 **tree**의 수를 지정. 대체로 많을 수록 좋으나, **overfitting**에 주의해야 함

❖ 참고: 신경망의 주요 파라미터 (MultilayerPerceptron in Weka)

- 구조 관련: **hiddenLayers,**
- 학습 과정 관련: **learningRate, momentum, trainingTime (epoch), seed**

패턴 분류 성능 평가



평가를 위한 데이터 집합 세팅

다양한 평가 방법 제공

패턴 분류 성능 평가

실행 정보

```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     iris-weka.filters.supervised.attribute.NominalToBinary-w
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:10-fold cross-validation
```

분류 모델
(훈련 데이터 학습 모델)

```
=== Classifier model (full training set) ===

J48 pruned tree
------------------

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves  :     5

Size of the tree :      9
```

평가 결과
- 종합적 요약
- 클래스별 성능
- Confusion Matrix

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.035
Root mean squared error                  0.1586
Relative absolute error                  7.8705 %
Root relative squared error             33.6353 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area  Class
                   0.94      0.03       0.94      0.94      0.94       0.952    Iris-versicolor
                   0.96      0.03       0.941     0.96      0.95       0.961    Iris-virginica
                   0.98      0          1         0.98      0.99       0.99     Iris-setosa
Weighted Avg.      0.96      0.02       0.96      0.96      0.96       0.968

=== Confusion Matrix ===

  a  b  c   <-- classified as
 47  3  0 |  a = Iris-versicolor
  2 48  0 |  b = Iris-virginica
  1  0 49 |  c = Iris-setosa
```

결과는 모델에 따라 변할 수 있음

학습정리

학습정리

Tacademy

지금까지 [Weka를 이용한 머신러닝 실습]에 대해서 살펴보았습니다.

Weka의 기능

데이터 전처리: 특징값 선별(feature selection) / 제거, 데이터 리샘플링
데이터 분류: 결정 트리, 랜덤 포레스트, 다층 퍼셉트론 등 다양한 알고리즘 선택
회귀 분석 및 시계열 예측

arff 파일

헤더: relation, attributes, data
데이터: csv format
Excel을 이용하여 csv 파일 생성 후, 헤더를 추가하여 arff 파일 생성

머신러닝 실습과 해석

테스트 옵션: cross validation
결과 해석: confusion matrix, classification accuracy, precision / recall

인공지능을 위한 머신러닝 알고리즘

14. Theano를 통한 머신러닝 구현

Tacademy

CONTENTS

학습 목표

- Theano의 기본적인 문법과 Symbolic Expression을 이해할 수 있다.
- Theano의 GPU 연산 과정을 이해할 수 있다.
- Theano를 사용하여 신경망을 구현해보고 실행할 수 있다.

1. Theano란?

Theano의 특징

- LISA Lab(https://mila.umontreal.ca/en/)에서 만든 Python 기반 오픈소스 Package

http://deeplearning.net/software/theano/

장점

- Symbolic 연산 철학으로 간결하고 빠르게 모델 구현 가능
- Symbolic 미분이 가능하므로 역전파 등을 직접 구현할 필요가 없음
- 동일한 코드를 CPU와 GPU에서 모두 사용 가능
- Python 기반이므로, numpy, scipy 등 다양한 Python 패키지와의 연동할 수 있음

단점

- 복잡하고 알기 어려운 에러 메시지

Symbolic expression 정의 - (1) scalar

```
from theano import tensor as T

x = T.scalar()
y = T.scalar()
z = x + y
w = z * x
a = T.sqrt(w)
b = T.exp(a)
c = a ** b
d = T.log(c)
```

- x, y: Symbolic 변수 정의
- z, w: Symbolic Expression
- a: Symbolic Expression ($a = w^2$)
- b: Symbolic Expression ($b = e^a$)
- c: Symbolic Expression ($c = a^b$)
- d: Symbolic Expression ($d = \ln(c)$)

Symbolic expression 정의 - (2) vector & matrix

