

컴퓨터 공학 정리 노트


청년 AI · Big Data 아카데미
10기 교육생 B4 이정하

목 차

- ① 컴퓨터 공학의 역사
- ② AI, ML, DL 의 특징과 구분
- ③ 디지털 트랜스포메이션
- ④ 컴퓨팅 사고력과 문제해결
- ⑤ 객체지향 프로그래밍과 ‘클래스’
- ⑥ 파이썬 문법

[부록] 컴퓨터 공학 키워드 사전

① 컴퓨터 공학의 역사 [부록] 키워드 참고

- 
- 1930년대-1940년대 : 컴퓨터 공학의 아버지, 앨런 튜링의 튜링 기계 발명
 - 1950년대-1960년대: **아스키 코드**로 체계 표준화, 초기 컴퓨터 제작
 - 1970년대: **운영체제**와 **C 언어** 등장, IBM, Microsoft 회사 설립
 - 1980년대: 기존의 **TUI** 방식의 한계로 인해 **GUI**의 필요성 대두
 - 1990년대: 개인 PC에서도 서버의 역할을 할 수 있는 **오픈 소스** 등장
 - 2000년대: 모바일의 등장, 구글의 **안드로이드** 개발과 앱 활성화
 - 2010년대~: 데이터의 증가와 하드웨어의 발전으로 **클라우드 사업** 확장
인공지능(머신러닝, 딥러닝) 데이터 학습을 위한 **GPU** 등장
딥러닝 프레임워크(텐서플로우, 파이토치, 케라스) 개발

② AI, ML, DL 의 특징과 구분

인공지능 [AI]

