

# 인공지능을 활용한 소재 연구

**양진훈**

(e-mail: [jhyang00@kriict.re.kr](mailto:jhyang00@kriict.re.kr) / [jk105214@gmail.com](mailto:jk105214@gmail.com))

한국화학연구원  
화학플랫폼연구본부  
화학데이터기반연구센터



## 양진훈 Jin-Hoon Yang

**B.S. 한국기술교육대학교 에너지신소재화학공학부 (2012)**

**M.S. 한국기술교육대학교 에너지신소재화학공학부 (2014)**

- DFT 계산을 통한 기능성 소재 연구; 연료전지 전해질 및 반도체 소재

**Ph.D. 광주과학기술원 신소재공학부 (2022)**

- Mg 가넷 계열 이차전지 양극재 연구
- ~~자연어처리모델을 활용한 차세대 전자 소재 연구~~
- ~~유기결정구조체 계산을 위한 MLP 활용 연구~~
- 탄소나노구조체 계산 연구

**한국화학연구원 화학데이터기반연구센터 (2022.03 ~ )**

- 인공지능을 활용한 소재 개발 및 소재 데이터 플랫폼 개발

# 일정 및 참고문헌

시간	내용
10:00 – 10:30	머신러닝 이론
10:30 – 12:00	머신러닝 실습 (데이터 다루기, scikit-learn을 활용한 머신러닝 모델링 기초)
13:00 – 14:00	인공신경망 이론 및 활용 사례
14:00 – 16:00	인공신경망 실습 (pytorch, graph neural network, machine learning potential)

O'REILLY®

## Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow

한즈온 머신러닝 [2판]

사이킷런, 케라스, 텐서플로 2를 활용한  
머신러닝, 딥러닝 완벽 실무



텐서플로 2 활용  
전반 가이드

## 유용한 사이트

<https://tensorflow.blog/handson-ml2/> # Hans on ML 1,2장

<https://www.codecademy.com/> # Python 기초

<https://google.com/> # 모르는 것은 google에

<https://chat.openai.com/> # 간단한 코드는 ChatGPT로

<https://jehyunlee.github.io/> # KRIE 이제현 박사님 블로그

## 강의자료

[https://github.com/git-jhyang/handson\\_gnn](https://github.com/git-jhyang/handson_gnn)

# 환경 구축

WSL을 이용한 Linux 하위 시스템

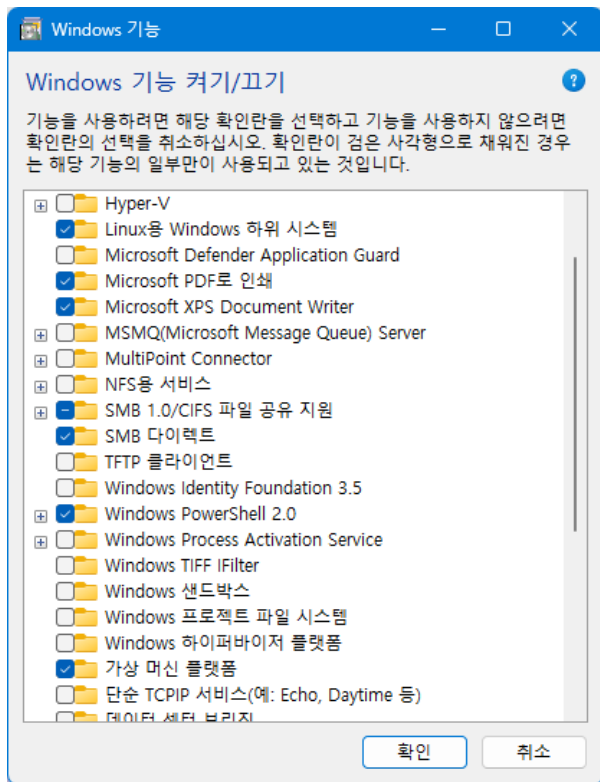
개발환경 Anaconda / VSCode / Github / Python

# 환경 구축 - WSL

## Windows 기능 켜기/끄기

- Linux용 Windows 하위 시스템
- 가상 머신 플랫폼

위 두 기능 체크 후 재부팅



## CMD 설정 및 Ubuntu 설치 (Windows side)

```
> wsl --set-default-version 2 # wsl 2로 변경
> wsl --list --online # 설치 가능한 배포 확인
> wsl --install Ubuntu-22.04 # Ubuntu 22.04 설치
> wsl # wsl 접속
```

```
C:\Users\Yang>wsl
jhyang@DESKTOP-S2P7L9Q:/mnt/c/Users/Yang$ |
```

## 기본 설정 및 프로그램 설치 (WSL / Server side)

```
$ sudo passwd root
```

```
$ ln -s [SOURCE] [DESTINATION]
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install vim git
```

<https://learn.microsoft.com/ko-kr/windows/wsl/install-manual>

<https://velog.io/@w00j00ng351/wsl-기능-활성화>

# 환경 구축 – Anaconda and VSCode

## Anaconda 설치 & 세팅 (WSL / Server side)

```
$ apt-get install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1  
libxcomposite1 libasound2 libxi6 libxtst6
```

```
$ curl -O https://repo.anaconda.com/archive/Anaconda3-2023.09-0-Linux-x86_64.sh
```

```
$ sh Anaconda3-2023.09-0-Linux-x86_64.sh
```

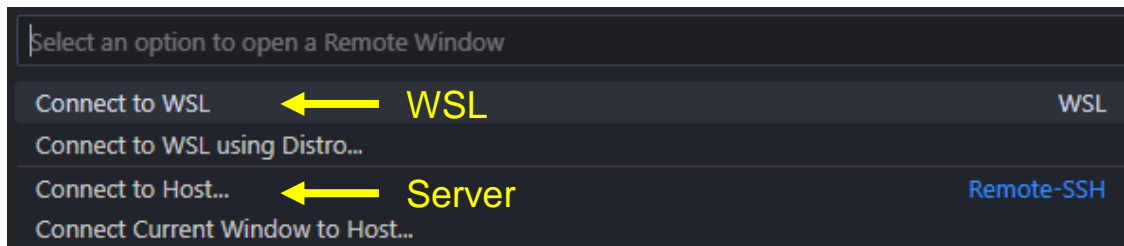
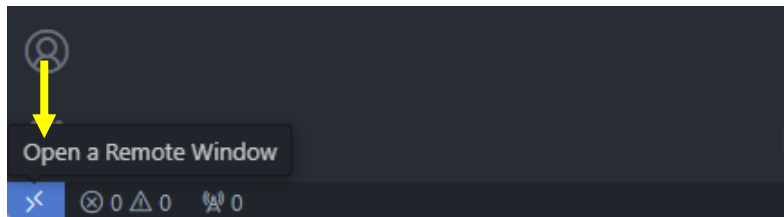
```
$ conda init
```

```
$ conda create -n gnn python=3.9
```

```
$ conda activate gnn
```

## VSCode 설치 & 세팅 (Windows side)

Windows에서 다운로드 및 설치 후 WSL 접속



<https://docs.anaconda.com/free/anaconda/install/linux/>  
<https://docs.anaconda.com/free/anaconda/release-notes/>  
<https://code.visualstudio.com/>