```
from theano import tensor as T

x = T.matrix()
y = T.matrix()
a = T.vector()
b = T.vector()
c = T.dot(x, y)
d = T.dot(x, a)
e = T.dot(a, b)
f = a * b
h = e + f
```

- x, y, a, b ← **Symbolic 변수 정의**
- c ← **Symbolic Expression (matrix-matrix product)**
- d ← **Symbolic Expression (matrix-vector product)**
- e ← **Symbolic Expression (vector-vector product)**
- f ← **Symbolic Expression (element-wise product)**
- h ← **Symbolic Expression (broadcasting)**

Symbolic expression 정의 - (3) theano function

```
>>> from theano import tensor as T
>>> x = T.scalar()
>>> y = T.scalar()
>>> from theano import function
>>> f = function([x, y], x + y)
>>> f(1., 2.)
array(3.0)
```

컴파일

첫 번째 인수는 입력 symbolic 변수들의 리스트
두 번째 인수는 출력 symbolic 변수

scalar를 사용하므로 scalar 값 입력

출력된 scalar 값

Symbolic 미분 연산

$Y= -\ln(3x^2+5x)$의 미분

```
>>> from theano import tensor as T
>>> x = T.scalar()
>>> y = -1*T.log(3*x**2+5*x)
>>> y_prime = T.grad(y,x)
>>> from theano import function
>>> f = function([x], y_prime)
>>> f(1)
array(-1.375, dtype=float32)
```

Symbolic 미분

자동으로 $Y=-\ln(3x^2+5x)$의 도함수 계산

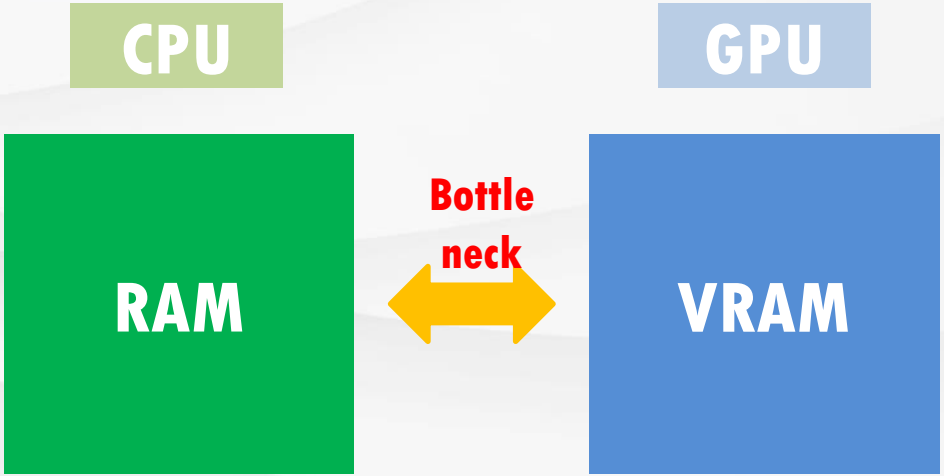
복잡한 역전파 계산을 직접 구현할 필요가 없음

2. Theano로 GPU 프로그래밍 실습하기

shared variables

shared variable은 데이터를 RAM에서 GPU의 VRAM로 옮기는 명령어

```
>>> from theano import shared
>>> x = shared(0.)
>>> x.get_value()
0.0
```



givens

- symbolic 변수에 shared 데이터를 대입
- Theano 함수 f = x + y가 존재할 때, symbolic 변수 x, y의 값 1, 2를 설정하는 방법

```
>>> f = function([x, y], x + y)
>>> f(1., 2.)
array(3.0)
```

RAM → VRAM → GPU 연산

```
>>> x_val = theano.shared(1)
>>> y_val = theano.shared(2)
>>> f = function([x, y], x + y, givens = [(x, x_val), (y, y_val)])
>>> f()
```

VRAM → GPU 연산

updates

- GPU 연산 결과를 이용해 shared 데이터를 수정

```
>>> x_val = theano.shared(1)
>>> f = function([], x_val, updates = (x_val, x_val+1))
>>> f()
```

실행 시 RAM을 거치지 않고 GPU 내에서 x_val을 1씩 증가시킴

shared variable 예제

