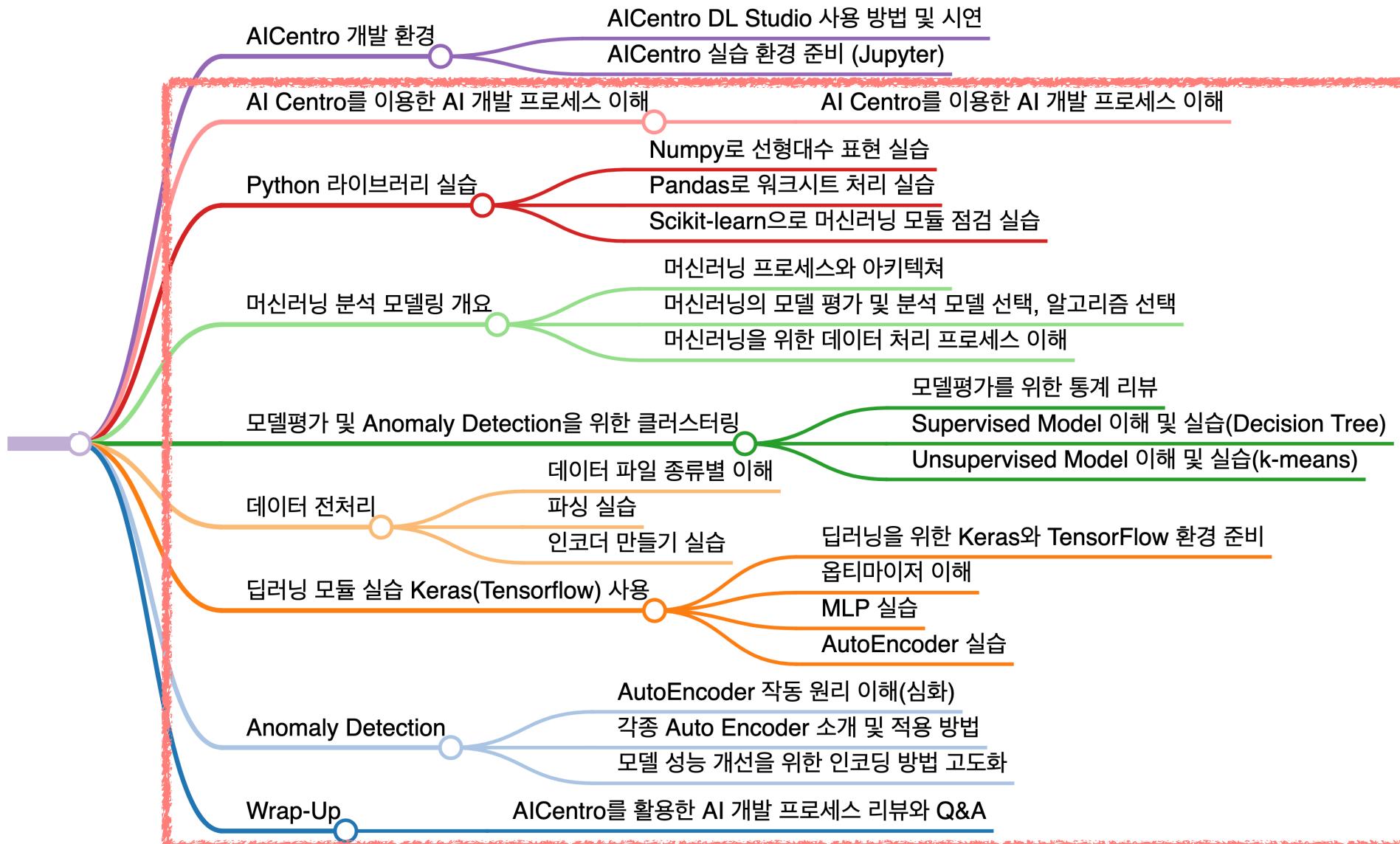




[이상 징후 탐지 Pre-Lab] AI Centro 플랫폼 기반 AI 서비스 개발

nowage@gmail.com

Agenda





AICentro를 이용한 AI 개발 프로세스 이해



AICentro



데이터 수집

-파일업로드/데이터 관리 기능



데이터 전처리

-데이터 Scaling 지원
-결측치 제거 및 보정 지원



모델 학습

-워크플로우 기반 모델 생성
-다양한 딥러닝 프레임워크 지원



모델 평가

-모델 성능 지표 관리
-학습 진행 상태 모니터링



모델 개선

-학습 히스토리 관리



모델 배포

-모델 형상 관리 지원
-배포 자동화 기능 지원



모델 사용

-실시간 Serving API 제공
-인증/인가 기능 제공



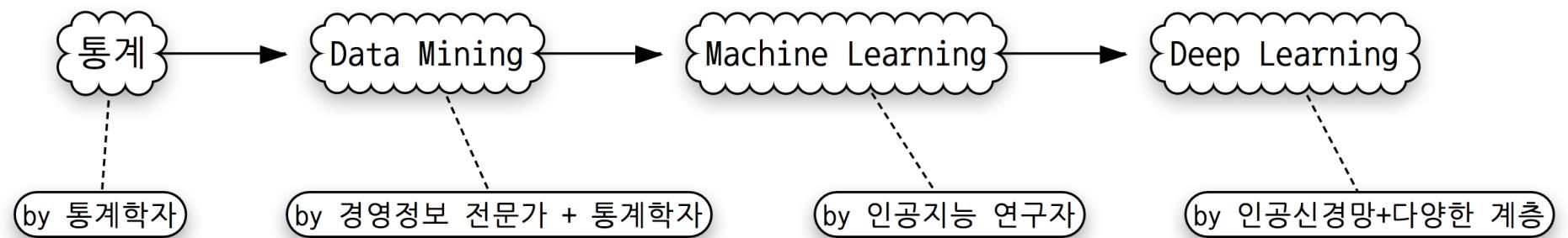
모니터링

-트래픽 모니터링 가능
-주요 성능 지표 모니터링

<https://www.aicentro.ai>

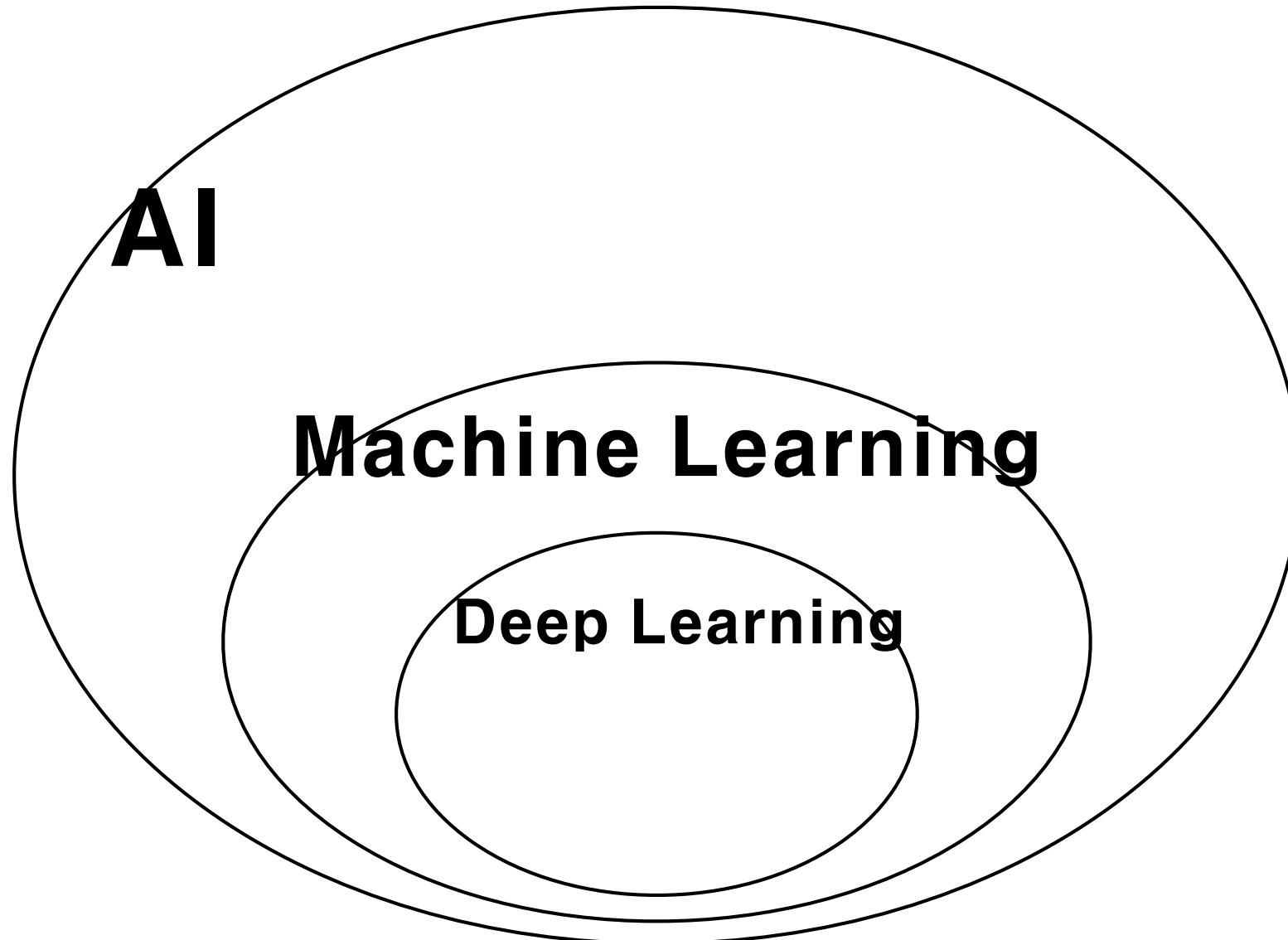


Analytic 분류





AI, Machine Learning, Deep Learning





Machine Learning Process

- 1단계 : 데이터 로딩
- 2단계 : 학습 데이터/ 평가 데이터로 분리
- 3단계 : 학습(Training)
- 4단계 : 평가(Test)
- 5단계 : 모델 저장
- 6단계 : 서비스 활용



Python 라이브러리 실습

Numpy, Pandas, Scikit-Learn 이해

Numpy로 선형대수 실습

Pandas로 워크시트 처리 실습

Scikit-learn으로 머신러닝 모듈 점검 실습



Numpy, Pandas, Scikit-Learn 올해



Python 기본 라이브러리 소개

... many many more ...

OpenCV

astropy

PySAL

BioPython

GDAL

PyTables

Numba

Sympy

NumExpr

scikit-image

statsmodels

scikit-learn

Cython

SciPy

Pandas

Matplotlib

NumPy



Python 기본 라이브러리 소개

- **Numpy**. Adds Python support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.
- **SciPy** is a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.
- **Pandas**. Software library written for data manipulation and analysis in Python. Offers data structures and operations for manipulating numerical tables and time series.
- **Scikit-learn** is a Python module for machine learning built on top of SciPy and distributed under the 3-Clause BSD license.



Python - NumPy

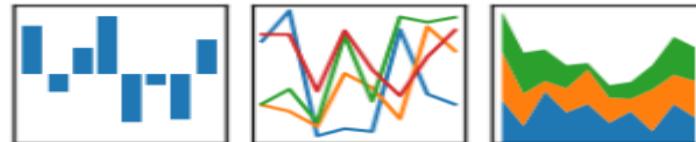
- Numpy : Adds Python support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.
- Numpy 특징
 - NumPy는 고성능의 과학계산 컴퓨팅과 데이터 분석에 필요한 기본 패키지입니다.
 - NumPy는 고성능의 과학계산 컴퓨팅과 데이터 분석에 필요한 기본 패키지입니다.