# 환경 구축 – Github and Python

## Github 다운로드

폴더에 들어간 후

```
$ git clone https://github.com/git-jhyang/handson_gnn.git
```

에러가 난다면

```
$ git config --global http.sslVerify false
```

## 패키지 설치

```
$ conda activate gnn
```

```
$ conda install --file requirements.txt
```

또는

```
$ conda install matplotlib numpy pandas scikit-learn ....
```

에러가 난다면

```
$ conda config --set ssl_verify false
```

## github, gitlab 등 적극 활용

개발을 통한 연구를 수행하는데 있어 백업, 업데이트 기록 등을 쉽게 남길 수 있어  
작업물 관리가 편리함

이후 본인의 작업물 및 성실함을 보여줄 수 있는 척도가 될 수 있음

Machine learning

# 기계학습 기초

인공지능

인공지능을 활용한 문제 해결

기계학습 방법론

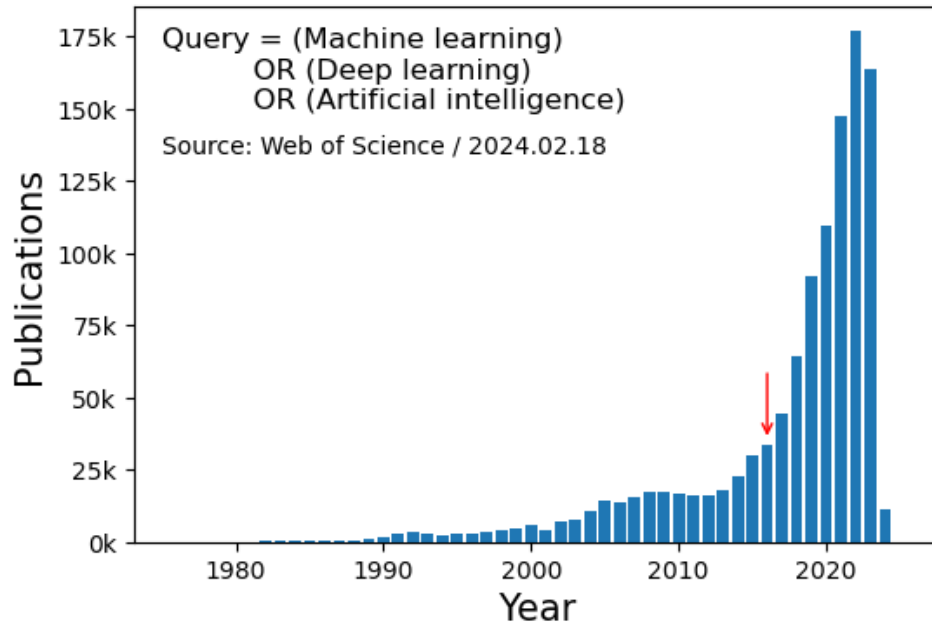
데이터



# 인공지능

- 인공지능을 활용한 연구

인공지능 논문 출판 경향



2016.03 / 알파고와 이세돌의 대국



- 강력한 인공지능 모델들



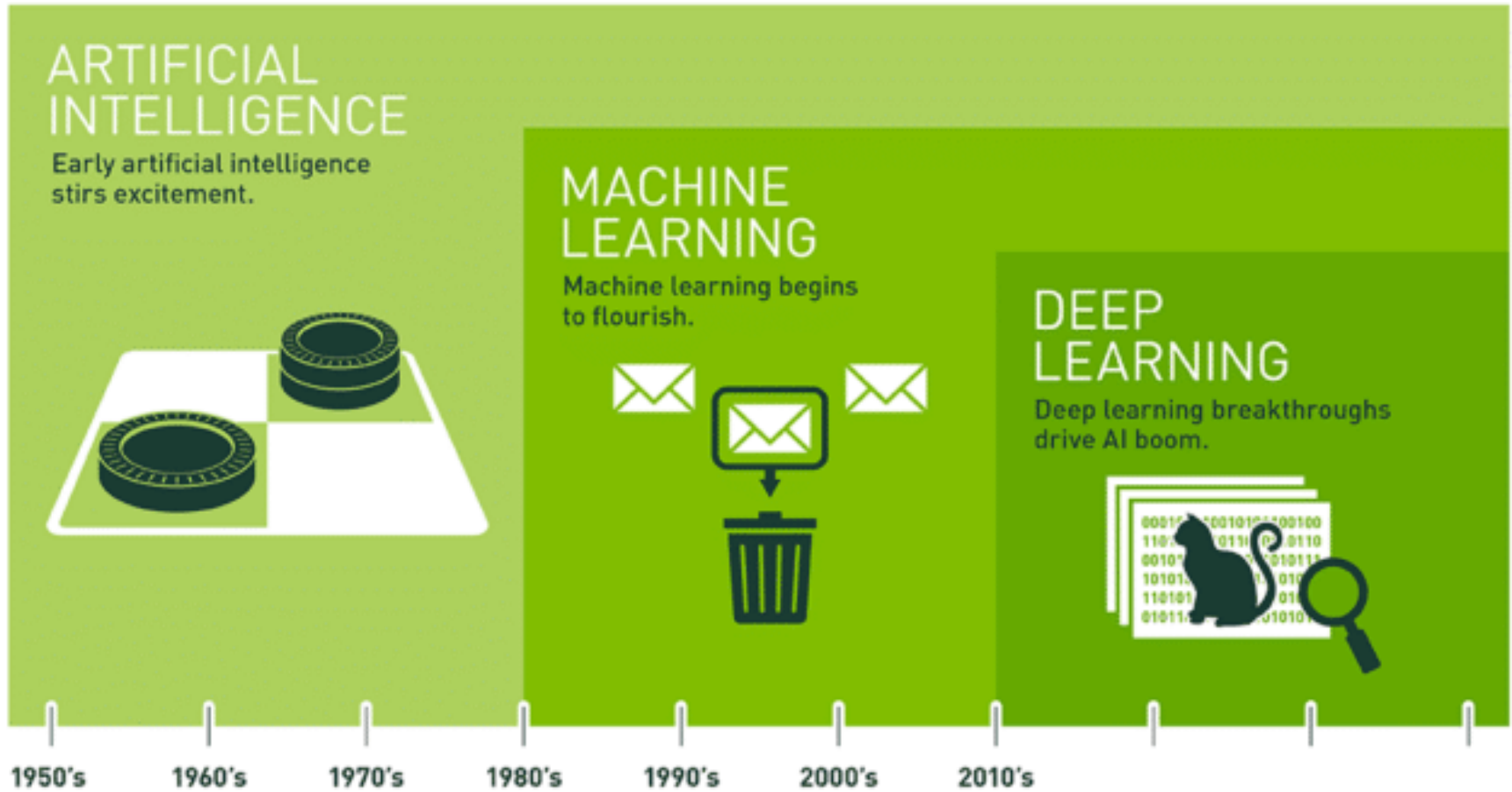
AlphaZero, AlphaFold, GNoME, ...



ChatGPT, DALLE, Sora, ...

중앙일보: <https://www.joongang.co.kr/article/2507487>

# 인공지능



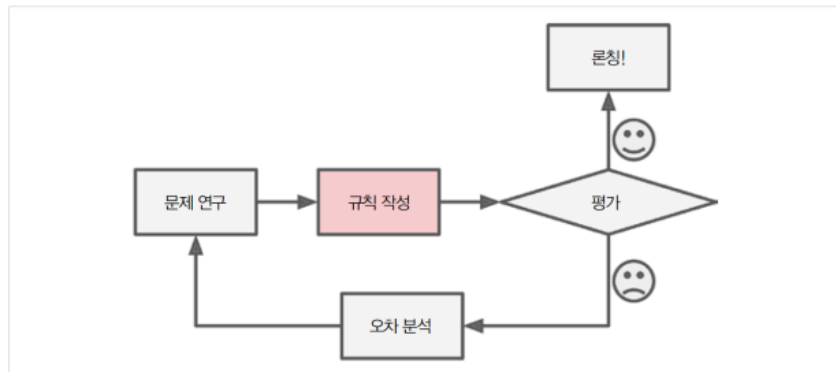
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

“명시적인 프로그래밍 없이 컴퓨터가 학습하는 능력을 갖추게 하는 연구분야” Arthur Samuel , 1959  
"the field of study that gives computers the ability to learn without being explicitly programmed."

<https://blogs.nvidia.com/blog/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