```
>>> from theano import shared
>>> x = shared(0.)
>>> from theano.compat.python2x import OrderedDict
>>> updates = OrderedDict()
>>> updates[x] = x + 1
>>> f = function([], updates=updates)
>>> f()
[]
>>> x.get_value()
1.0
>>> x.set_value(100.)
>>> f()
[]
>>> x.get_value()
101.0
```

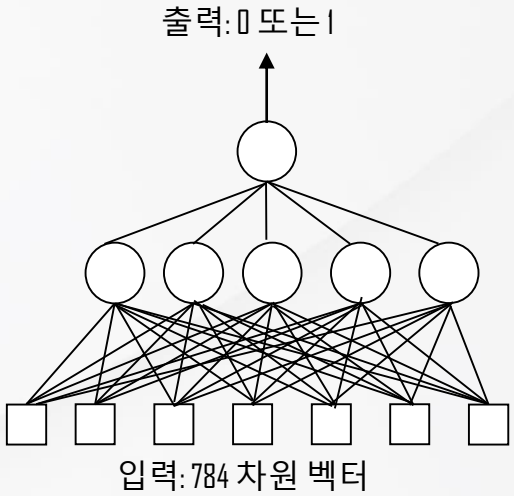
3. Theano로 신경망 구현하기

신경망 코드

```
import numpy
import theano
import theano.tensor as T
rng = numpy.random

N = 400
feats = 784
D = (rng.randn(N, feats).astype(numpy.float32), rng.randint(size = N, low=0,
high=2).astype(numpy.float32))
```

데이터 샘플 개수

특징 차원 크기



D = {X,Y}

N = 400

feats = 784

0.12	0.49	1.65	-1.45	2.12	0.21	-2.12	0.23

N = 400

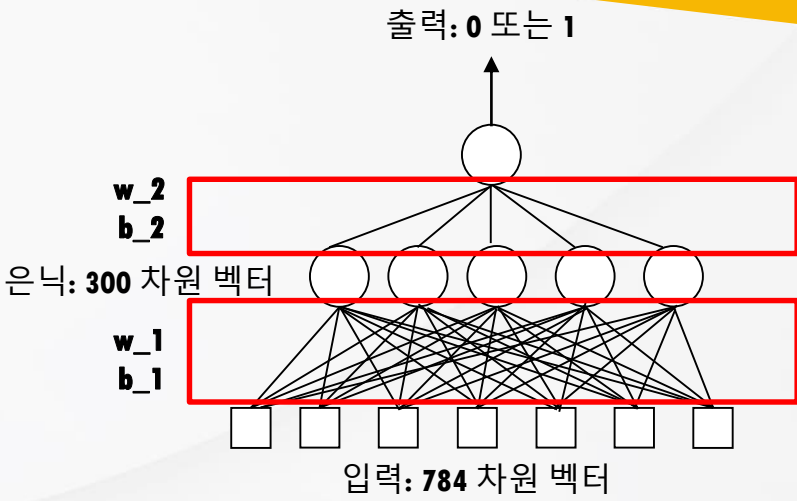
1
1
0
0
1
1

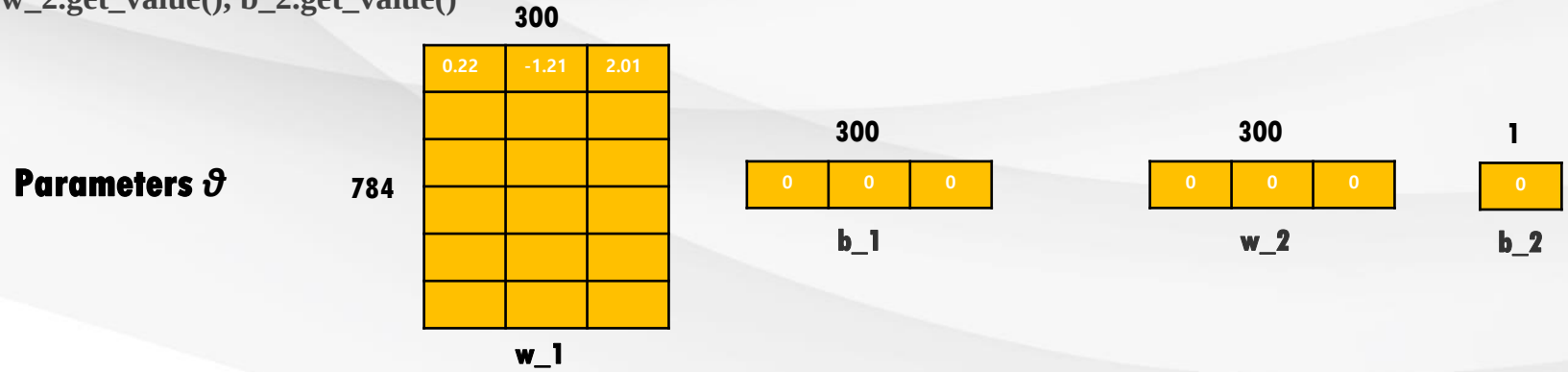
레이블

각 특징 값들은 평균 0, 분산 1인 가우시안 분포에서 무작위로 샘플링

신경망 코드