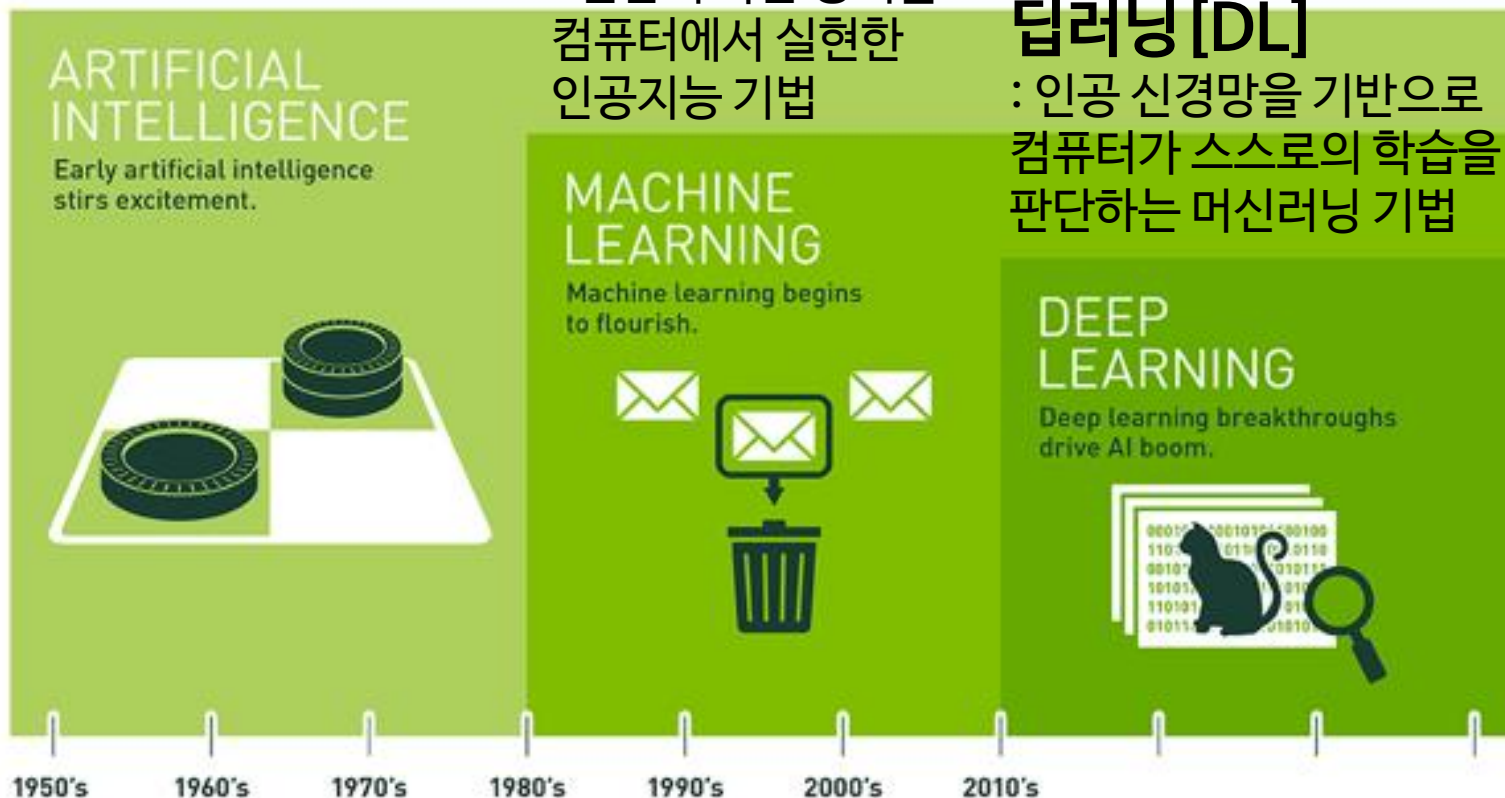
: 컴퓨터 프로그램으로
인간의 지능을 구현한 기술

머신러닝 [ML]

: 인간의 학습 능력을
컴퓨터에서 실현한
인공지능 기법

딥러닝 [DL]

: 인공 신경망을 기반으로
컴퓨터가 스스로의 학습을 통해
판단하는 머신러닝 기법



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

- 머신러닝과 딥러닝, 정확히 무엇이 다를까?