# 기계학습 모델을 활용한 문제 해결

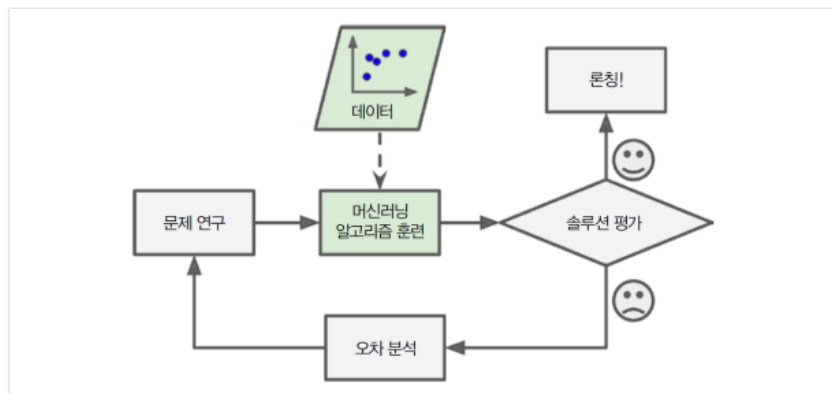
그림 1-1 전통적인 접근 방법



**적절한 규칙을 찾기 위한 수많은 실험**

전통적인 trial & error 방식을 통한 현상 해석  
많은 시간과 비용을 투자해야 함

그림 1-2 머신러닝 접근 방법

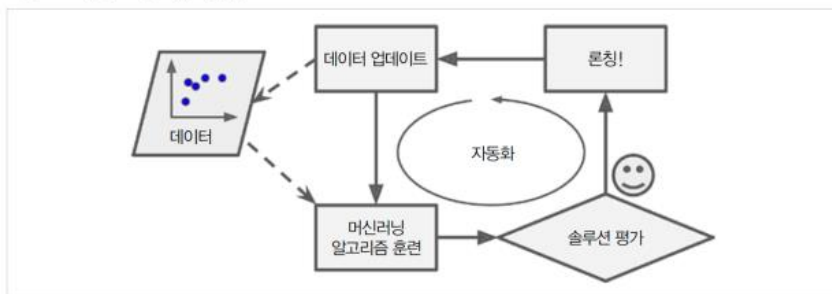


**기계학습 모델을 활용한 개발 과정 가속**

데이터에 숨겨진 패턴을 찾아 데이터를  
나타낼 수 있는 모델을 학습

*데이터의 확률 분포를 학습*

그림 1-3 자동으로 변화에 적응함

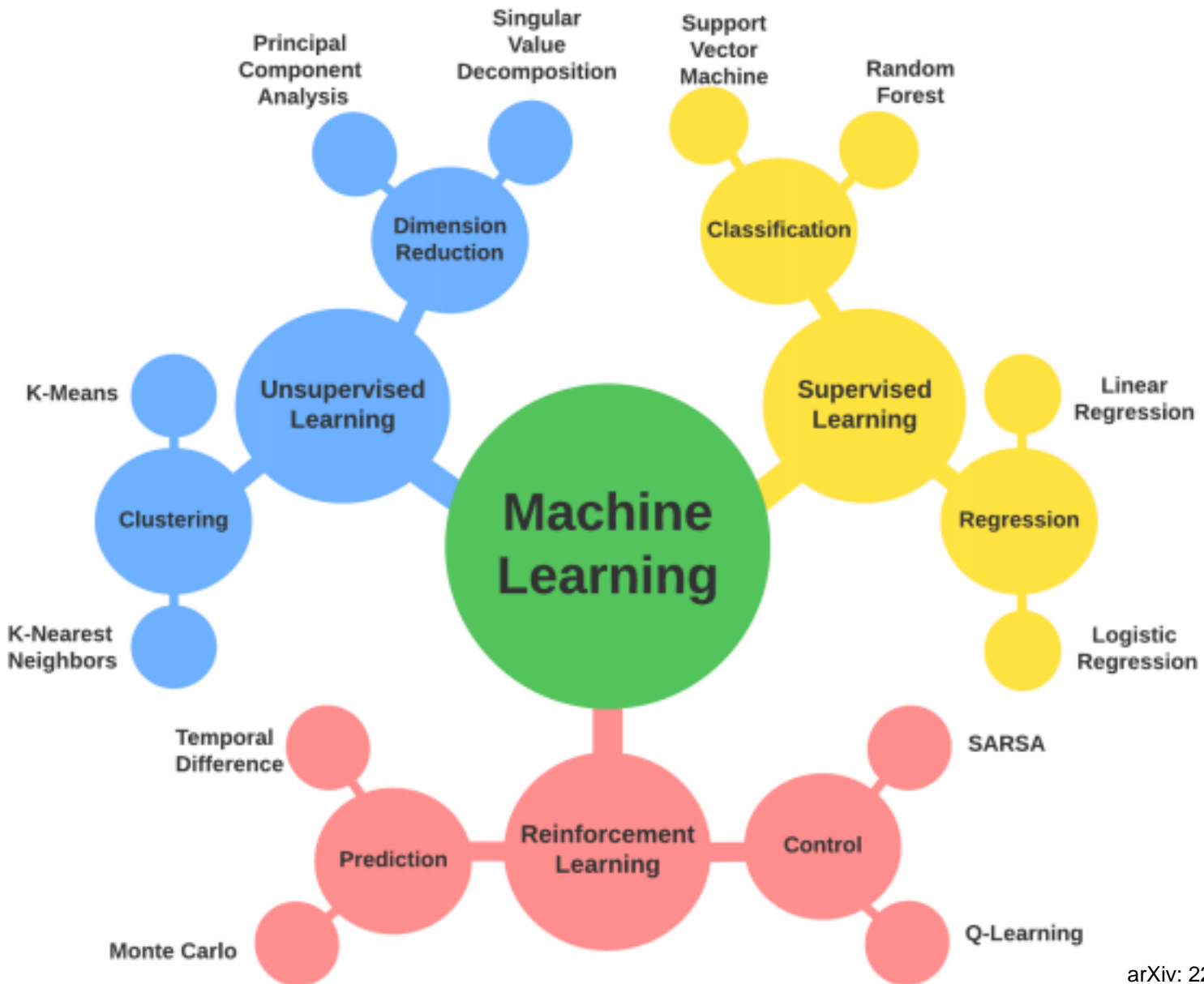


**Active learning을 통한 closed-loop 완성**

기계학습으로 예측된 target에 대해,

- (1) 실험을 통해 새로운 데이터를 생성하고
- (2) 생성된 데이터를 다시 모델에 학습시켜 모델을 지속적으로 업데이트

# 기계학습 (Machine learning, ML)



arXiv: 2204.09579

# Clustering & Dimension reduction

그림 1-8 군집

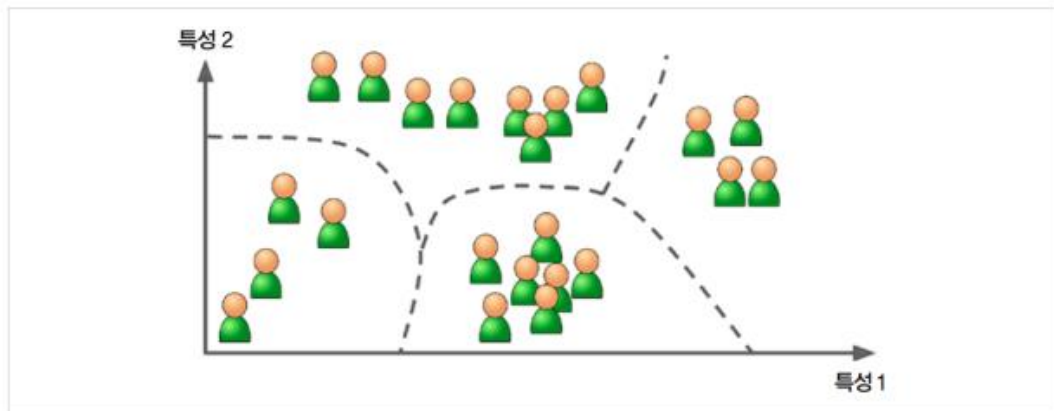
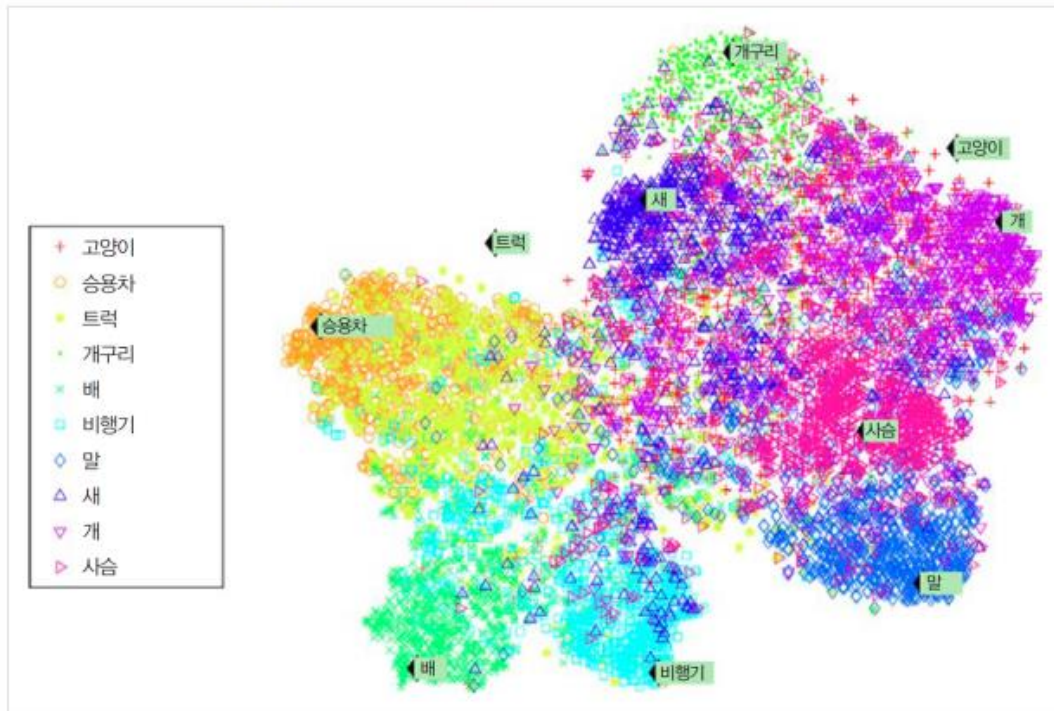


그림 1-9 의미 있는 군집을 강조한 t-SNE 시각화의 예<sup>9</sup>



## 군집화 Clustering

데이터간의 유사도를 측정하여  
비슷한 데이터를 묶는 방법

*이상치 탐지, 데이터 분포 파악*

*K-means, DBSCAN*

## 차원 축소 Dimension reduction

데이터의 차원을 축소하여 무의미  
한 데이터를 제거하고 시각화를 가  
능케함

*이상치 탐지, 데이터 압축*

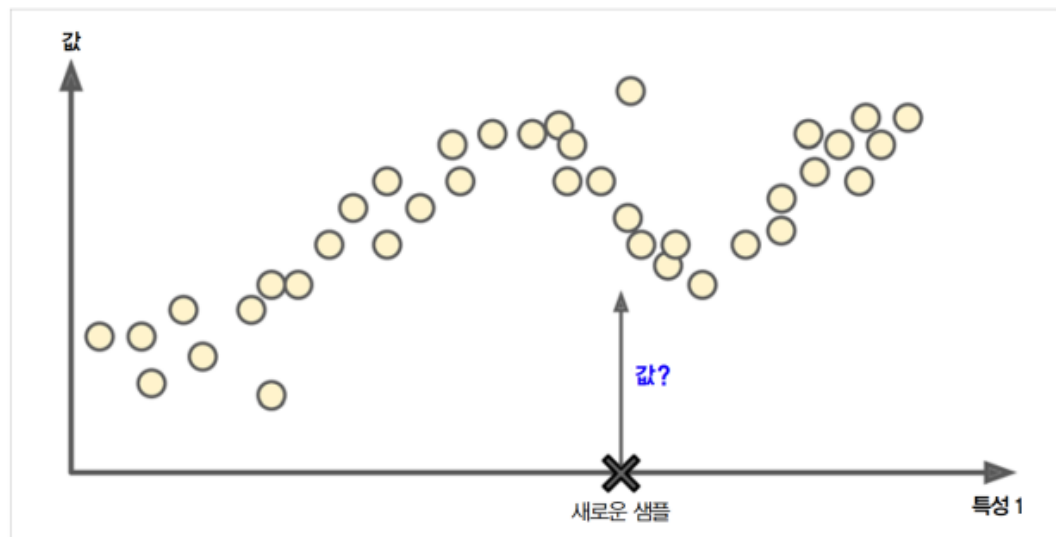
*PCA, t-SNE, U-MAP*

# Classification & Regression

그림 1-5 지도 학습에서 레이블된 훈련 세트(예를 들면 스팸 분류)



그림 1-6 회귀



## 분류 Classification

주어진 데이터를 특정 class에 할당 / 클래스에 대한 확률로 표현

예) 스팸일 확률 75%

적절한 cutoff를 정해야 함

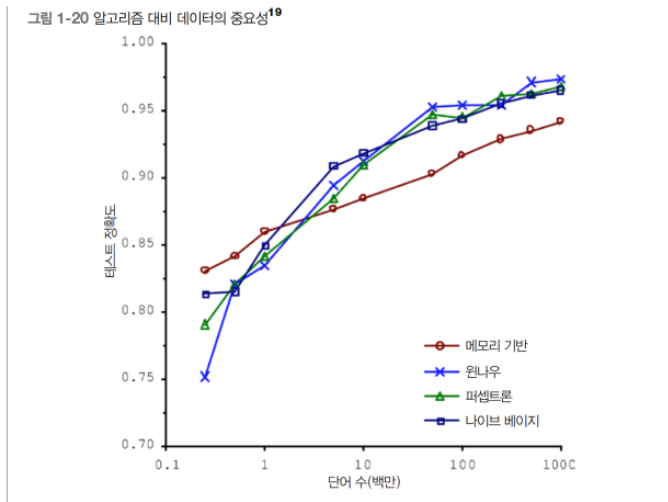
## 회귀 Regression

주어진 데이터의 확률분포를 기반으로, 새로운 데이터의 값을 예측

예) 내일 낮 온도

Machine learning: Gaussian processor, XGBoost, Support vector machine, ...  
Deep learning: Neural network (large parameters)

# 데이터의 중요성



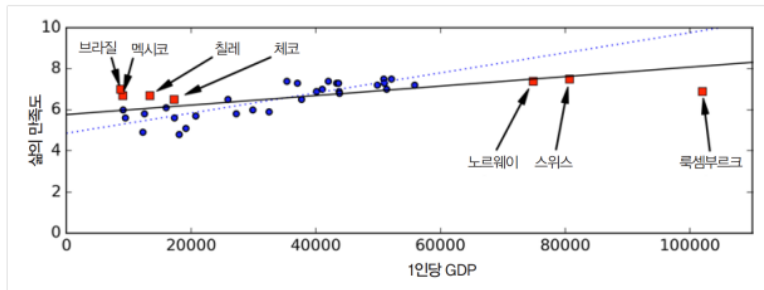
## 알고리즘 vs 데이터

양질의 많은 데이터가 주어진다면, 알고리즘에 관계 없이 높은 성능을 낼 수 있음

하지만, 데이터 확보에는 한계가 있으므로 결국 알고리즘 또한 중요함

*고품질의 많은 데이터를 확보하는 것이 가장 중요함*

그림 1-21 대표성이 더 큰 훈련 샘플



## 일반화

훈련 데이터는 실제로 예측하려는 새로운 데이터들의 분포를 대표해야 함

좌측 그림에서 **훈련 데이터 (파란색)**는 **새로운 데이터 (붉은색)**를 대표하지 못했기 때문에 큰 오차를 발생시킴.