 - 메모리를 효율적으로 사용하며, 빠른 벡터 산술연산과 브로드캐스팅 기능을 제공함
 - 반복문 없이 전체 데이터에 빠른 연산을 제공함
 - 배열 데이터를 디스크에 읽고, 쓰기를 제공함
 - 선형대수, 난수발생기, 푸리에 변환 기능 제공함
 - C/C++, 포트란으로 쓰여진 코드를 통합하는 도구



Pandas

- Pandas는 높은 수준의 데이터 구조와 쉽고 빠른 데이터 분석 도구를 제공합니다.
- 데이터 정렬과 잘못 정렬된 데이터의 오류를 예방
- 통합된 시계열/비시계열 기능
- 다양한 데이터 색인(indexing) 방식을 제공

pandas
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



[overview](#) // [get pandas](#) // [documentation](#) // [community](#) // [talks](#) // [donate](#)

Python Data Analysis Library

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the [Python](#) programming language.

pandas is a [NUMFocus](#) sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project, and makes it possible to [donate](#) to the project.

A Fiscally Sponsored Project of
NUMFOCUS
OPEN CODE = BETTER SCIENCE

v0.20.2 Final (June 4, 2017)

VERSIONS

Release
0.20.2 - June 2017
[download](#) // [docs](#) // [pdf](#)

Development
0.21.0 - 2017
[github](#) // [docs](#)

Previous Releases
0.19.2 - [download](#) // [docs](#) // [pdf](#)
0.18.1 - [download](#) // [docs](#) // [pdf](#)
0.17.1 - [download](#) // [docs](#) // [pdf](#)
0.16.2 - [download](#) // [docs](#) // [pdf](#)
0.15.2 - [download](#) // [docs](#) // [pdf](#)
0.14.1 - [download](#) // [docs](#) // [pdf](#)
0.13.1 - [download](#) // [docs](#) // [pdf](#)



Data Structures

- Pandas를 쓰는 이유. Self-describing data structure를 제공하며, 이러한 data structure는 data를 이해하고, 전 처리하고, 탐색하는데 유용합니다
- A Series is a 1D data-structure.
- A DataFrame is a 2D data-structure that can be viewed as a dictionary of Series.
- A Panel is a 3D data-structure that can be viewed as a dictionary of DataFrames.

Diagram illustrating the structure of a Pandas DataFrame as a 3D cube:

- The vertical axis is labeled *index*.
- The horizontal axis is labeled *columns*.
- The depth axis is labeled *items*.

The DataFrame is labeled **TAMU** and **UT**.