다음 데이터 셋을 보고 어떤 사진이
아기 고양이인지 아이스크림인지
구분하고자 할 때,

머신러닝은 사람이 각각의 특징을
직접 추출해 학습을 시켜야 하는 반면,

딥러닝은 인공 신경망을 통해
자동으로 데이터의 특징을 학습한다.

따라서 딥러닝은 머신러닝과 달리
학습에서 사람의 개입을 필요로 하지 않는다.

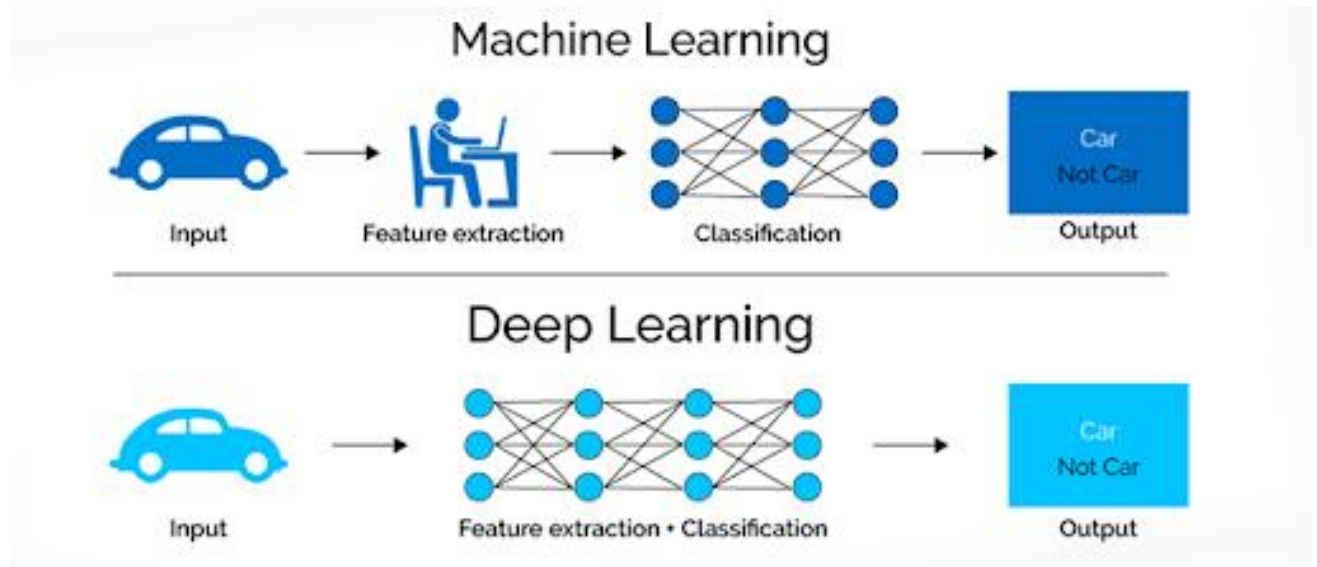
그리고 특징을 자동으로 추출하는 딥러닝이
머신러닝보다 훨씬 정확한 인식률을 자랑한다.