```
training_steps = 10,000
x = T.matrix('x')
y = T.vector('y')
w_1 = theano.shared(rng.randn(784, 300), name = 'w1')
b_1 = theano.shared(numpy.zeros(300,), name = 'b1')
w_2 = theano.shared(rng.randn(300,), name = 'w2')
b_2 = theano.shared(0., name = 'b2')

print w_1.get_value(), b_1.get_value()
print w_2.get_value(), b_2.get_value()
```





신경망 코드

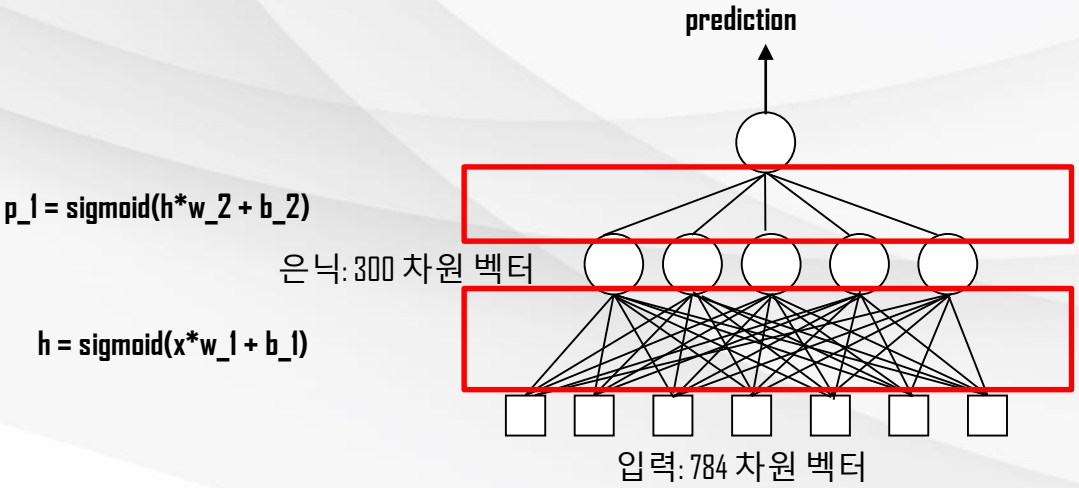
```
p_1 = T.nnet.sigmoid(T.dot(T.nnet.sigmoid(T.dot(x, w_1)+b_1), w_2)+b_2)
prediction = p_1 > 0.5
xent = -y*T.log(p_1) - (1-y)*T.log(1-p_1)
cost = xent.mean() + 0.01 * ((w_1**2).sum()
                + (w_2**2).sum() + (b_1**2).sum() + (b_2**2).sum())
gw_1, gb_1, gw_2, gb_2 = T.grad(cost, [w_1, b_1, w_2, b_2])
```

타겟이 1일 확률

Cross entropy

L2 regularization를 적용한 손실 함수

손실 함수에 대한 파라미터의 변화량



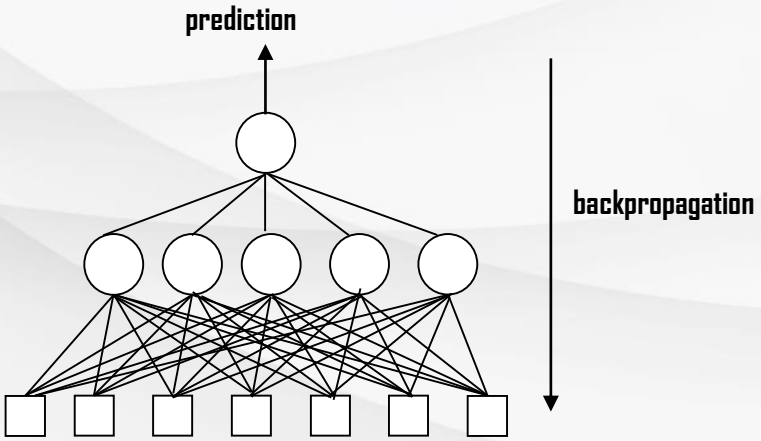
신경망 코드