	Rk	Fst N	Lst N	DoB	Gend	Grade	offset
1	4	John	Doe	1981	M	4.18	-1.18
2	2	Jill	Ford	1975	F	5.26	1.26
3	5	Chris	Jones	NaN	M	3.91	0.91
4	6	Dave	Smith	1965	M	1.23	NaN
5	1	Ellen	Frank	1973	F	5.52	0.52
6	3	Frank	Hart	1976	M	4.71	-0.71



Python-Scikit-learn

- Scikit-learn is a Python module for machine learning built on top of SciPy and distributed under the 3-Clause BSD license

<http://scikit-learn.org/stable/>

The screenshot shows the official scikit-learn website at <http://scikit-learn.org/stable/>. The header includes the scikit-learn logo, navigation links for Home, Installation, Documentation, Examples, and a search bar. Below the header is a grid of 24 small plots illustrating various machine learning algorithms like SVM, K-Means, and PCA. The main content area is titled "scikit-learn: Machine Learning in Python" and lists the following features:

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

The page is organized into several sections:

- Classification**: Identifying to which category an object belongs to. **Applications**: Spam detection, Image recognition. **Algorithms**: SVM, nearest neighbors, random forest, ...
- Regression**: Predicting a continuous-valued attribute associated with an object. **Applications**: Drug response, Stock prices. **Algorithms**: SVR, ridge regression, Lasso, ...
- Clustering**: Automatic grouping of similar objects into sets. **Applications**: Customer segmentation, Grouping experiment outcomes. **Algorithms**: k-Means, spectral clustering, mean-shift, ...
- Dimensionality reduction**: Reducing the number of random variables to consider. **Applications**: Visualization, Increased efficiency. **Algorithms**: PCA, feature selection, non-negative matrix factorization.
- Model selection**: Comparing, validating and choosing parameters and models. **Goal**: Improved accuracy via parameter tuning. **Modules**: grid search, cross validation, metrics.
- Preprocessing**: Feature extraction and normalization. **Application**: Transforming input data such as text for use with machine learning algorithms. **Modules**: preprocessing, feature extraction.



Scikit-Learn 패키지의 소개

- Scikit-Learn 패키지는 머신 러닝 교육 및 실무를 위한 파이썬 패키지로 다음과 같이 구성됨
 - 벤치마크용 샘플 데이터 세트
 - 데이터 전처리(preprocessing) 기능
 - Supervised learning
 - Unsupervised learning
 - 모형 평가 및 선택



scikit-learn 패키지에서 제공하는 머신 러닝 모형

Supervised learning : 1

- http://scikit-learn.org/stable/supervised_learning.html
- Generalized Linear Models
 - Ordinary Least Squares
 - Ridge/Lasso/Elastic Net Regression
 - Logistic regression
 - Polynomial regression
 - Perceptron



scikit-learn 패키지에서 제공하는 머신 러닝 모형

Supervised learning : 2

- Linear and Quadratic Discriminant Analysis
- Support Vector Machines
- Stochastic Gradient Descent
- Nearest Neighbor Algorithms
- Gaussian Processes
- Naive Bayes
- Decision Trees
- Ensemble methods
 - Random Forests
 - AdaBoost



Numpy로 선형대수 실습



Numpy 배열

1. ndarray : 다차원 배열 객체

```
import numpy as np  
data = np.random.randn(2, 3)  
>>> data  
array([[-0.06742085, 1.76854625, -0.1363485 ],  
[-0.19151926, 2.15434917, 0.8728469 ]])
```



ndarray의 생성

- array 함수를 사용한 배열 생성 배열을 가장 쉽게 생성하는 방법은 array함수를 사용하여 리스트를 배열로 바꾸는 것입니다.

```
list = [ 5, 2.2, 6, 8, 9, 10 ]
arr = np.array(list)
>>> print arr
[ 5. 2.2 6. 8. 9. 10. ]
>>> print arr.dtype
float64
```

- 다차원 리스트도 array 함수를 통해 배열로 변경할 수 있습니다.

```
list2 = [[1, 2, 3, 4], [5, 6, 7, 8]]
arr = np.array(list2)
>>> arr
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```



ndarray의 생성

- zeros나 ones를 사용한 배열 생성 다음은 zeros 함수와 ones 함수 등을 사용해서 initializing 할 수 있습니다.

```
np.zeros(10)
np.zeros((2, 5))
np.ones(10)
np.ones((2, 5))
np.arange(15)
np.empty((2, 3, 5))
```

Array에 정의된 operators에 대해서

```
list = [[1, 2, 3], [4, 5, 6]]
arr = np.array(list)
>>> arr * 10
array([[10, 20, 30],
       [40, 50, 60]])
>>> arr - arr
array([[0, 0, 0],
       [0, 0, 0]])
>>> 1.0 / arr
array([[ 1., 0.5 , 0.33333333],
       [ 0.25 , 0.2 , 0.16666667]])
>>> arr ** 0.5
array([[ 1. , 1.41421356, 1.73205081],
       [ 2. , 2.23606798, 2.44948974]])
# multiplication(*), addition(+) 에 대한 정의가 python List와 다르다는 것에 주의!
# -, /, ** => list에는 정의되지 않음.
>>> data.reshape(1, 6)
array([[1, 2, 3, 4, 5, 6]])
```

Array가 가진 유용한 attribute들

- `ndarray.ndim`

the number of axes (dimensions) of the array. In the Python world, the number of dimensions is referred to as rank.

- `ndarray.shape`

the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with n rows and m columns, shape will be (n,m) . The length of the shape tuple is therefore the rank, or number of dimensions, ndim .

- `ndarray.size`

the total number of elements of the array. This is equal to the product of the elements of shape .

- `ndarray.dtype`

an object describing the type of the elements in the array. One can create or specify dtype's using standard Python types. Additionally NumPy provides types of its own. `numpy.int32`, `numpy.int16`, and `numpy.float64` are some examples.

- `ndarray.itemsize`

the size in bytes of each element of the array. For example, an array of elements of type `float64` has itemsize 8 (=64/8), while one of type `complex32` has itemsize 4 (=32/8). It is equivalent to

- `ndarray.dtype.itemsize` .



slice

- Python 리스트와의 차이는 array slice는 원본 array의 view라는 점입니다. slice 자체에 대한 직접적인 값의 입력은 원본 array의 값 역시 변화시킵니다.

```
>>> arr = np.arange(10)
>>> arr
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> new = arr[5:8]
array([5, 6, 7])
>>> new[0] = 100
>>> arr
array([0, 1, 2, 3, 4, 100, 6, 7, 8, 9])
```

실습 : Numpy로 배열 곱하기



1. $[[1, 2], [3, 4]]$ 와 $[[2, 0], [0, 2]]$ 배열을 만들기
2. 두개의 배열 곱하기
3. 답이 맞나요? $[[2, 4], [6, 8]]$
4. Hint! $\cdot == \text{dot product}$





Pandas로 워크시트 처리 실습



SERIES

```
import pandas as pd

# Get Series Values and index
series1.values
series1.index

# Get Values by index
series1 = pd.Series ([1,2], index = ['a', 'b'])
series1[1] # access elementes based on position
series1['a'] # access elements based on label
series1[['b','a']] # access elements based on labels

# missing inputs check!
series1.isnull()
series1.notnull()
```



DATA FRAME

```
# Create Data Frame
```

```
>>> data = { 'state':['ohio', 'ohio', 'ohio', 'nevada', 'nevada'], 'year':  
[2000,2001, 2002, 2001, 2002], 'pop' : [1.5, 1.7, 3.6, 2.4, 2.9] }
```

```
>>> frame = pd.DataFrame(data = data, columns = ['state', 'year'], index =  
['a', 'b' , 'c', 'd', 'd'])
```

```
# 접근은 사전(dict) 형태의 표기법이나 속성(attribute) 형태로 할 수 있다.
```

```
>>> frame['state']
```

or

```
>>> frame.state
```



DATA FRAME

새로운 column의 추가는 Series로, 그리고 index를 사용해서 입력해줄 수 있다.

```
>>> val = pd.Series([3.2, -1.5, -1.7], index = ['a', 'b', 'c'])
```

```
>>> frame['debt'] = val
```

```
>>> frame
```

	state	year	debt
a	ohio	2000	-1.2
b	ohio	2001	-1.5
c	ohio	2002	-1.7
d	nevada	2001	NaN
d	nevada	2002	NaN

index를 가진 상태에서 boolean 형을 가진 series는 Series나 DataFrame의 인덱스 그자체로 사용될 수 있다.