- 머신러닝 VS 딥러닝 구분하기

	머신러닝 Machine Learning	딥러닝 Deep Learning
특징 추출과 학습	사람이 한다	데이터를 이용해 자동으로 한다

ML

DL



③ 디지털 트랜스포메이션

: 디지털 기술을 사회 전반에 적용해서 기존의 사회를 혁신하는 것

표준화

아스키 코드 (ASCII)

미국에서 표준화한 정보교환용 7비트 부호 체계.
총 128개의 숫자가 문자에 할당되어 있다.

eg) 영화 [마션]에서는 화성에 조난 당한 주인공이
지구와 소통하기 위해 아스키 코드를 사용한다



유니코드 (Unicode)

전 세계의 모든 문자를 컴퓨터에서
일관되게 다룰 수 있도록 만들어진 산업 표준.

= 표준화 코드는 컴퓨터에서 모든 활동이 가능하게 함으로써
디지털 트랜스포메이션의 시작을 열었다.

- 일상 속의 디지털 트랜스포메이션 사례



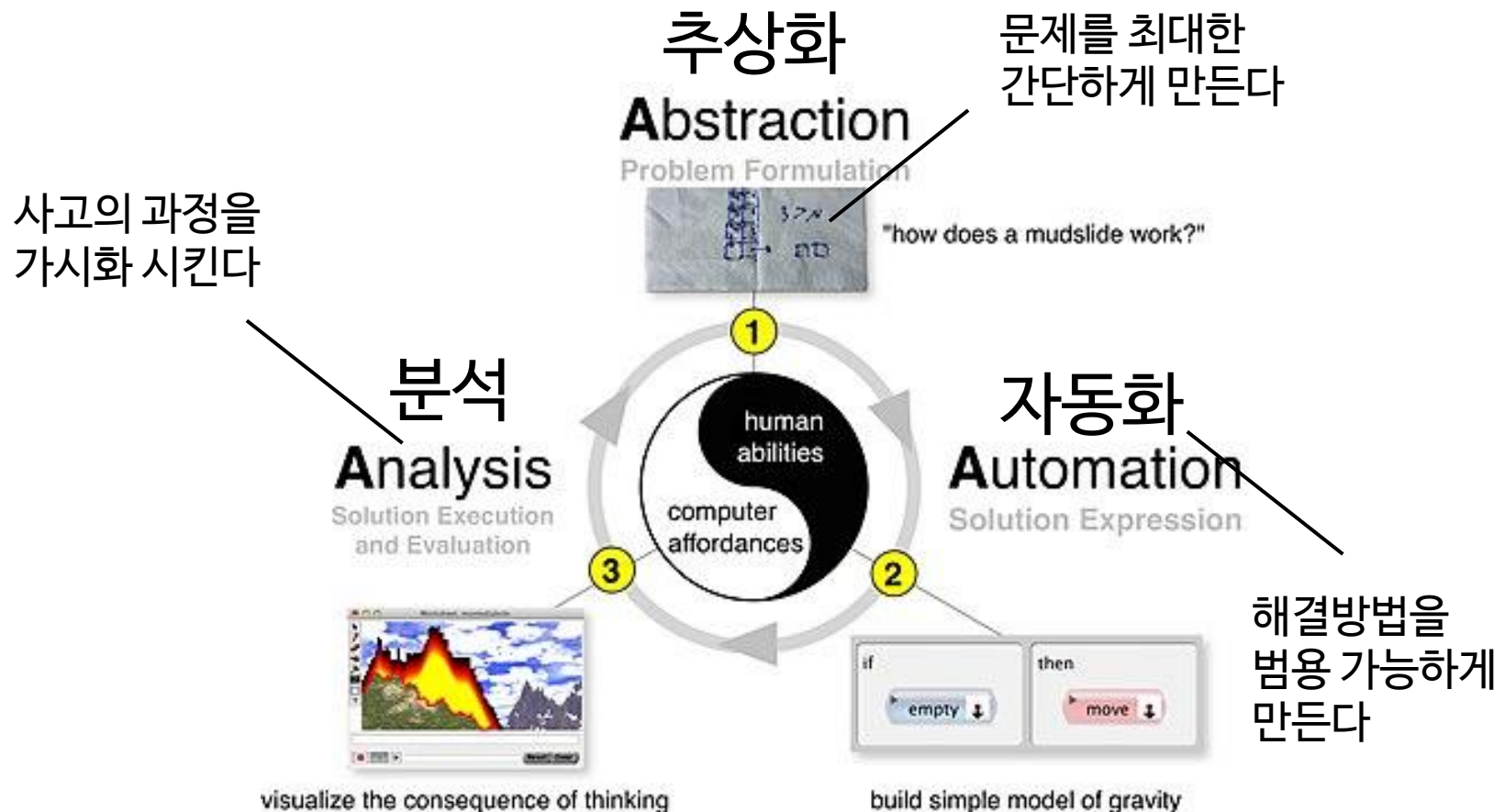
전기차 회사 ‘테슬라’의 자율주행 기술이 완벽히 구현되면 자동차는 단순한 이동수단이 아니라 주거 공간의 개념으로 확장이 가능

‘스타벅스’의 사이렌 오더와 드라이브 스루의 차량 진입 자동 인식 서비스는 이용자의 편의성을 극대화, 기존의 판매 방식에서 벗어난 새로운 방식을 제안함