## 낮은 품질의 데이터

특성 (feature)이 누락된 데이터나, 특성 혹은 레이블 (label)에 노이즈가 큰 경우

## 무의미한 특성

목표로 하는 모델 성능과 필요 없는 특성



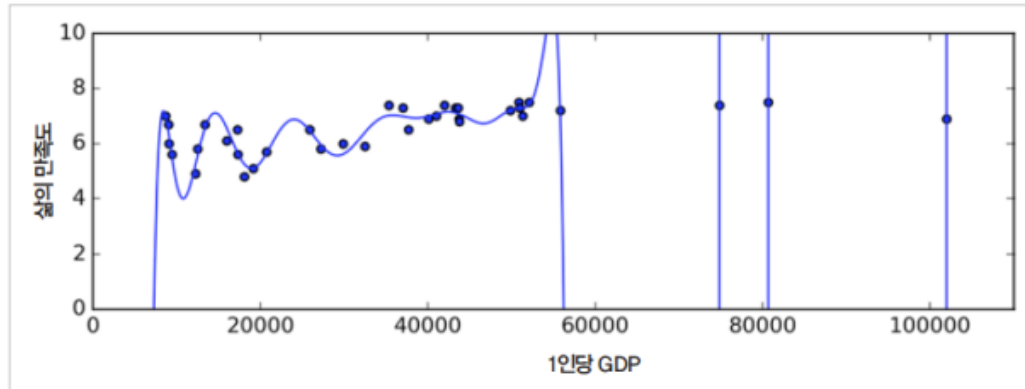
# 알고리즘의 중요성

## 과대적합 Overfitting

모델이 데이터의 세부적인 부분을 학습하여, 새로운 데이터에 대해 틀린 예측을 하는 경우

*일반적으로 문제의 복잡도보다 모델의 파라미터 수가 더 많을 때 발생하며, 단순한 모델을 사용하거나 특성 수를 줄여 해결할 수 있음*

그림 1-22 훈련 데이터에 과대적합



## 과소적합 Underfitting

모델이 단순하여 데이터를 충분히 학습하지 못하는 경우  
*보다 복잡한 모델을 사용하거나 특성을 추가하여 해결할 수 있음*



# 테스트와 검증

## 데이터 분할

모델 배포에 앞서, 모델이 잘 훈련되었는지 평가하기 위해선 데이터가 필요함.

훈련 데이터를 미리 나누어 학습과 테스트에 활용. 일반적으로 훈련 8: 테스트 2

가정) 훈련 데이터와 테스트 데이터, 그리고 실제 외부 데이터는 동일한 확률 분포를 갖는다.

- 모델을 훈련시킨 후 테스트 데이터에 대해 오차를 계산.
- 훈련 오차가 낮지만 테스트 오차가 높다면 이는 과대적합을 의미함.

## 검증 데이터 Validation

테스트 데이터를 기준으로 모델 hyperparameter를 최적화 할 경우 모델이 테스트 데이터에 최적화되어 새로운 데이터에 대한 일반화 성능이 떨어질 수 있음.

이를 방지하기 위해 데이터를 한번 더 분할하여 총 **3개의 데이터 세트**를 운용

*다양한 hyperparameter에 대해 1) 훈련 데이터로 모델을 학습시키고,  
2) 검증 데이터로 hyperparameter를 최적화한 후, 3) 테스트 데이터로 최종 평가.*

## 교차 검증 Cross-validation

훈련 데이터를 여러 개의 서브셋으로 나누어,  
서브셋의 일부 조합으로 모델을 훈련하고  
나머지 서브셋으로 검증에 활용

Round 1:	Tr	Tr	Tr	V	V	Te
Round 2:	Tr	V	Tr	V	Tr	Te

# 차원 축소

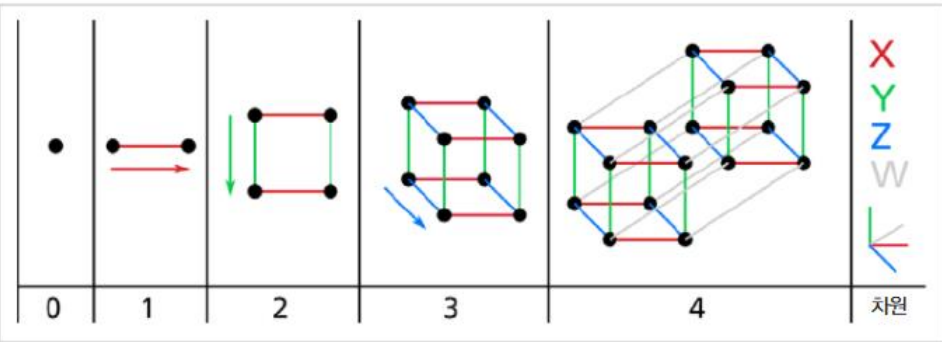
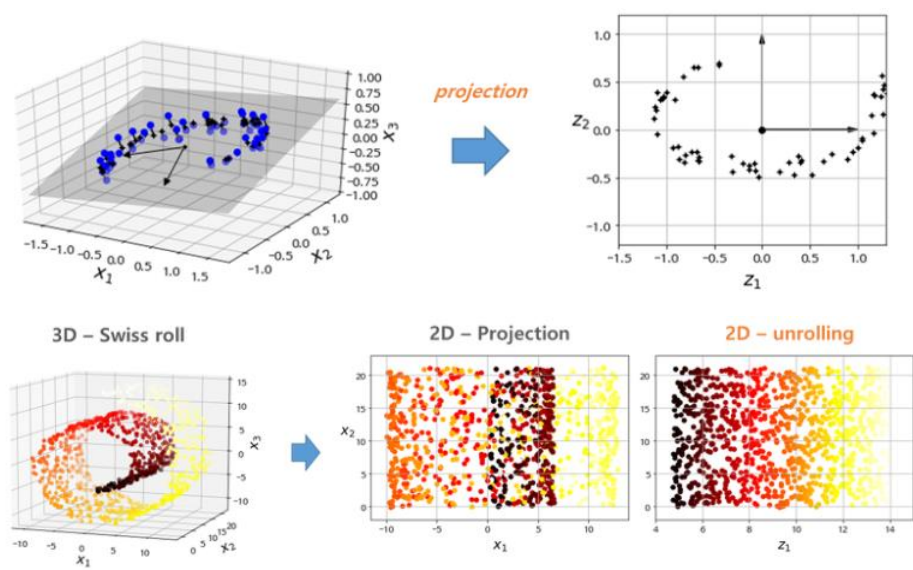


그림 8-1 점, 선, 정사각형, 정육면체, 테서렉트(0차원에서 4차원까지의 초입방체)<sup>2</sup>

## 투영

고차원 데이터를 저차원으로 투영하여 표현

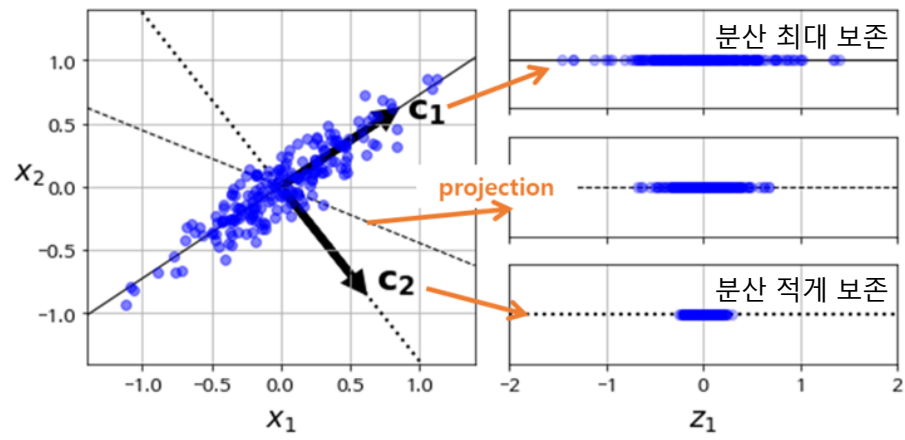


## 고차원 데이터의 문제점

- 차원 (특성의 수)이 커질수록 데이터 사이의 거리가 멀어짐 (sparse 데이터)
- 데이터의 크기 대비 담고있는 정보량이 줄어들어 연산량이 많아짐
- 특성이 데이터보다 많을 경우 제대로 학습이 안됨
- 많은 데이터를 훈련하거나 특성 차원을 축소하여 해결**

## PCA

분산이 최대가 되는 축을 탐색하여 정보 보존 (최대 분산 보존 축 == 정보 손실 최소)



# 실습 1: Machine learning

데이터 시각화 및 분석

간단한 machine learning 모델 활용

**notebooks/1\_data\_visualization.ipynb**

**notebooks/2\_machine\_learning.ipynb**

# 번외: ChatGPT

J

You

iris.csv  
Spreadsheet

이 데이터로 scikit-learn을 이용해 classification을 할 거야. Linear classification, support vector machine 등을 이용해서 classification하는 코드를 만들어줘. 가능하면 다른 classifier를 사용해도 좋아. 5종 이상의 서로 다른 모델 부탁해

DA

Data Analyst

먼저 업로드한 데이터를 확인해 보겠습니다. 데이터를 불러온 후에는 다음과 같은 5종류의 분류 모델을 사용해 분류를 진행해 볼 예정입니다:

1. **Logistic Regression (로지스틱 회귀)** - 선형 분류 모델 중 하나입니다.
2. **Support Vector Machine (SVM)** - 비선형 및 선형 분류에 널리 사용됩니다.
3. **Decision Tree (결정 트리)** - 직관적이고 이해하기 쉬운 모델로, 복잡한 데이터셋의 패턴을 학습할 수 있습니다.
4. **Random Forest (랜덤 포레스트)** - 결정 트리의 앙상블 모델로, 과적합을 줄이고 더 나은 성능을 제공합니다.
5. **K-Nearest Neighbors (KNN)** - 인스턴스 기반 학습으로, 가장 가까운 K 개의 이웃 데이터를 기반으로 분류합니다.

먼저, 업로드한 `iris.csv` 파일을 읽어서 데이터셋의 구조와 내용을 확인해 보겠습니다.

데이터셋에는 다음과 같은 열이 포함되어 있습니다: `Id`, `SepalLengthCm`, `SepalWidthCm`, `PetalLengthCm`, `PetalWidthCm`, 그리고 `Species`. `Species`는 분류 대상인 타겟 변수입니다.