```
train = theano.function(inputs = [x, y],
                        outputs = [prediction, xent],
                        updates = {w_1 : w_1-0.1*gw_1, b_1 : b_1-0.1*gb_1,
                                   w_2 :  w_2-0.1*gw_2, b_2 : b_2-0.1*gb_2})
predict = theano.function(inputs = [x], outputs = prediction)
```

컴파일

학습률:0.1

예측 함수



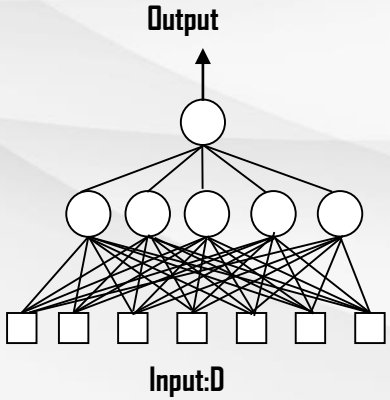
신경망 코드

```
for i in range(training_steps):
        pred, err = train(D[0], D[1])
```

훈련

```
print "Final model:"
print w_1.get_value(), b_1.get_value()
print w_2.get_value(), b_2.get_value()
print "target values for D: ", D[1]
print "predictions on D: ", predict(D[0])
```

예측



학습정리

학습정리

Tacademy

지금까지 [Theano를 통한 머신러닝 구현]에 대해서 살펴보았습니다.

Theano란?

symbolic 연산 철학으로 간결하고 빠르게 모델 구현 가능
symbolic 미분이 가능하므로 역전파 등을 직접 구현할 필요가 없음 (grad 함수 사용)