④ 컴퓨팅 사고력 (CT)과 문제 해결

: 컴퓨터가 문제를 해결하는 방식으로 복잡한 문제를 해결하는 능력



C
T

자료구조

Data Structure

방대한 양의 데이터를 효율적으로
관리할 수 있는 데이터의 구조



추상화

Abstraction

실제 세계의 문제를
해결 가능한 형태로 표현

+

자동화

Automation

추상화 과정에서 만들어진 해결 모델을
컴퓨터가 이해할 수 있게 만드는 과정



알고리즘

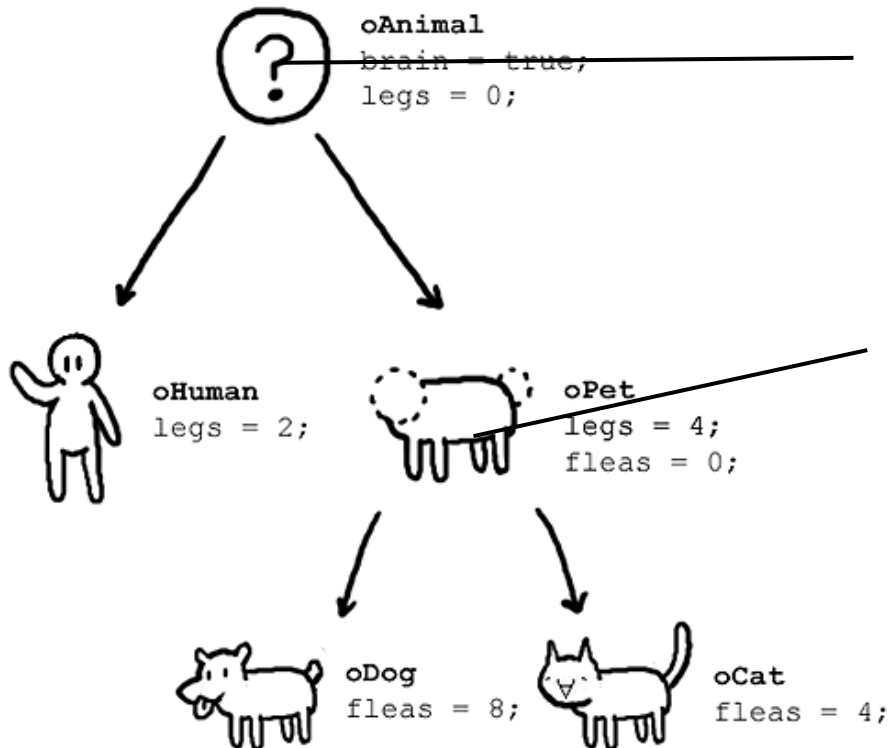
문제를 해결하는 방법

Algorithm

→ CT 를 통해서 현실의 문제를 효과적으로 해결 가능

⑤ 객체지향 프로그래밍과 ‘클래스’

: 일상의 문제를 컴퓨터가 사람의 시각에서 사물을 바라보는 관점으로 설계하는 프로그래밍 기법. 프로그램을 객체(Object)의 모임으로 파악한다.



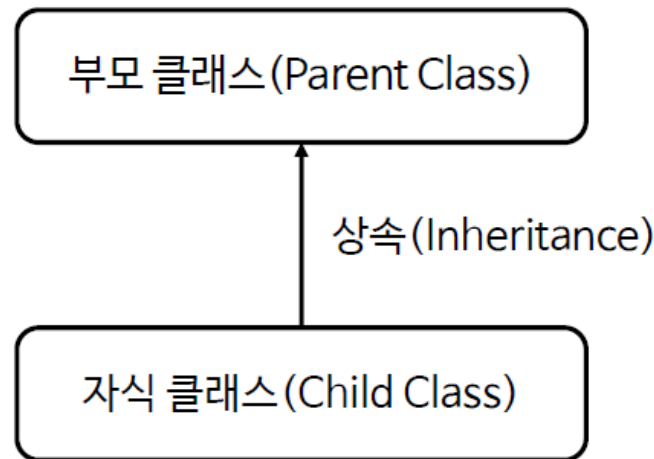
클래스는 객체를 생성하는
자료형을 결정하는 역할을 한다.

클래스는 서로 다른 자료형을
하나로 묶어서 선언해준다.

그리고 그 안에서 특정 객체가 생성되면
그 객체를 ‘인스턴스’라고 부른다.

- 클래스 상속(Class Inheritance)

: 기존 클래스의 메서드를 그대로 물려받는 새로운 클래스를 만드는 것



자식 클래스가 부모 클래스의 내용 상속

←

```
class 자식 클래스(부모 클래스):  
    // 이 부분에 자식 클래스의 내용을 코딩
```

- 메서드 오버라이딩 (Method Overriding)

: 상속 관계에서 상위 클래스의 메서드를 하위 클래스에서 재정의

```
class Car:
    speed = 0
    def upSpeed(self, value):    # upSpeed 함수 정의(부모 클래스)
        self.speed += value

class Sedan(Car):
    def upSpeed(self, value):    # upSpeed 함수 동일한 함수명으로 재정의 (자식 클래스)
        self.speed += value
        if self.speed > 150:
            self.speed = 150
```

메서드의 이름을 동일하게 지정,
하위 클래스에서 이름만 빌리고 재정의하며

다형성 (Polymorphism, 폴리모피즘) 을 지원

상속 관계 간의 다른 클래스의 인스턴스가 같은 함수 호출에 대해 각각 다르게 반응,
적은 코딩으로 다양한 객체들에게 유사한 작업을 수행할 수 있도록 해준다.