```
frame['eastern'] = frame['state'] == 'ohio'
```



Sorting

```
# Series Sorting  
  
>>> series.sort()  
  
# DataFrame sorting  
>>> frame.sort_index(by = 'year')  
    #하나의 column에 대해서  
>>> frame.sort_index(by = ['year', 'eastern'])  
    #두 개 이상의 columns에 대해서
```



Handling Missing inputs

(1) dropna()

```
>>> data = pd.Series([1, np.NaN, 3.5, np.NaN, 7])
>>> data
0    1.0
1    NaN
2    3.5
3    NaN
4    7.0
dtype: float64
```

(2) fillna()

```
>>> frame.fillna(1)
#모든 nan은 1로
>>> frame.fillna({'debt' : 1, 'year' : 2001})
#각 column 별로 따라 값을 줄 수도 있다.
```



Merging

- Key 하나 이상을 사용해 데이터집합의 로우를 합친다.

```
import pandas as pd
import numpy as np
data1 = {'key' : ['b', 'b', 'a', 'c', 'a', 'a', 'b'], 'data1' : range(7)}
data2 = {'key' : ['a', 'b', 'd'], 'data2' : range(3)}
```

#각 data를 DataFrame으로 만들면

```
>>> df1 = pd.DataFrame(data1)
```

```
>>> df1
```

```
   data1 key
0      0   b
1      1   b
2      2   a
3      3   c
4      4   a
5      5   a
6      6   b
```



Merging

```
>>> df2 = pd.DataFrame(data2)
>>> df2
   data2 key
0      0   a
1      1   b
2      2   d
```

#병합할 칼럼을 명시적으로 표현해주는 것이 좋다.

#how에는 inner join(교집합), outer join(합집합), left(왼쪽 데이터 기준), right(오른쪽 데이터기준)

```
>>> pd.merge(df1, df2, on = 'key', how = 'inner')
```

```
   data1 key  data2
0      0   b      1
1      1   b      1
2      6   b      1
3      2   a      0
4      4   a      0
5      5   a      0
```

Getting data, CSV & JSON

TEXT FORMAT (CSV)

```
df1 = pd.read_csv(file/URL/file-like-object,
sep = ',', header = None)

# Type-Inference : do NOT have to specify which columns are
numeric, integer, boolean or string.
```

In Pandas, missing data in the source data is usually empty string, NA, -1, #IND or NULL. You can specify missing values via option i.e.: na_values = ['NULL'].

Default delimiter is comma.

Default is first row is the column header.

```
df1 = pd.read_csv(..., names = [...])
```

Explicitly specify column header, also imply first row is data

```
df1 = pd.read_csv(..., names = [...,
'date'], index_col = 'date')
```

Want 'date' column to be row index of the returned DF

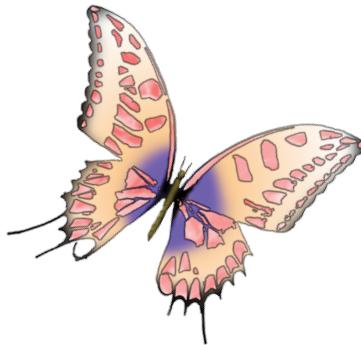
```
df1.to_csv(filepath/sys.stdout, sep = ',')
```

Missing values appear as empty strings in the output. Thus,
It is better to add option i.e.: na_rep = 'NULL'

Default is row and column labels are written. Disabled by
options: index = False, header = False

```
>>> df = pd.read_csv('tmp/xxx.csv', sep = ',', header =
None)
>>> frame.to_csv('xx.csv', na_rep = "NULL", header =
False, index = False)
```

실습 : Pandas 기초



1. Csv파일 읽어 봅시다.
2. 여섯번째 열만 뽑아 봅시다.
3. for문으로 출력해 봅시다.





Scikit-learn으로 머신러닝 모듈 점검 실습



scikit-learn 패키지에서 제공하는 머신 러닝 모형 Unsupervised learning

- http://scikit-learn.org/stable/unsupervised_learning.html
- Gaussian mixture models
- Manifold learning
- Clustering
 - K-means
 - DBSCAN
- Biclustering
- Decomposing
 - Principal component analysis (PCA)
 - Factor Analysis
 - Independent component analysis (ICA)
 - Latent Dirichlet Allocation (LDA)
- Covariance estimation
- Novelty and Outlier Detection
- Density Estimation



scikit-learn의 서브 패키지

- 모형:

- sklearn.base: Base classes and utility functions
- sklearn.pipeline: Pipeline
- sklearn.linear_model: Generalized Linear Models
- sklearn.naive_bayes: Naive Bayes
- sklearn.discriminant_analysis: Discriminant Analysis
- sklearn.neighbors: Nearest Neighbors
- sklearn.mixture: Gaussian Mixture Models
- sklearn.svm: Support Vector Machines
- sklearn.tree: Decision Trees
- sklearn.ensemble: Ensemble Methods
- sklearn.cluster: Clustering

- 자료 제공:

- sklearn.datasets: 샘플 데이터 세트 제공

- 자료 전처리:

- sklearn.preprocessing: imputation, encoding 등 단순 전처리
- sklearn.feature_extraction: Feature Extraction

- 모형 평가:

- sklearn.metrics: Metrics
- sklearn.cross_validation: Cross Validation
- sklearn.grid_search: Grid Search



scikit-learn의 Class

- 전처리용 클래스

- Transformer 클래스

- ▶ 자료 변환

- 공통 메서드

- fit(): 모형 계수 추정, 트레이닝(training)

- transform(): 자료 처리

- fit_transform(): 모형 계수 추정 및 자료 처리 동시 수행

- 머신러닝 모형 클래스 그룹

- Regressor 클래스

- Classifier 클래스

- Cluster 클래스

- 공통 메서드

- ▶ fit(): 모형 계수 추정, 트레이닝
(training)

- ▶ predict(): 주어진 x값에 대해 y 예측

- ▶ score(): 성과 분석

- Pipeline 클래스

- 복수의 Preprocessor와 Model을 연결하여 하나의 Model처럼 행동

- Model 클래스가 제공하는 공통 메서드를 모두 제공

- pipeline 내부에서 Preprocessor에서 자료를 계속 변형한 후 마지막으로 Model에 입력



Scikit-Learn 패키지의 샘플 데이터 사용

- `sklearn.datasets` 서브패키지는 모형 실습을 위한 예제 데이터 세트를 제공한다.
- 예제 데이터 세트를 불러오는 명령들은 크게 다음과 같은 세가지 계열의 명령으로 나눌 수 있다.
 - `load` 계열 명령: 저장된 dataset `import`
 - `fetch` 계열 명령: 인터넷에서 캐쉬로 `download` 후 `import`
 - `make` 계열 명령: 가상 dataset을 생성



Load 계열 Sample data

- `load_boston()`: 회귀 분석용 보스턴 집값
- `load_diabetes()`: 회귀 분석용 당뇨병 자료
- `load_linnerud()`: 회귀 분석용 linnerud 자료
- `load_iris()`: classification용 iris 자료
- `load_digits()`: classification용 숫자 필기 이미지 자료
- `load_sample_image()`: 압축용 이미지

Scikit-Learn 패키지의 샘플 : 보스턴 집값

- Feature

- CRIM: 범죄율
- INDUS: 비소매상업지역 면적 비율
- NOX: 일산화질소 농도
- RM: 주택당 방 수
- LSTAT: 인구 중 하위 계층 비율
- B: 인구 중 흑인 비율
- PTRATIO: 학생/교사 비율
- ZN: 25,000 평방피트를 초과 거주지역 비율
- CHAS: 찰스강의 경계에 위치한 경우는 1, 아니면 0
- AGE: 1940년 이전에 건축된 소유주택의 비율
- RAD: 방사형 고속도로까지의 거리
- DIS: 직업센터의 거리
- TAX: 재산세율

- Target

- Boston, Owner's House Price, 1978
- 506 Towns' Median Prices (unit: USD 1,000)