```
x = T.scalar()    y = -1*T.log(3*x**2+5*x)    y_prime = T.grad(y,x)
```

Theano로 GPU 프로그래밍 실습하기

shared variables: RAM에서 GPU VRAM으로 데이터를 옮겨줌
givens: symbolic 변수에 shared variables를 대입
updates: GPU 연산 결과를 이용해 shared variables의 값을 수정

Theano로 신경망 구현하기

Theano를 사용하여 머신러닝 알고리즘을 구현할 경우, GPU를 사용하여 빠른 학습 가능
간결한 코드 작성 가능, 미분의 자동 계산으로 프로그래머의 일을 줄여줌

인공지능을 위한 머신러닝 알고리즘

15. Keras를 통한 딥러닝 구현 및 실습

Tacademy

CONTENTS

학습 목표

- Keras 라이브러리의 특징과 기본적인 함수들을 사용할 수 있다.
- Keras로 컨볼루션 신경망을 구현할 수 있다.
- Keras를 사용하여 이미지를 분석할 수 있다.

1. Keras란?

Keras의 특징

- Keras의 기본 아이디어는 모델의 층들과 그것들의 입력과 출력에 있음
- Keras를 사용하여 모델 구축하기
  1. 입력/출력 데이터를 준비
  2. 첫 번째 층을 생성하고 입력 데이터에 맞게 설정해 줌
  3. 마지막 층을 생성하고 출력 데이터에 맞게 설정해 줌
  4. 입력 층과 출력 층 사이에 원하는 레이어를 생성해 줌

층 (layer)

- **Keras**는 미리 구현되어 있는 다양한 층들을 제공해줌:
  - 다층 퍼셉트론에서 사용되는 **Dense**층

Dense
keras.layers.core.Dense(output_dim, init='glorot_uniform', activation=**None**, weights=**None**, W_regularizer=**None**, b_regularizer=**None**, activity_regularizer=**None**, W_constraint=**None**, b_constraint=**None**, bias=**True**, input_dim=**None**)

  - 재현층, **LSTM, GRU, etc**

Reccurent
keras.layers.recurrent.Recurrent(weights=**None**, return_sequences=**False**, go_backwards=**False**, stateful=**False**, unroll=**False**, consume_less='cpu', input_dim=**None**, input_length=**None**)

층 (layer)

1D 컨볼루션 층

Convolution1D
keras.layers.convolutional.Convolution1D(nb_filter, filter_length, init='glorot_uniform', activation=**None**, weights=**None**, border_mode='valid', subsample_length=1, W_regularizer=**None**, b_regularizer=**None**, activity_regularizer=**None**, W_constraint=**None**, b_constraint=**None**, bias=**True**, input_dim=**None**, input_length=**None**)

2D 컨볼루션 층

Convolution2D
keras.layers.convolutional.Convolution2D(nb_filter, nb_row, nb_col, init='glorot_uniform', activation=**None**, weights=**None**, border_mode='valid', subsample=(1, 1), dim_ordering='default', W_regularizer=**None**, b_regularizer=**None**, activity_regularizer=**None**, W_constraint=**None**, b_constraint=**None**, bias=**True**)

활성함수 (activation function)

❖ 다양한 활성함수

- 간단한 활성함수: **Sigmoid, tanh, ReLu, softplus, hard_sigmoid, linear**
- 복잡한 활성함수: **LeakyReLu, PReLu, ELU, Parametric Softplus, Thresholded linear and Thresholded Relu**

❖ 목표 함수

- 에러 측정용 목표함수: **rmse, mse, mae, mape, msle**
- 최대 마진 목표함수: **squared_hinge, hinge**
- 분류용 목표함수: **binary_crossentropy, categorical_crossentropy**

저장/로드

❖ 모델 구조의 저장/로드 가능

```
from models import model_from_json

json_string = model.to_json()
model = model_from_json(json_string)
```

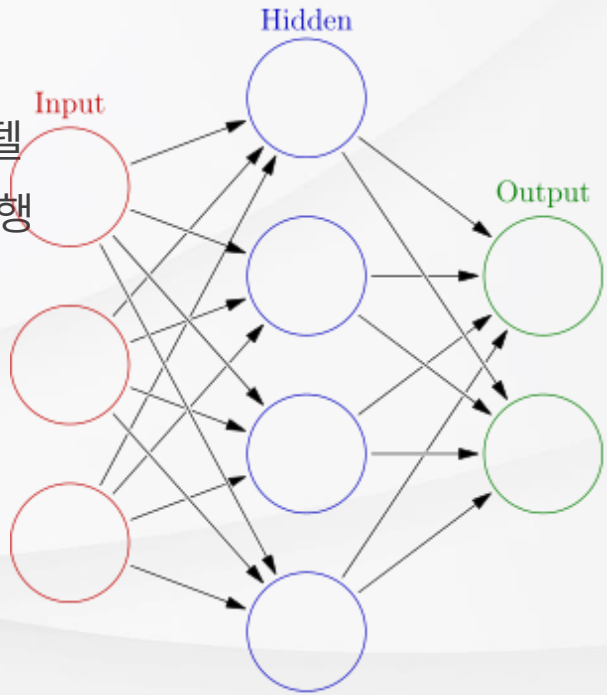
```
from models import model_from_yaml