이제 5가지 분류 모델을 사용하여 분류를 진행해 보겠습니다. 우선 데이터를 특성과 타겟으로 분리하고, 훈련 세트와 테스트 세트로 나누겠습니다. 그 후에 각 모델을 훈련시키고, 간단한 성능 평가를 진행하겠습니다. [↗]

Analysis

×

```
python Copy code

import pandas as pd

# 데이터 파일 경로
file_path = '/mnt/data/iris.csv'

# 데이터 불러오기
iris_data = pd.read_csv(file_path)

# 데이터셋의 처음 몇 줄 확인
iris_data.head()
```

Result

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalW.
0	1	5.1	3.5	1.4	
1	2	4.9	3.0	1.4	
2	3	4.7	3.2	1.3	
3	4	4.6	3.1	1.5	
4	5	5.0	3.6	1.4	

# 번외: ChatGPT

각 분류 모델의 테스트 세트에 대한 정확도는 다음과 같습니다:

- **Logistic Regression (로지스틱 회귀):** 100%
- **SVC (Support Vector Classifier):** 100%
- **Decision Tree (결정 트리):** 100%
- **Random Forest (랜덤 포레스트):** 100%
- **KNN (K-Nearest Neighbors):** 100%

이 결과는 Iris 데이터셋이 상대적으로 간단하며, 각 클래스가 잘 구분되어 있기 때문에 다양한 분류 모델들이 모두 높은 정확도를 달성할 수 있음을 보여줍니다. 분석 과정에서 데이터를 표준화하고, 훈련 세트와 테스트 세트로 나누어 모델을 학습시키고 평가했습니다. 모든 모델이 완벽한 성능을 보인 것은 Iris 데이터셋의 특성과, 분류하기 쉬운 패턴을 가지고 있기 때문입니다. [↩] ←

```
python Copy code

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# 필요한 모델 import
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

# 타겟과 특성 분리
X = iris_data.drop(['Id', 'Species'], axis=1)
y = iris_data['Species']
```

```
# 훈련 세트와 테스트 세트 분리
X_train, X_test, y_train, y_test = train_test_split(X, y,

# 데이터 표준화
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 모델 정의
models = {
    "Logistic Regression": LogisticRegression(),
    "SVC": SVC(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "KNN": KNeighborsClassifier()
}

# 각 모델 훈련 및 평가
results = {}
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    accuracy = accuracy_score(y_test, y_pred)
    results[name] = accuracy

results

Result
{'Logistic Regression': 1.0,
 'SVC': 1.0,
 'Decision Tree': 1.0,
 'Random Forest': 1.0,
 'KNN': 1.0}
```

# 번외: PyCaret

- 기계학습 모델 성능 비교
  - 약간의 코딩으로 많은 양의 기계학습 모델을 훈련시키고 비교할 수 있으며, hyperparameter까지 최적화 해 줌
  - 직접 모델을 개발한다면 이것보다 성능이 우수해야 함

## 표: iris dataset을 활용한 classification 결과

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9536	0.9857	0.9536	0.9634	0.9528	0.9298	0.9356	0.052
et	Extra Trees Classifier	0.9445	0.9935	0.9445	0.9586	0.9426	0.9161	0.9246	0.119
catboost	CatBoost Classifier	0.9445	0.9922	0.9445	0.9586	0.9426	0.9161	0.9246	0.039
xgboost	Extreme Gradient Boosting	0.9355	0.9868	0.9355	0.9440	0.9343	0.9023	0.9077	0.058
dt	Decision Tree Classifier	0.9264	0.9429	0.9264	0.9502	0.9201	0.8886	0.9040	0.037
rf	Random Forest Classifier	0.9264	0.9903	0.9264	0.9343	0.9232	0.8886	0.8956	0.120
gbc	Gradient Boosting Classifier	0.9264	0.9688	0.9264	0.9343	0.9232	0.8886	0.8956	0.151

## PyCaret Tutorial

<https://pycaret.gitbook.io/docs/get-started/tutorials>

Neural networks

# 인공신경망

인공신경망 기초와 훈련을 위한 기법

그래프 신경망 기초

소재 데이터베이스

재료과학 연구분야의 인공지능 적용 사례

Tinker With a **Neural Network** Right Here in Your Browser.  
Don't Worry, You Can't Break It. We Promise.

Epoch 000,922    Learning rate 0.03    Activation ReLU    Regularization L2    Regularization rate 0.001    Problem type Regression

## DATA

Which dataset do you want to use?



Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

## FEATURES

Which properties do you want to feed in?

$X_1$   
 $X_2$   
 $X_1^2$   
 $X_2^2$   
 $X_1X_2$   
 $\sin(X_1)$   
 $\sin(X_2)$

+ - 2 HIDDEN LAYERS

+ -  
5 neurons

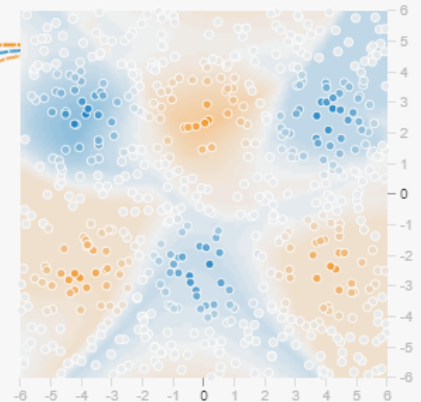
+ -  
3 neurons

This is the output from one neuron. Hover to see it larger.

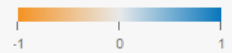
The outputs are mixed with varying weights, shown by the thickness of the lines.

## OUTPUT

Test loss 0.017  
Training loss 0.016



Colors shows data, neuron and weight values.



☐ Show test data    ☐ Discretize output



# 퍼셉트론

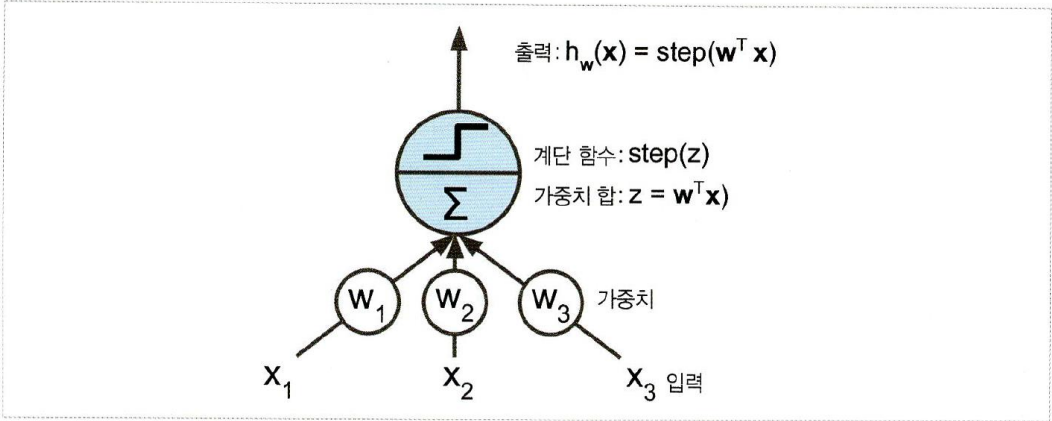
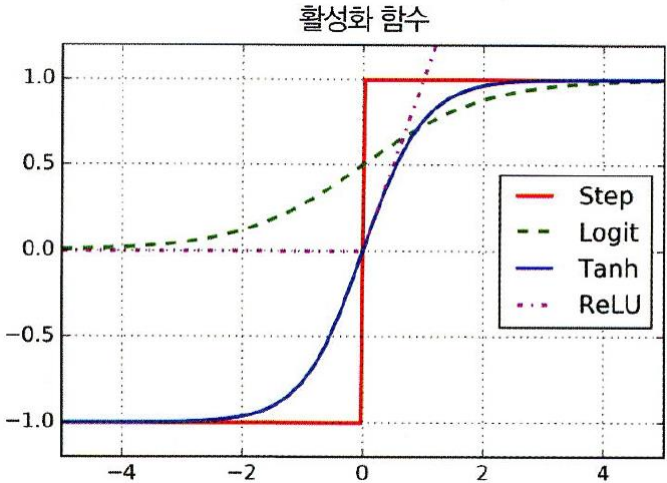


그림 10-4 TLU: 입력의 가중치 합을 계산한 다음 계단 함수를 적용하는 인공 뉴런



**퍼셉트론** Perceptron  
신경망의 기본 단위로, 다수의 입력  
신호를 연산하여 하나의 신호를 출력

**활성화 함수** Activation function  
활성화 함수가 있음으로써 비선형  
정보를 학습할 수 있음

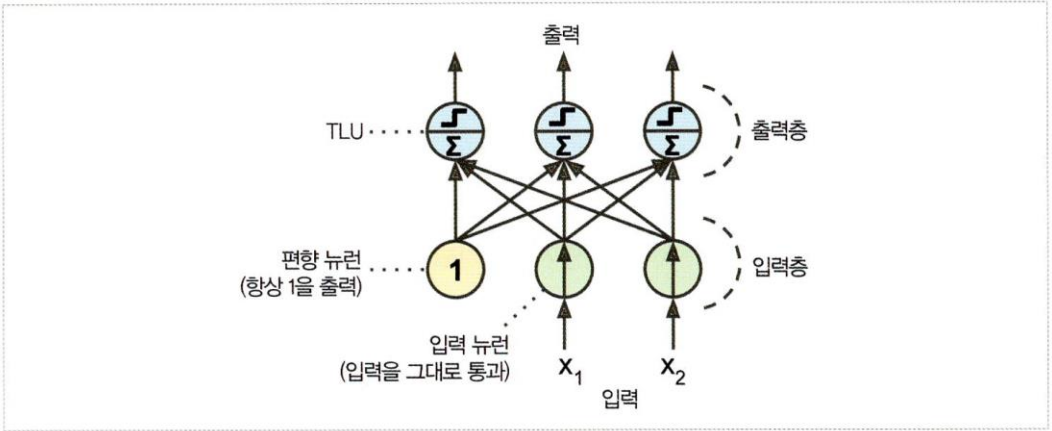


그림 10-5 입력 뉴런 두 개, 편향 뉴런 한 개, 출력 뉴런 세 개로 구성된 퍼셉트론의 구조

**다층 퍼셉트론**  
흔히 알고있는 인공신경망이 다층  
퍼셉트론에 해당함

# 신경망의 종류

## 생성형 모델 Generative models

Variational autoencoder<sup>1</sup>: 새로운 데이터를 Gaussian으로 sampling

Generative adversarial network<sup>2</sup>: 생성자와 판별자를 동시에 훈련시킴

## 거대언어모델 Large language models

Transformer<sup>3</sup>: 현재 가장 널리 활용되는 네트워크 구조

BERT<sup>4</sup>: Transformer를 적용한 언어 모델

GPT-4: ChatGPT의 기본이 되는 모델

## 순환신경망<sup>5</sup> Recurrent neural network

순서를 갖는 데이터를 처리하는데 활용 (RNN, LSTM, GRU, ...)