- 연산자 오버로딩 (Operator Overloading)

: 연산자 중복을 위해 두 개의 밑줄 문자가 앞 뒤로 있는
메서드를 클래스 내부에 미리 정의하는 것 ex) `__abs__(self)`

메서드(Method)	연산자(Operator)	사용 예
<code>__add__(self, other)</code>	<code>+</code> (이항)	<code>A + B</code> , <code>A += B</code>
<code>__pos__(self)</code>	<code>+</code> (단항)	<code>+A</code>
<code>__sub__(self, other)</code>	<code>-</code> (이항)	<code>A - B</code> , <code>A -= B</code>
<code>__neg__(self)</code>	<code>-</code> (단항)	<code>-A</code>
<code>__mul__(self, other)</code>	<code>*</code>	<code>A * B</code> , <code>A *= B</code>
<code>__truediv__(self, other)</code>	<code>/</code>	<code>A / B</code> , <code>A /= B</code>
<code>__floordiv__(self, other)</code>	<code>//</code>	<code>A // B</code> , <code>A //= B</code>
<code>__mod__(self, other)</code>	<code>%</code>	<code>A % B</code> , <code>A %= B</code>
<code>__pow__(self, other)</code>	<code>pow()</code> , <code>**</code>	<code>pow(A, B)</code> , <code>A ** B</code>
<code>__lshift__(self, other)</code>	<code><<</code>	<code>A << B</code> , <code>A <<= B</code>
<code>__rshift__(self, other)</code>	<code>>></code>	<code>A >> B</code> , <code>A >>= B</code>
<code>__and__(self, other)</code>	<code>&</code>	<code>A & B</code> , <code>A &= B</code>
<code>__xor__(self, other)</code>	<code>^</code>	<code>A ^ B</code> , <code>A ^= B</code>
<code>__or__(self, other)</code>	<code> </code>	<code>A B</code> , <code>A = B</code>
<code>__invert__(self)</code>	<code>~</code>	<code>~A</code>
<code>__abs__(self)</code>	<code>abs()</code>	<code>abs(A)</code>

클래스를 사용해 객체끼리 수치 연산을 하기 위해서는

연산자 오버로딩을 통해 연산자를 다시 재정의 해야 한다

+ 피 연산자의 순서가 바뀐 경우 연산자 앞에 r을 붙여 표현 ex) `__radd__`

[부록] 컴퓨터 공학 키워드 사전

표준화 코드

연관어: ASCII, 표준 정보 교환 코드

아스키 코드

7비트의 이진수 조합으로 이루어진 부호이다. 이를 통해 숫자와 로마 글자, 기호를 표준화해 나타낼 수 있으며 표준화 체계를 확립하는데 기여했다.

소프트웨어

연관어: 컴퓨터 사용자와 소통, 시스템 소프트웨어

운영 체제

시스템 하드웨어를 관리하고 응용 소프트웨어를 실행하기 위해 하드웨어 추상화 플랫폼과 공통 시스템 서비스를 제공하는 시스템 소프트웨어

운영 체제

연관어: 시분할 운영체제(Time-Sharing OS)

유닉스

현대의 컴퓨터 운영 체제의 원형이 된 운영 체제. 리눅스, 안드로이드, mac OS, iOS 등 많은 운영 체제의 뿌리이다.

프로그래밍 언어

연관어: Structured Programming, 분할 정복법, C++

C 언어

분할정복법(Divide and Conquer)에 의해 만들어진 운영체제와 하드웨어를 컨트롤 할 수 있는 모든 문법이 포함되어있는 언어. 컴퓨터 관점으로 디자인 되었다.

인터페이스

연관어: Text User Interface

TUI

텍스트 유저 인터페이스. 텍스트로만 이루어져있어 프로그램을 실행시키기 위해서는 파일명과 명령어를 알고 있어야 하는 인터페이스이다.