yaml_string = model.to_yaml()
model = model_from_yaml(yaml_string)
```

❖ 모델 파라미터의 저장/로드 가능

- **model.save_weights(filepath):** 모델의 가중치를 **HDF5 file**로 저장
- **model.load_weights(filepath, by_name=False):**
  - 모델의 가중치를 **HDF5** 파일로부터 불러옴(**save_weights**에 의해 만들어짐)
  - 모델의 구조가 다를 때 불러오고자 한다면, **by_name=True**로 설정
  - 층의 이름에 맞게 파라미터가 불러옴 ▪ 모델의 구조는 변하지 않음

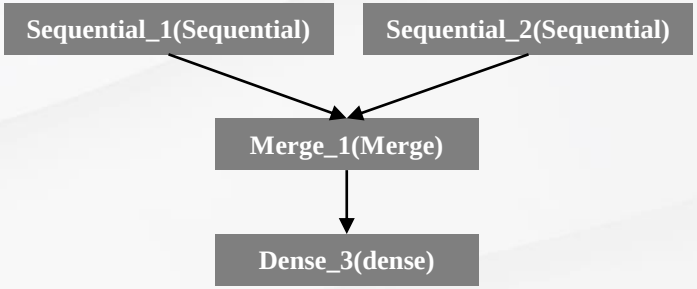
모델 종류: Sequential

- **Sequential** 모델은 층들의 스택(**stack**)으로 이뤄짐
- 이전 강의들에서 배운 컨볼루션/재현 신경망과 같은 모델
- 각 층은 그 다음 층에 입력을 제공해주는 객체의 역할 수행



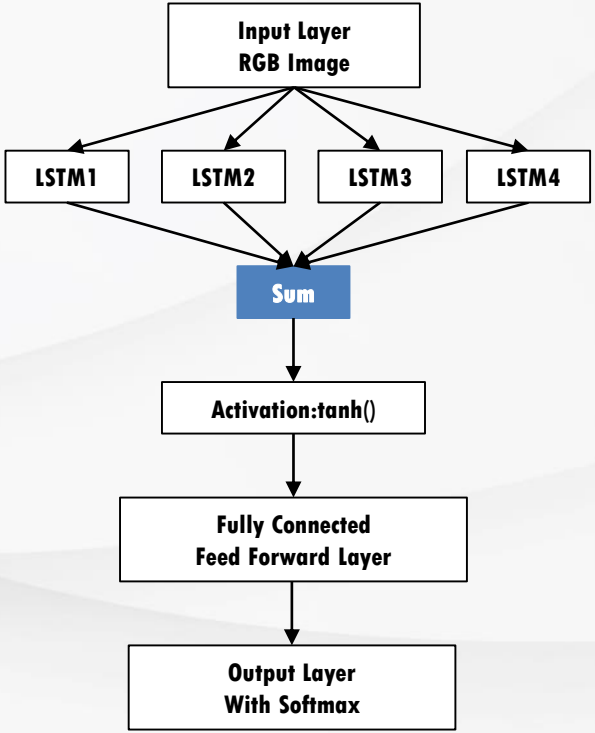
Sequential 모델의 예시

```
from keras.layers import Merge

left_branch = Sequential()
left_branch.add(Dense(32, input_dim=784))

right_branch = Sequential()
right_branch.add(Dense(32, input_dim=784))

merged = Merge([left_branch, right_branch], mode = ‘concat’)

final_model = Sequential()
final_model.add(merged)
final_model.add(Dense(10, activation=‘softmax’))
```



모델 종류: Graph

- **Graph** 모델에서 두 개 이상의 서로 다른 모델이 합쳐지거나 여러 모델로 나뉘어질 수 있음
- 다양한 개수의 입력과 출력 가능
- **Sequential** 모델과 다른 함수 제공



Graph 모델의 예시

```
def ranking_loss(y_true, y_pred):
    pos = y_pred[:,0]
    neg = y_pred[:,1]
    loss = -K.sigmoid(pos-neg) # use loss = K.maximum(1.0 + neg - pos, 0.0) if you want to use margin ranking loss
    return K.mean(loss) + 0 * y_true


def get_graph(num_users, num_items, latent_dim):

    model = Graph()
    model.add_input(name='user_input', input_shape=(num_users,))
    model.add_input(name='positive_item_input', input_shape=(num_items,))
    model.add_input(name='negative_item_input', input_shape=(num_items,))

    model.add_node(layer=Dense(latent_dim, input_shape = (num_users,)),
                   name='user_latent',
                   input='user_input')
    model.add_shared_node(layer=Dense(latent_dim, input_shape = (num_items,)),
                          name='item_latent',
                          inputs=['positive_item_input', 'negative_item_input'],
                          merge_mode=None,
                          outputs=['positive_item_latent', 'negative_item_latent'])