## 그래프 Graph neural network

그래프 형태의 데이터에 적용되는 신경망 (예: 분자구조)

1. [https://youtu.be/o\\_peo6U7IRM?si=23qzWW4WjQzHFBq](https://youtu.be/o_peo6U7IRM?si=23qzWW4WjQzHFBq)
2. <https://pseudo-lab.github.io/Tutorial-Book/chapters/GAN/Ch1-Introduction.html>
3. <https://arxiv.org/abs/1706.03762>
4. <https://arxiv.org/abs/1810.04805>
5. <https://casa-de-feel.tistory.com/39>

## ChatGPT 4 ~1.76 Trillion parameters

### 그래디언트 소실 / 폭주

심층 신경망을 진행할수록 그래디언트가 0으로 수렴하거나, 무한대로 발산하는 경우로, 하위 층 훈련이 어려워짐

Xavier initialization, dropout, batch normalization, activation function, ....

### 대규모 신경망 훈련을 위한 데이터

데이터가 충분하지 않을 경우, 새로운 데이터 작성 (데이터 수집 및 레이블 부여 등)에 많은 비용 발생.

Transfer learning, data sampling, data augmentation, self- and semi-supervised learning, ...

Overfitting과 같은 문제 발생

### 느린 훈련 속도

많은 파라미터로 인한 높은 연산능력 요구

TPU, parallelization, Cloud computing, ...

# 훈련 속도를 높이기 위한 알고리즘

## 최적화 알고리즘 Optimizer

경사하강법 (gradient descent)을 보다 발전시킨 다양한 방법론들이 존재함

훈련 과정에서 기울기 (gradient)와 학습률 (learning rate)을 조절하며 loss가 빠르게 수렴할 수 있도록 함.

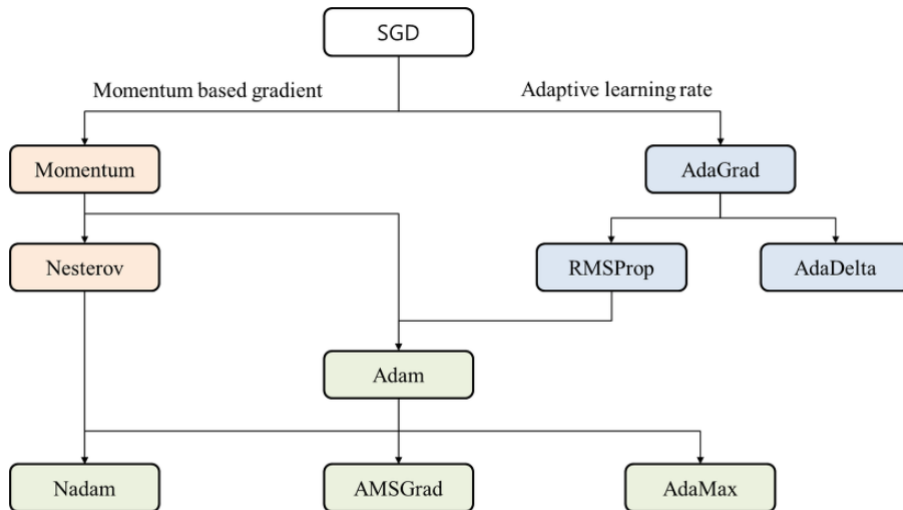


그림 1. SGD 기반의 알고리즘 분류도

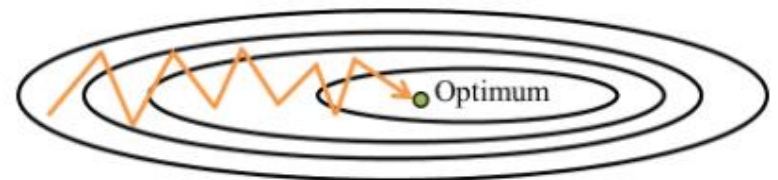


그림 2. 기본적인 SGD를 통한 최적화

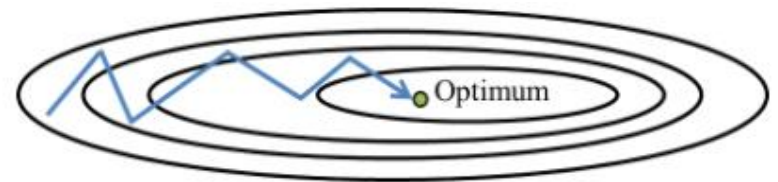


그림 3. SGD와 momentum을 결합한 최적화

# 그래프와 그래프 신경망

## 그래프

고정된 형태가 없으며, 시각화가 어려움  
관계/상호작용 등 추상적인 개념 다루기에 좋음  
COVID-19의 확산, 미디어의 영향, 소셜 네트워크 등

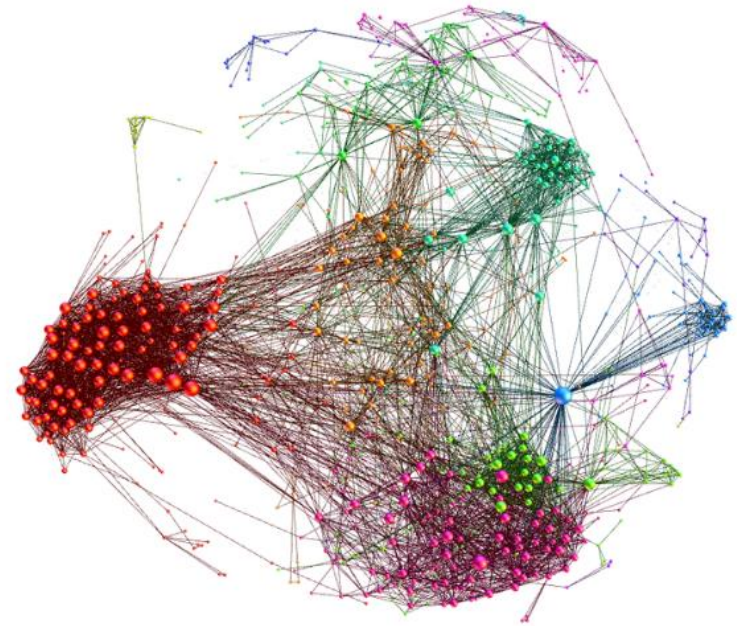
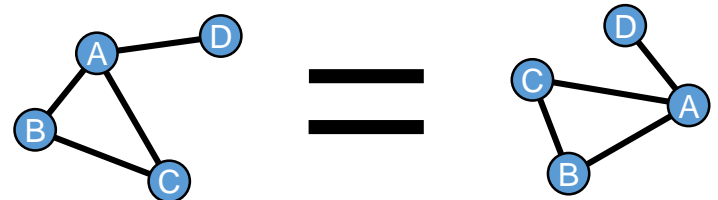
$$G = (V, E) \text{ \# vortex, edge}$$

## 그래프 신경망 Graph neural network

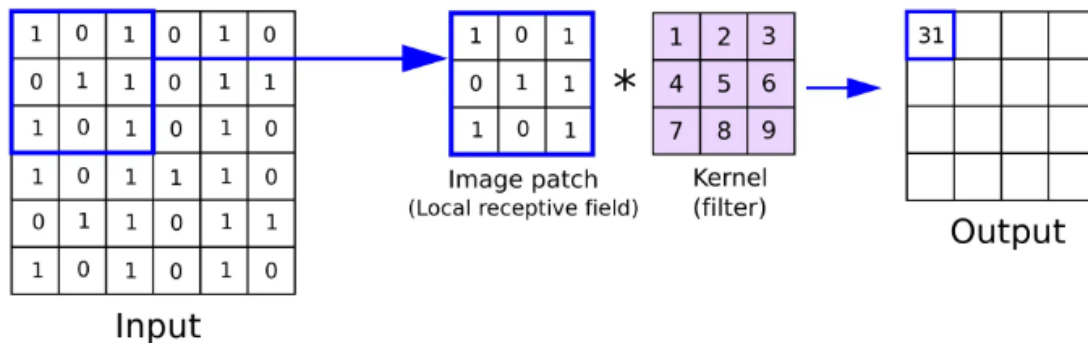
그래프로 나타낸 데이터를 인공신경망을 이용해  
분석하는 기법

노드 정보를 업데이트하는 방식에 따라 다양한  
방법이 존재함

Message passing, convolution, attention, ...

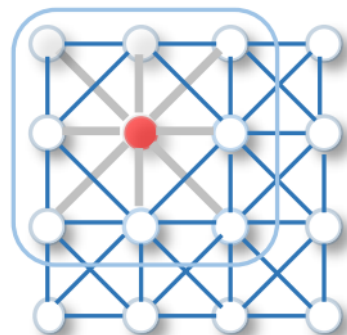


# Convolution and graph convolution



## Convolution

정해진 커널을 이용해 일정 범위  
내의 픽셀 데이터를 연산  
모델에서는 커널의 수치를 학습



(a) 2D Convolution. Analogous to a graph, each pixel in an image is taken as a node where neighbors are determined by the filter size. The 2D convolution takes the weighted average of pixel values of the red node along with its neighbors. The neighbors of a node are ordered and have a fixed size.



(b) Graph Convolution. To get a hidden representation of the red node, one simple solution of the graph convolutional operation is to take the average value of the node features of the red node along with its neighbors. Different from image data, the neighbors of a node are unordered and variable in size.

## Graph convolution

그래프에서는 edge로 연결된 노드들의 정보를 convolution함

Convolution 과정에 edge 자체에 포함된 정보들을 연산할 수 있음 (edge attributions)

Convolution된 정보들은 다음 층에서 노드 정보가 됨

Fig. 1: 2D Convolution vs. Graph Convolution.