인터페이스

연관어: Graphic User Interface

GUI

그래픽 유저 인터페이스. 그래픽 화면으로 기존의 TUI 방식보다 보기 편하게 구현되었다. 모니터로 사용자가 아이콘 같은 그래픽을 컨트롤해 프로그램을 실행시킨다.

프로그래밍 언어

연관어: 객체 지향 프로그래밍

C++

기존의 C 언어에 여러 가지 기능을 추가하여 만든 프로그래밍 언어. 객체 지향 프로그래밍의 특징과 C 언어의 절차 지향적 프로그래밍의 특징을 모두 가졌다.

프로그램 설계

연관어: C++, 클래스

OOP

객체 지향 프로그래밍(Object Oriented Programming). 프로그램을 수많은 '객체'로 나누고, 그 객체들의 상호작용을 프로그램으로 보는 설계 방식이다.

소프트웨어

연관어: 분산형 네트워크

오픈 소스

소스 코드가 공개된 소프트웨어.
개인 PC에서도 서버의 역할을 할 수 있는 운영 체제가
필요해짐과 분산형 네트워크가 발전하며 등장했다.

네트워크

연관어: 오픈 소스

분산형 네트워크

1대의 메인 컴퓨터가 아니라
복수의 개별 장치에 의해 실행되는
정보 통신 네트워크이다.

운영 체제

연관어: 모바일

안드로이드

리눅스 커널 기반의 휴대용 장치를 위한 운영 체제. 구글에서 만들었고, 무료이기 때문에 세계에서 가장 많은 사용자를 보유하고 있다.

인터넷 컴퓨팅

연관어: 즉시 제공(on-demand), 가상화

클라우드

컴퓨터 통신망이 복잡한 내부에 대한 지식이 없어도 클라우드 내의 컴퓨터 자원으로 자신이 원하는 작업을 할 수 있음을 뜻한다.

인공지능

연관어: 기계 학습

머신러닝

인간의 학습 능력과 같은 기능을 컴퓨터에서 실현하고자 하는 기술이다. 컴퓨터에 인간이 데이터를 부여해 학습하게 하고, 이를 통해 새로운 지식을 얻어낸다.

인공지능

연관어: 신경망, GPU

딥러닝

머신러닝의 일종으로, 은닉층을 갖는 인공신경망을 이용해 많은 데이터로부터 특징을 자동으로 직접 학습한다. 딥러닝에서 대량의 데이터를 처리할 때에는 GPU를 통해 처리한다.

인공지능

연관어: 머신러닝, Supervised Learning

지도학습

문제와 문제에 대한 정답을 모두 input 하여 감독 하에 학습을 시키는 머신러닝 기법이다.

인공지능

연관어: 머신러닝, Unsupervised Learning

비지도학습

문제에 대한 정답을 알려주지 않고 문제만 input 해 감독을 배제한 학습을 시키는 머신러닝 기법이다.

인공지능

연관어: 머신러닝, 에이전트, Reinforcement Learning

강화학습

기계 학습 중에 컴퓨터가 주어진 상태에서 상과 벌을 통해 최적의 행동을 선택하는 머신러닝의 기법.
에이전트는 자신이 취할 행동을 표현하는 정책을 수립해 최대의 보상을 받도록 노력한다.

인공지능

연관어: Natural Language Processing(NLP)

자연어 처리

컴퓨터를 이용해 자연어를 처리하고 분석하는 인공지능 기술의 하나. 정보검색, 번역, 질의응답 등에 사용된다.

인공지능

연관어: Computer Vision(CV), Object Detection, Object Tracking

컴퓨터 비전

사람의 눈처럼 이미지를 분석하는 기술.
Detection은 이미지나 동영상에서 객체를 찾고,
Tracking은 객체를 쫓는 것을 의미한다.

프로세서

연관어: Graphics Processing Unit

GPU

그래픽 처리를 위한 고성능의 프로세서이다. GPU는 빠른 속도로 컴퓨터가 데이터를 학습할 수 있도록 도와주기 때문에 인공지능이 빠르게 발전할 수 있었던 핵심이 되었다.

이정하