    model.add_node(layer=Activation('linear'), name='user_pos', inputs=['user_latent', 'positive_item_latent'], merge
    model.add_node(layer=Activation('linear'), name='user_neg', inputs=['user_latent', 'negative_item_latent'], merge

    model.add_output(name='triplet_loss_out', inputs=['user_pos', 'user_neg'])
    model.compile(loss={'triplet_loss_out': ranking_loss}, optimizer=Adam())#Adagrad(lr=0.1, epsilon=1e-06))

    return model
```

Keras의 장단점

장점

- 쉬운 모델 구현
- 다양한 함수 제공
- 손쉽게 커스터마이징 가능
- GPU를 활용한 빠른 계산
- 직관적인 함수 제공
- 활발한 커뮤니티

단점

- 생성 모델의 부족
- 고수준 라이브러리
- Theano 사용에 따른 에러 분석의 어려움

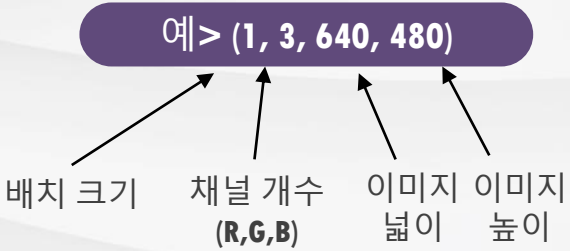
2. Keras로 컨볼루션 신경망 구현하기

VGG-16 모델 정의

- 모델 종류: **Sequential**
- 역할: 층들의 스택을 위한 빈 모델 생성

```
# build the VGG16 network
model = Sequential()
```

- **Sequential** 모델의 첫 층에서는 입력 데이터의 모양이 어떻게 생겼는지에 관한 정보를 알고 있어야 함
- 인수
  - **batch_input_shape:** 데이터 튜플의 모양



VGG-16 모델 정의

- 층 이름: **ZeroPadding2D**
- 층 종류: 컨볼루션 층
- 역할: **2D** 입력(예> 이미지)를 위한 제로 패딩 층
- 인수
  - 패딩: **int** 형식의 길이 **2** 또는 길이 **4** 튜플
  - **batch_input_shape**

```
model.add(ZeroPadding2D((1, 1),
                        batch_input_shape=(1, 3,
                        img_width, img_height)))
```

VGG-16 모델 정의

- 층 이름: **Convolution2D**
- 층 종류: 컨볼루션 층
- 역할: **2D** 입력에 대해 컨볼루션 계산
- 인수
  - **nb_filter:** 사용할 컨볼루션 필터 개수
  - **nb_row:** 컨볼루션 커널의 행 개수
  - **nb_col:** 컨볼루션 커널의 열 개수
  - 활성함수: 사용할 활성함수 이름, 입력하지 않을 시 선형 활성함수 사용 **(a(x) = x)**

```
model.add(Convolution2D(64, 3, 3,
                        activation='relu',name='conv1_1'))
```

VGG-16 모델 정의

- 층 이름: **MaxPooling2D**
- 층 종류: 컨볼루션 층
- 역할: 시계열 데이터에 대해서 최대 풀링 계산
- 인수
  - **pooling_length:** 최대 풀링을 적용할 지역 크기 설정
  - **stride:** 다운 스케일할 **int** 값 (예**> 2**는 입력의 크기를 절반으로 줄임)

```
model.add(MaxPooling2D((2, 2),
                       strides=(2, 2)))
```

학습정리

학습정리

Tacademy

지금까지 [Keras를 통한 딥러닝 구현 및 실습]에 대해서 살펴보았습니다.

Keras란?

Keras는 고수준의 다양한 모델 층, 활성함수, 목표함수 등을 제공하여 손쉽게 딥러닝 모델을 구현하게 해주는 라이브러리
사용할 입력/출력 데이터를 모델의 처음과 마지막 층에 설정하고 가운데 층을 목적에 맞게 선택가능

Keras로 컨볼루션 신경망 구현하기

Sequential 모델의 ZeroPadding2D / Convolution2D / MaxPooling2D 함수를 사용하여 VGG-16 모델 구현

컨볼루션 신경망이 보는 세상의 모습

컨볼루션 신경망의 컨볼루션층들은
직선/색 등의 저차원부터 도형 등의 고차원 특징 추출