# 소재 데이터 플랫폼 및 데이터베이스

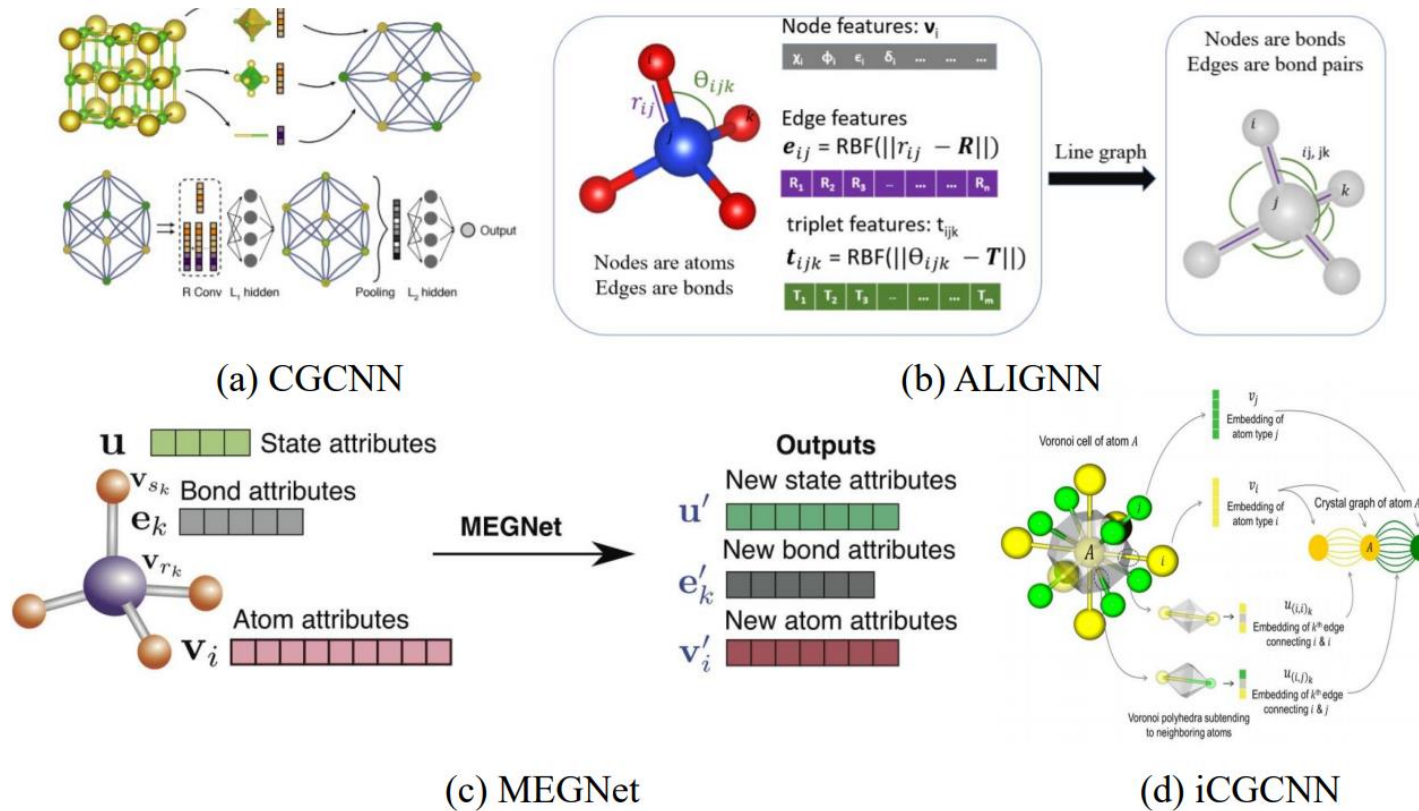


## DFT 계산 기반 DB

AFLOWLIB, JARVIS, **Materials Project**, MatHub-3d, NOMAD, OQMD, TEdesignLab, 2DmatPedia, C2DB, Materials Cloud, Open Catalyst, ...

Adv. Sci. 6 (2019) 1900808

# 그래프를 이용한 소재 표현



**Fig. 2** Schematic representations of an atomic structure as a graph. a) CGCNN model in which crystals are converted to graphs with nodes representing atoms in the unit cell and edges representing atom connections. Nodes and edges are characterized by vectors corresponding to the atoms and bonds in the crystal, respectively [Reprinted with permission from ref. [95] Copyright 2019 American Physical Society], b) ALIGNN [93] model in which the convolution layer alternates between message passing on the bond graph and its bond-angle line graph c) MEGNet in which the initial graph is represented by the set of atomic attributes, bond attributes and global state attributes [Reprinted with permission from ref. [33] Copyright 2019 American Chemical Society] model, d) iCGCNN model in which multiple edges connect a node to neighboring nodes to show the number of Voronoi neighbors [Reprinted with permission from ref. [133] Copyright 2019 American Physical Society]