```
1. Default (Python)
>>> from sklearn.datasets import load_boston
>>> boston = load_boston()
>>> print(boston.DESCR)
Boston House Prices dataset
=====
Notes
-----
Data Set Characteristics:

:Number of Instances: 506

:Number of Attributes: 13 numeric/categorical predictive

:Median Value (attribute 14) is usually the target

:Attribute Information (in order):
- CRIM      per capita crime rate by town
- ZN       proportion of residential land zoned for lots over
           25,000 sq.ft.
```

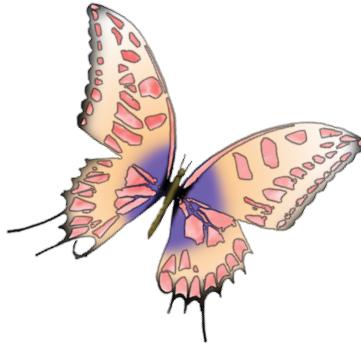
Scikit-Learn 패키지의 샘플 : 보스턴 집값

```
1. Default (Python)

>>>
>>> import pandas as pd
>>> dfX = pd.DataFrame(boston.data, columns=boston.feature_names)
>>> dfy = pd.DataFrame(boston.target, columns=["MEDV"])
>>> df = pd.concat([dfX, dfy], axis=1)
>>> df.tail()
      CRIM    ZN  INDUS  CHAS    NOX     RM    AGE     DIS     RAD    TAX \
501  0.06263  0.0   11.93  0.0   0.573  6.593  69.1  2.4786  1.0  273.0
502  0.04527  0.0   11.93  0.0   0.573  6.120  76.7  2.2875  1.0  273.0
503  0.06076  0.0   11.93  0.0   0.573  6.976  91.0  2.1675  1.0  273.0
504  0.10959  0.0   11.93  0.0   0.573  6.794  89.3  2.3889  1.0  273.0
505  0.04741  0.0   11.93  0.0   0.573  6.030  80.8  2.5050  1.0  273.0

      PTRATIO      B    LSTAT   MEDV
501       21.0  391.99   9.67  22.4
502       21.0  396.90   9.08  20.6
503       21.0  396.90   5.64  23.9
504       21.0  393.45   6.48  22.0
505       21.0  396.90   7.88  11.9
>>> █
```

실습 : Scikit-Learn 데이터 로딩



1. Scikit-Learn 셈플 데이터 중 iris를 로딩하시오.
2. Data의 마지막 몇개만 출력하시오.





머신러닝 분석 모델링 개요

머신러닝 프로세스와 아키텍쳐

머신러닝을 위한 데이터 처리 프로세스 이해

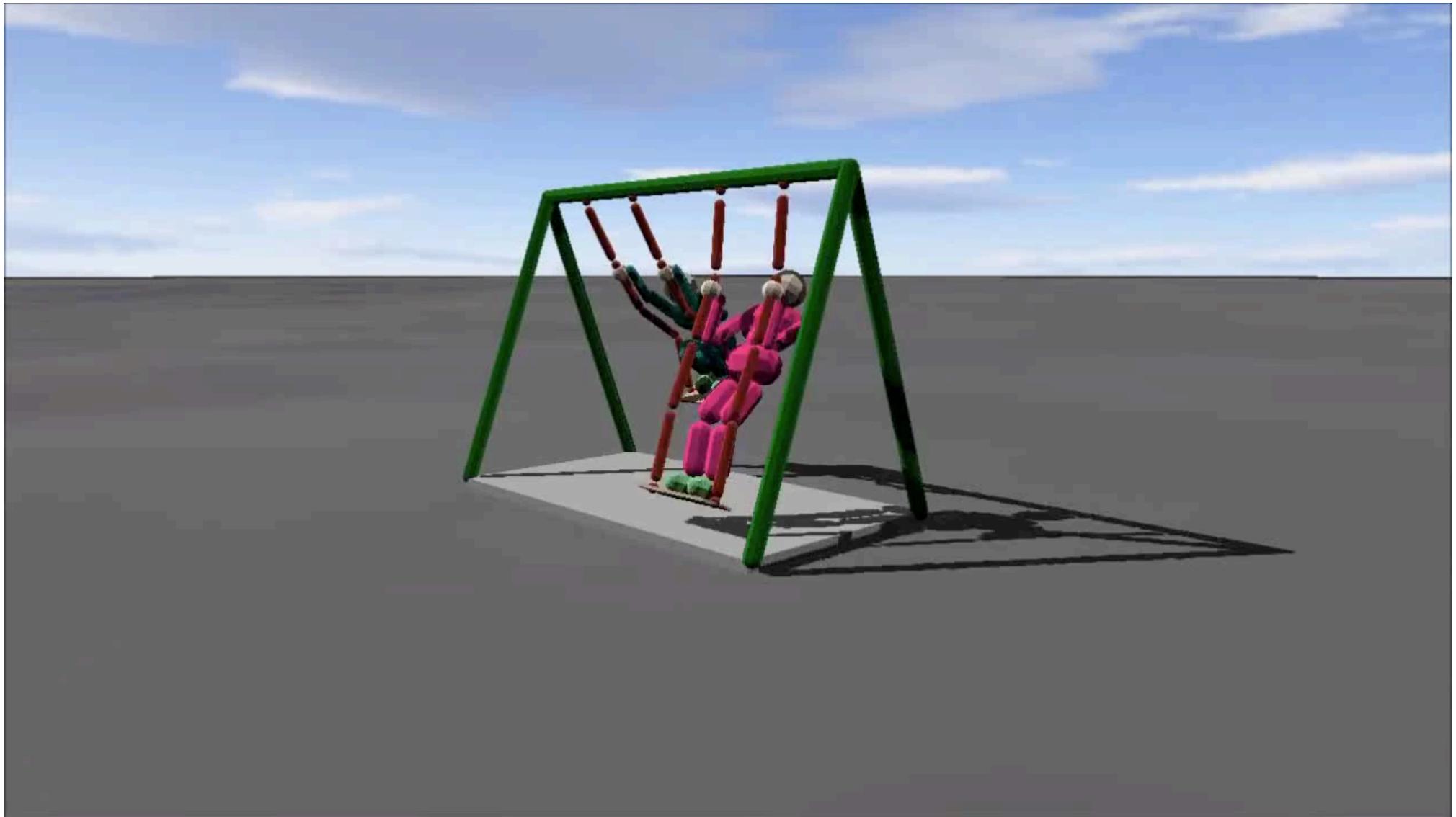
머신러닝의 모델 평가 및 분석 모델 선택, 알고리즘 선택



머신러닝 프로세스와 아키텍쳐



최적화? 인공지능?



AI



- Computer and Its Evolution
- Data/Big Data and Data Mining
- Artificial Intelligence
 - Traditional AI, Expert Systems, Neural Networks
 - Bayesian Networks
- Network, Internet and Web Evolution
 - Internet of Things, Linked Open Data
- Open Source Software Movement, Computer Security
 - Hacking, Cracking, Phishing, Smishing
- e-Commerce/Social Commerce
- Mobile Evolution and Smart Devices



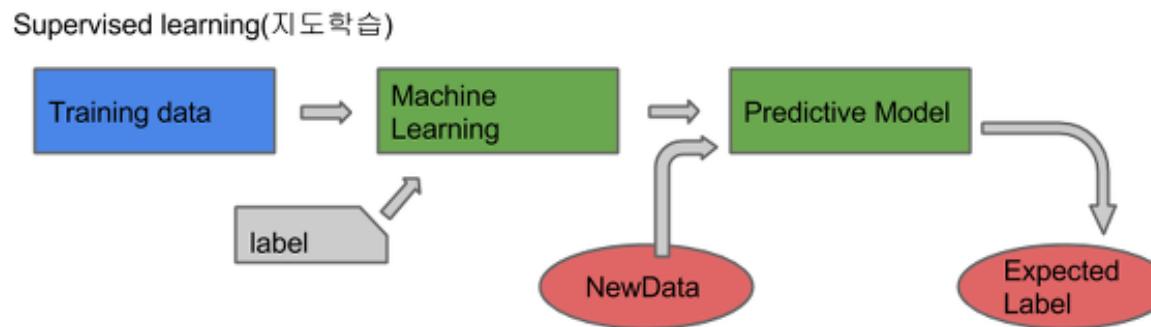
Challenges of Artificial Intelligence

- Can machines think?
 - Solve math problems
 - Play games
 - ▶ Play chess, play go, play quiz games
 - Understand human language
 - Sense things
 - Learn from experience
 - Write a plan to achieve a goal

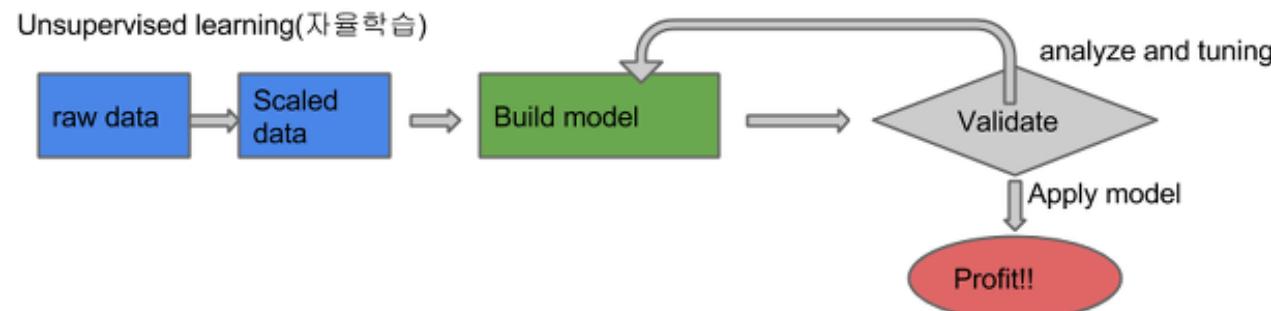


Supervised Learning vs Unsupervised Learning

- Supervised learning(지도학습) : 훈련데이터가 존재하고 그 훈련데이터로 machine learning을 해 predictive model을 만든다. 그리고 그 모델을 이용해 새로 도입되는 데이터가 어떤 class에 속하는지 파악



- Unsupervised Learning (비지도학습) : supervised learning과는 다르게 사전정보가 없는것이 특징. 데이터가 어떻게 구성되어 있는지 알아내는데 사용



Machine Learning 이론

Machine Learning



what society thinks I do



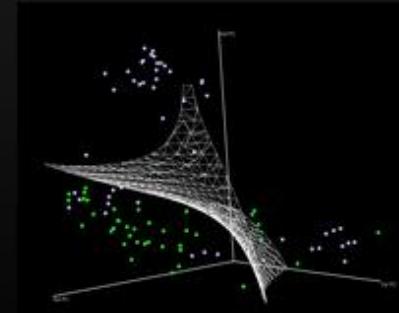
what my friends think I do



what my parents think I do

$$\begin{aligned} L_{\ell} &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^n \alpha_i \\ \alpha_i &\geq 0, \forall i \\ \mathbf{w} &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i, \sum_{i=1}^n \alpha_i = 0 \\ \nabla g(\theta_t) &= \frac{1}{n} \sum_{i=1}^n \nabla \ell(x_i, y_i; \theta_t) + \nabla r(\theta_t). \\ \theta_{t+1} &= \theta_t - \eta_t \nabla \ell(x_{i(t)}, y_{i(t)}; \theta_t) - \eta_t \cdot \nabla r(\theta_t). \\ \mathbb{E}_{i(t)} [\ell(x_{i(t)}, y_{i(t)}; \theta_t)] &= \frac{1}{n} \sum_i \ell(x_i, y_i; \theta_t), \end{aligned}$$

what other programmers think I do



what I think I do