## Machine learning potentials (MLP)를 사용한 large scale simulation

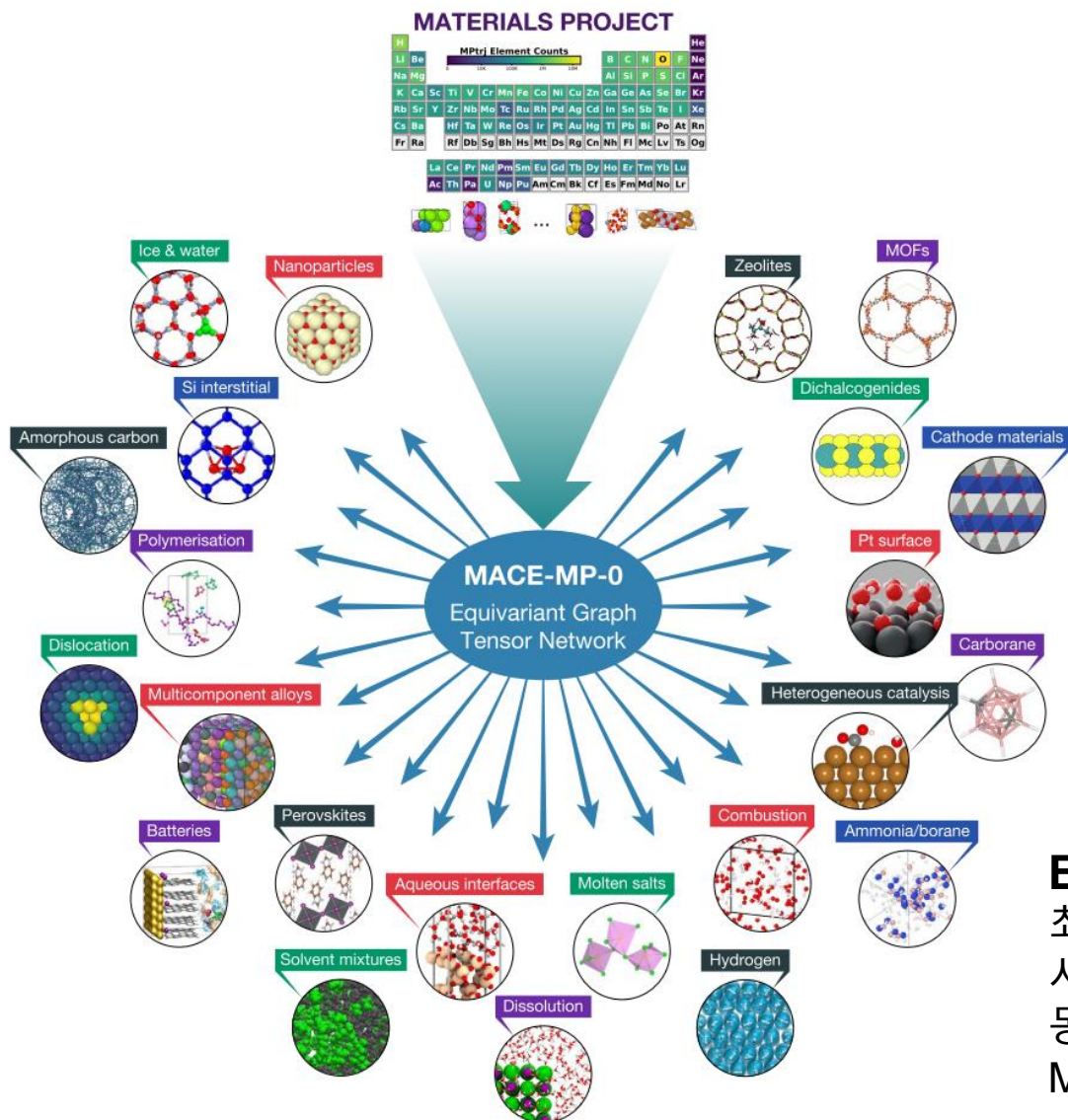


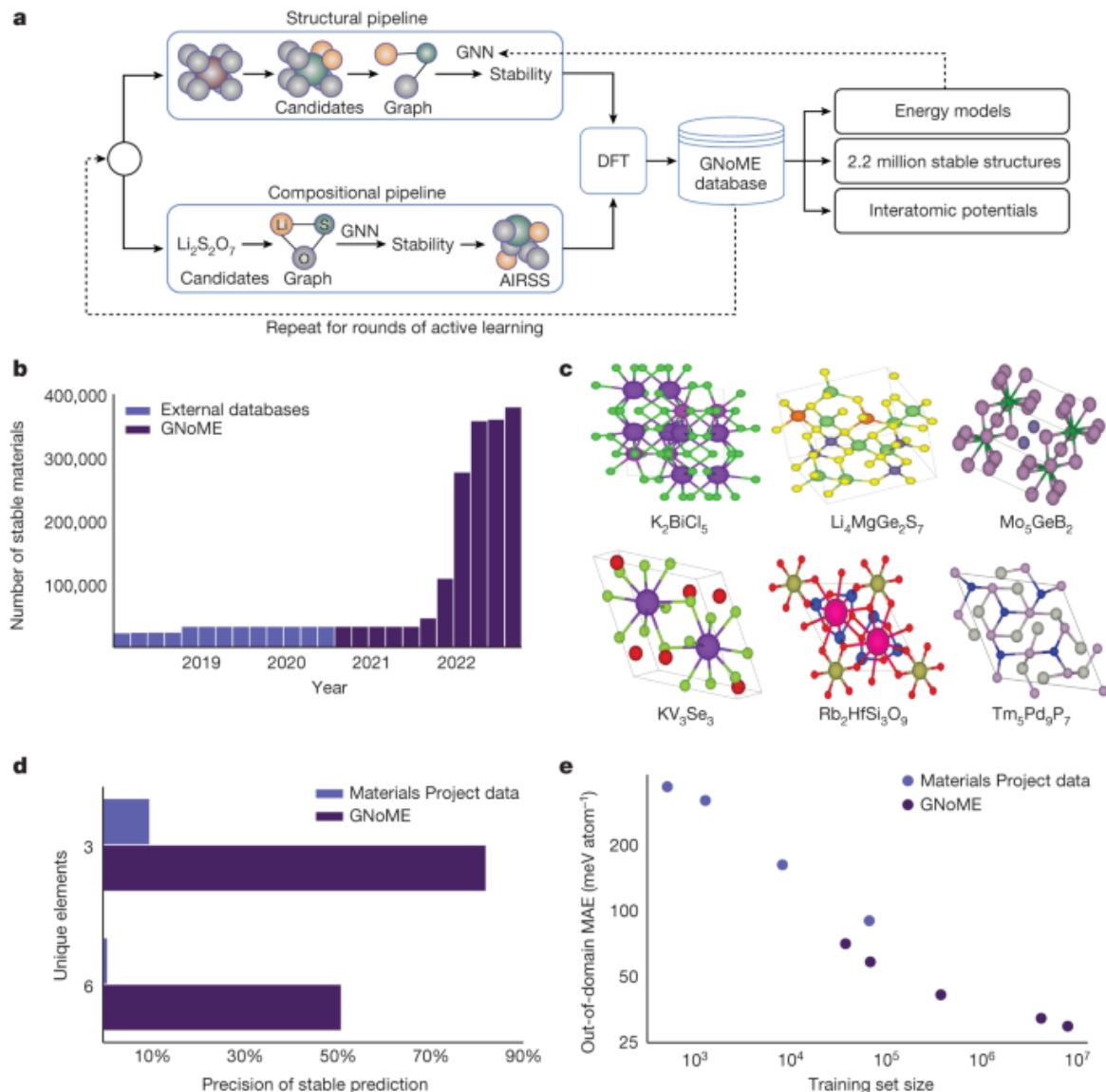
Figure 1: A foundation model for materials modelling. Trained only on Materials Project data (19) which consists primarily of inorganic crystals and is skewed heavily towards oxides, MACE-MP-0 is capable of molecular dynamics simulation across a wide variety of chemistries in the solid, liquid and gaseous phases.

**Equivariant graph neural network**  
최근 주목받는 기법으로, 결정구조 내에서의 transformation이 feature 공간에도 동일하게 영향을 주도록 설계한 신경망.  
MACE, e3nn

arXiv: 2401.00096

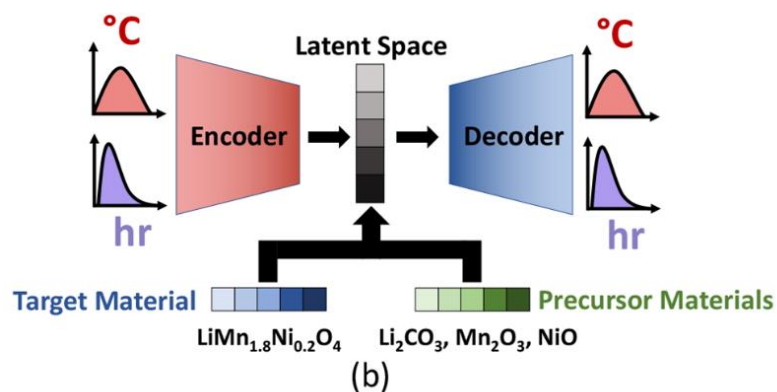
# 그래프를 활용한 소재 연구

## Google DeepMind의 GNoME – MLP 및 특성 예측

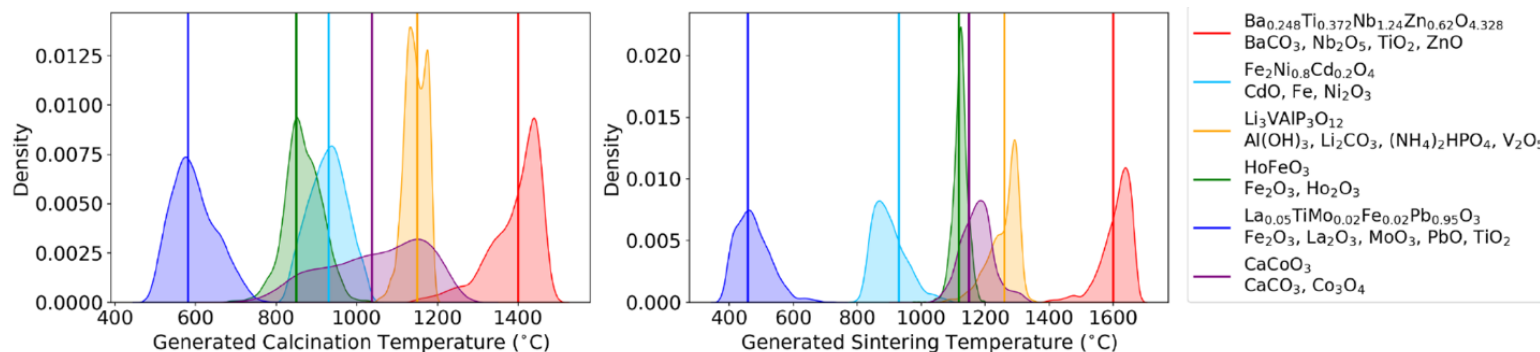


**a**, A summary of the GNoME-based discovery shows how model-based filtration and DFT serve as a data flywheel to improve predictions. **b**, Exploration enabled by GNoME has led to 381,000 new stable materials, almost an order of magnitude larger than previous work. **c**, 736 structures have been independently experimentally verified, with six examples shown. **d**, Improvements from graph network predictions enable efficient discovery in combinatorial regions of materials, for example, with six unique elements, even though the training set stopped at four unique elements. **e**, GNoME showcases emergent generalization when tested on out-of-domain inputs from random structure search, indicating progress towards a universal energy model.

## VAE를 활용한 소재 합성법 예측



**Figure 1.** ML model architectures for predicting inorganic reaction conditions. Model architecture for prediction reaction condition distributions using the conditional variational autoencoder model.



**Figure 7a.** Generated solid-state calcination (left) and sintering (right) temperature distributions for a random selection of samples in the held-out test set, with the ground truth denoted by the vertical colored line for each reaction.

# 생성형 모델을 활용한 소재 연구

## 언어모델을 활용한 분자구조 생성

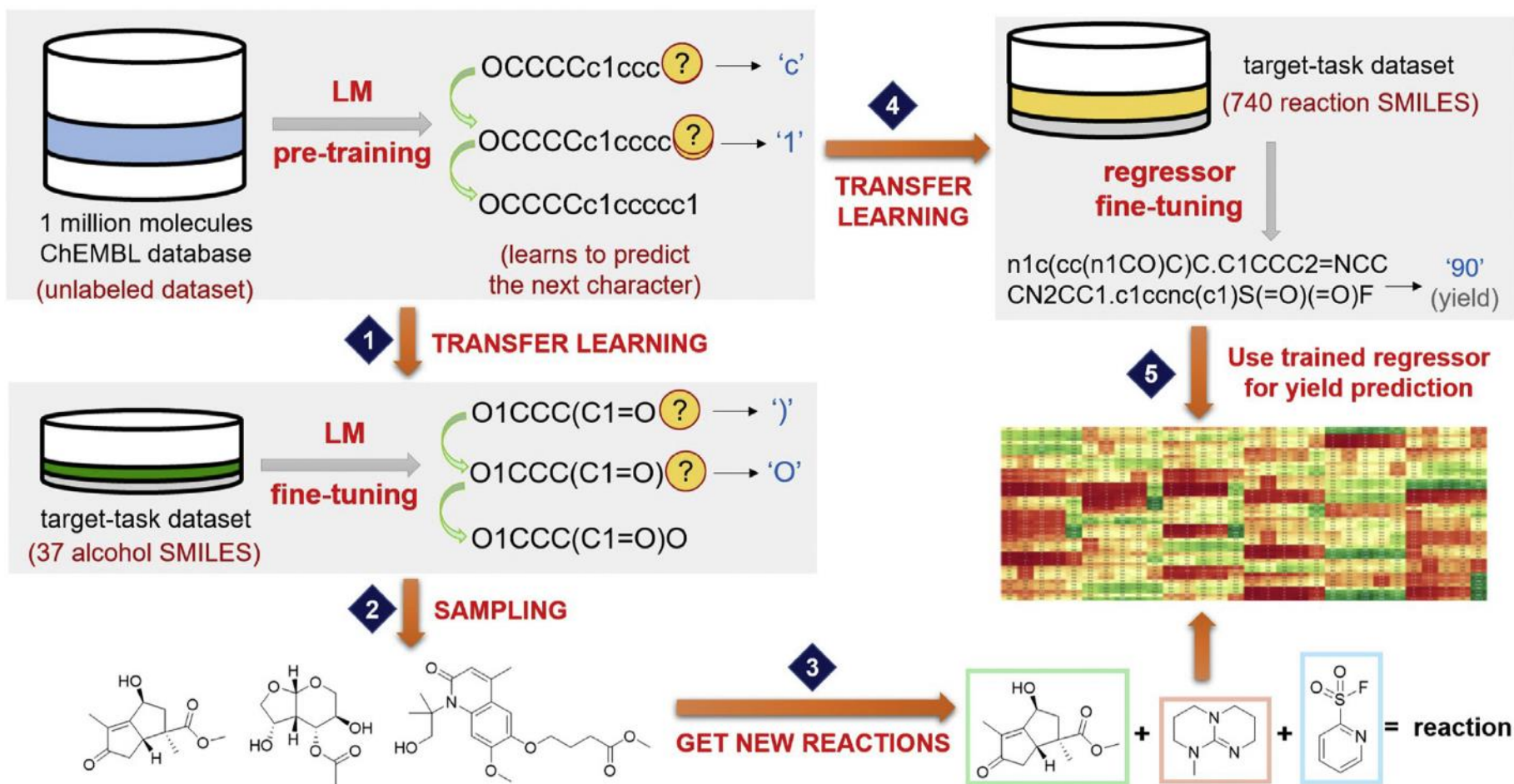


Figure 2. An overview of the transfer learning approach used for generating the focused library of novel substrates.

# 실습 2: Neural network

Pytorch를 이용한 NN 구현

Materials project API 및 Pymatgen 활용

Pytorch geometric을 이용한 GNN 구현

**notebooks/3\_neural\_networks.ipynb**

**notebooks/4\_materials\_project.ipynb**

**notebooks/5\_graph\_neural\_networks.ipynb**