```
>>> from scipy import SVM
```

what I really do



기계학습

- 머신 러닝(영어: machine learning) 또는 기계 학습(機械 學習)은 인공 지능의 한 분야로, 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야를 말한다. 가령, 기계 학습을 통해서 수신한 이메일이 스팸인지 아닌지를 구분할 수 있도록 훈련할 수 있다.
- 기계 학습의 핵심은 표현(representation)과 일반화(generalization)에 있다. 표현이란 데이터의 평가이며, 일반화란 아직 알 수 없는 데이터에 대한 처리이다. 이는 전산 학습 이론 분야이기도 하다. 다양한 기계 학습의 응용이 존재한다. 문자 인식은 이를 이용한 가장 잘 알려진 사례이다.



머신러닝을 위한 데이터 처리 프로세스 이해

Machine Learning 실습



1단계 : 데이터 로딩

2단계 : 학습 데이터/ 평가 데이터로 분리

3단계 : 학습(Training)

4단계 : 평가

5단계 : 모델 저장

6단계 : 서비스 활용

K-Nearest Neighbors
 Decision Tree Classifier
 Bayesian Classifier
 Logistic regression
 Support Vector Machines

K-Means Clustering
 Hierarchical Clustering
 Association Rule
 Collaborative Filtering



머신러닝의 모델 평가 및 분 석 모델 선택, 알고리즘 선택



kNN 분류란? =지도학습

- kNN방식은 분류되어있지 않은 레코드들을 분류된 레코드들 중 가장 비슷한 속성을 가진 레코드로 할당하는 것.
- Spark ML link :

<http://spark.apache.org/docs/latest/mllib-clustering.html>

최근접이웃 분류방식은 kNN 알고리즘으로 활용되며, 다음과 같은 강점과 약점을 가진다.

강점	약점
<ul style="list-style-type: none">•간단하고 효과적이다.•분석하는 데이터에 대한 기본적인 분포가정이 없다•학습과정이 빠르다.	<ul style="list-style-type: none">•모델구축을 하지 않아서 속성들 간의 관계에서 새로운 인사이트를 얻는데 제한이 있다.•분류시간이 오래 걸린다.•각각의 개체에 대해 거리를 비교하므로 메모리소모량이 많다.•명목변수와 결측값은 따로 처리를 해줘야 한다.

Decision Tree Classifier

의사 결정 나무 개요

=지도학습

1. 2개(오른쪽 가지, 왼쪽 가지)의 유형으로 분류하는데

가장 적합한 속성과 그 분리값을 찾는다.

2. 분기된 가지(오른쪽, 왼쪽)로 간 다음,

아이템들의 레이블이 모두 같으면 그 값을 반환한다.

3. 각 가지에 속한 아이템의 레이블이 동일하지 않으면 1번(분류)을 실행한다.

-> 재귀 순환

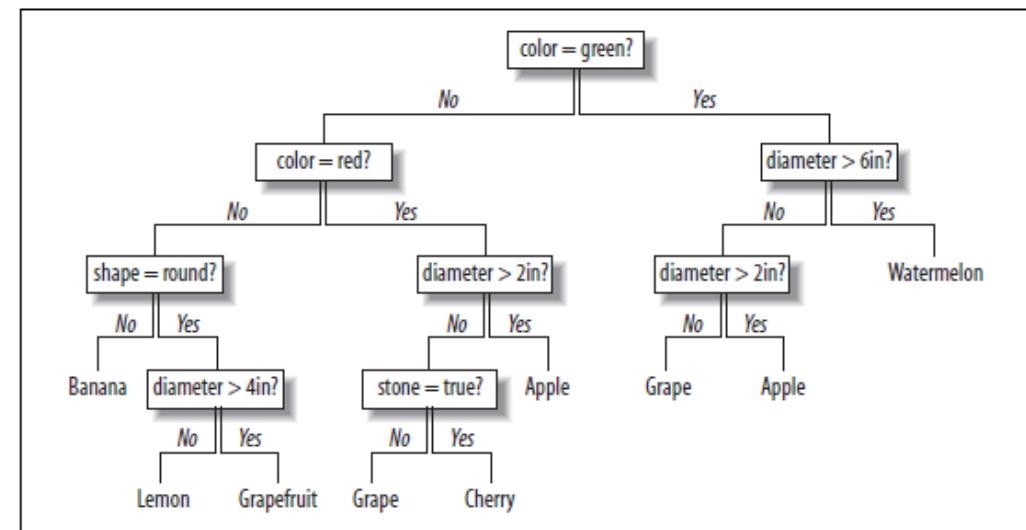
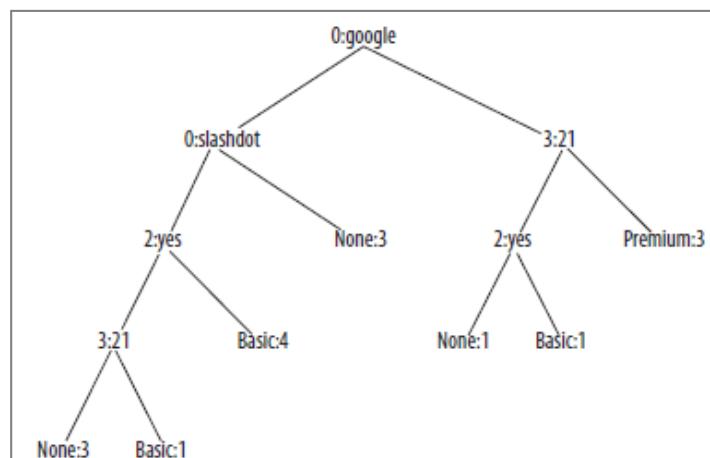


Figure 12-1. Example decision tree



Logistic Regression

Logistic regression 개요

=지도학습, 분류모델

- 회귀 : 데이터를 가장 잘 표현하는 최적선(Best Fit Line)을 그리는 것
- 로지스틱 회귀 : 데이터를 가장 잘 분류하는 Best Regression Coefficients 찾기

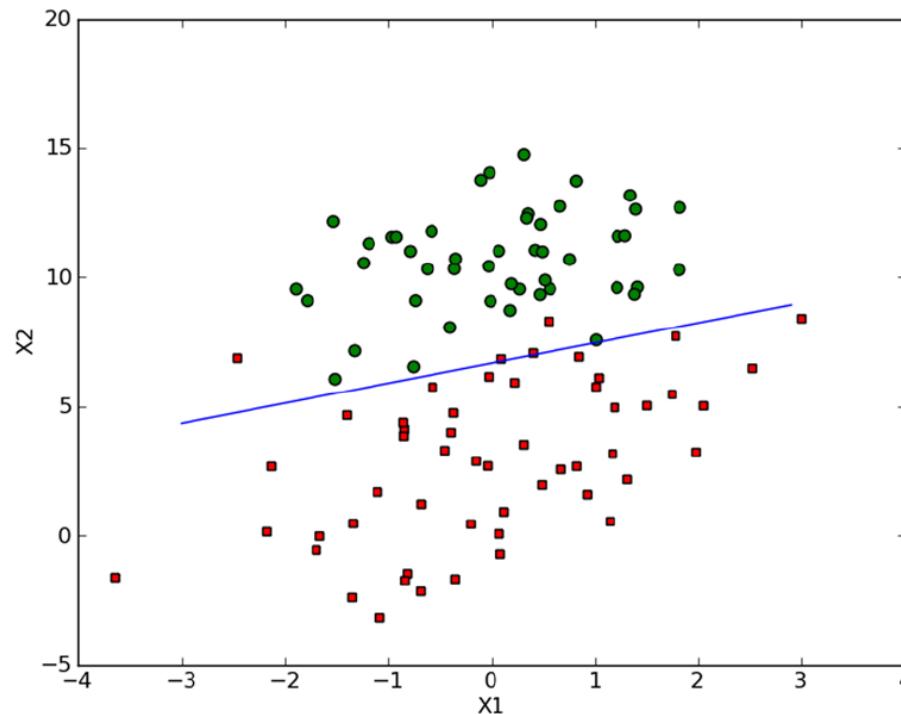


Figure 5.4 The logistic regression best-fit line after 500 cycles of gradient ascent



What is Clustering

● Clustering (클러스터링)

=비지도학습

- Task of grouping a set of objects in such a way that objects in the same group (called cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters)
- 클러스터 : 비슷한 특성을 가진 데이터들의 집단
- 데이터를 일일이 확인하지 않고 각 클러스터의 대표값을 확인해 전체 데이터의 특성을 파악할 수 있음



Clustering

군집 개요

=비지도학습

- 군집 : 유사한 속성을 가진 아이템들을 하나의 그룹으로 묶기
- UnSupervised Learning
- 일종의 자동 분류 시스템
 - > 군집의 다른 말은 비지도 분류(UnSupervised Classification)
- 훈련을 통해 각 아이템의 레이블을 생성한 후, 다양한 용도로 활용이 가능



Clustering =비지도학습

주요 군집 알고리즘 : Hierarchical vs K-Means

Table 12-7. Simple table for clustering

계층형 :
Hierarchical
Clustering

분류형 :
K-Means
Clustering

Item	P1	P2
A	1	8
B	3	8
C	2	6
D	1.5	1
E	4	2

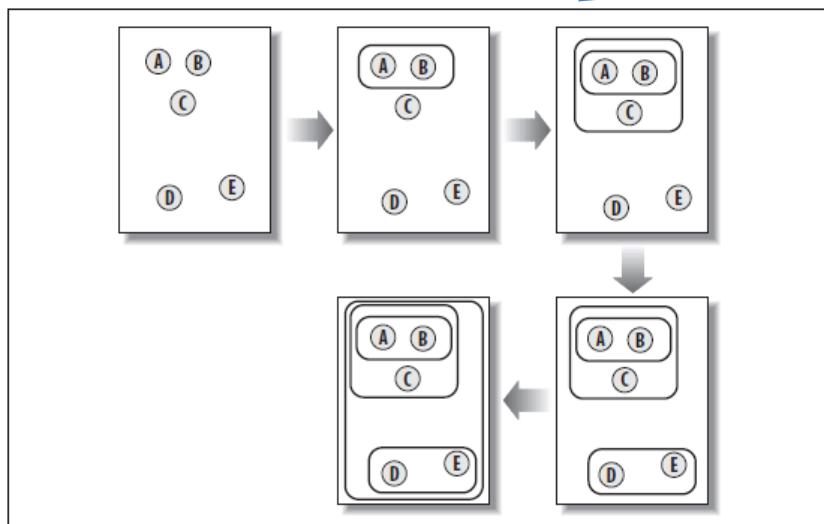


Figure 12-12. Process of hierarchical clustering

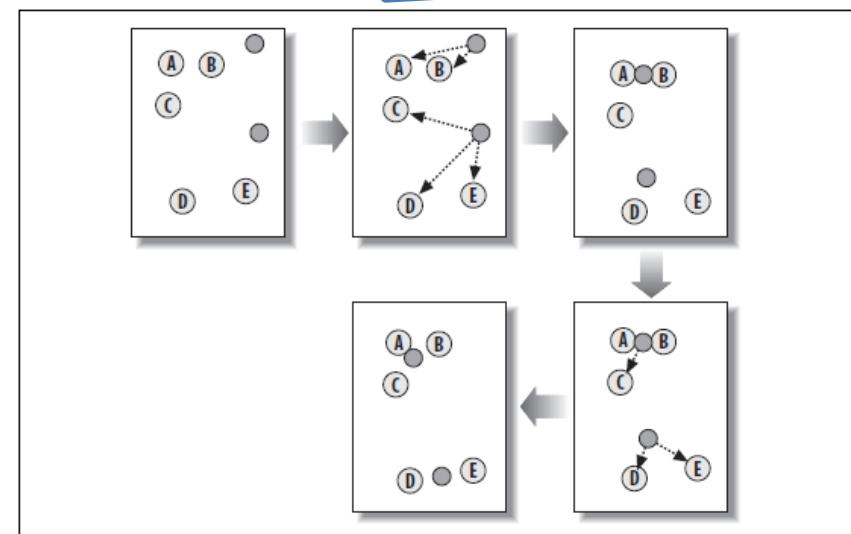


Figure 12-14. Process of K-means clustering

Hierarchical Clustering

Hierarchical Clustering 개요

=비지도학습

- 계층형 군집 : 이진 트리 형태로 유사한 아이템을 연속적으로 묶는 알고리즘
- 알고리즘

- 1.전체 아이템의 상호 거리를 계산한다.
- 2.가장 가까운 두개의 아이템을 찾는다.
- 3.두개의 아이템을 하나의 아이템으로 병합한다.
- 4.아이템이 1개만 남을때까지 1번에서 3번까지의 과정을 반복한다.

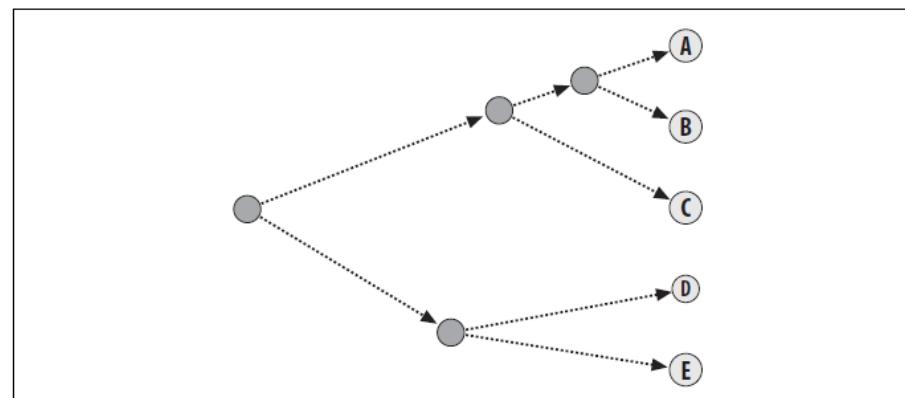


Figure 3-2. A dendrogram is a visualization of hierarchical clustering

Hierarchical Clustering

Hierarchical Clustering 개요

=비지도학습

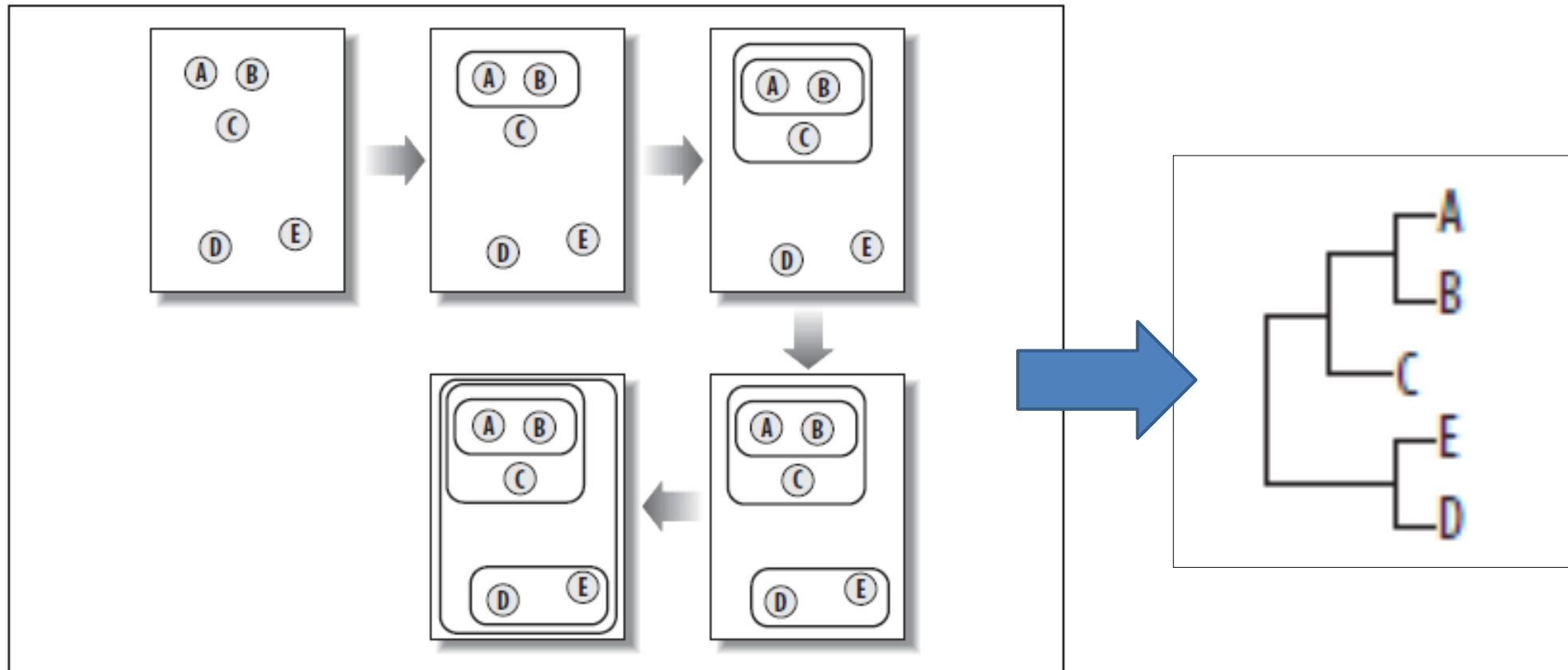


Figure 12-12. Process of hierarchical clustering



K-Means Clustering

K-Means Clustering 개요

=비지도학습

- K-Means : K개의 서로 다른 군집(그룹)을 찾는 알고리즘
- 알고리즘

1. 각 그룹의 중심이 되는 K개의 점(속성값)을 임의로 정한다.
2. 전체 아이템과 K개의 중심점과의 거리를 구한다.
3. 가장 가까운 그룹에 아이템들을 할당한다.
4. 각 그룹에 속한 아이템들의 중심점(K개)을 다시 계산한다.
5. 중심점(K개)의 값이 변하지 않을 때까지 2번에서 4번까지의 과정을 반복한다.
6. 후처리(옵션)



K-Means clustering

K-Means Clustering 개요

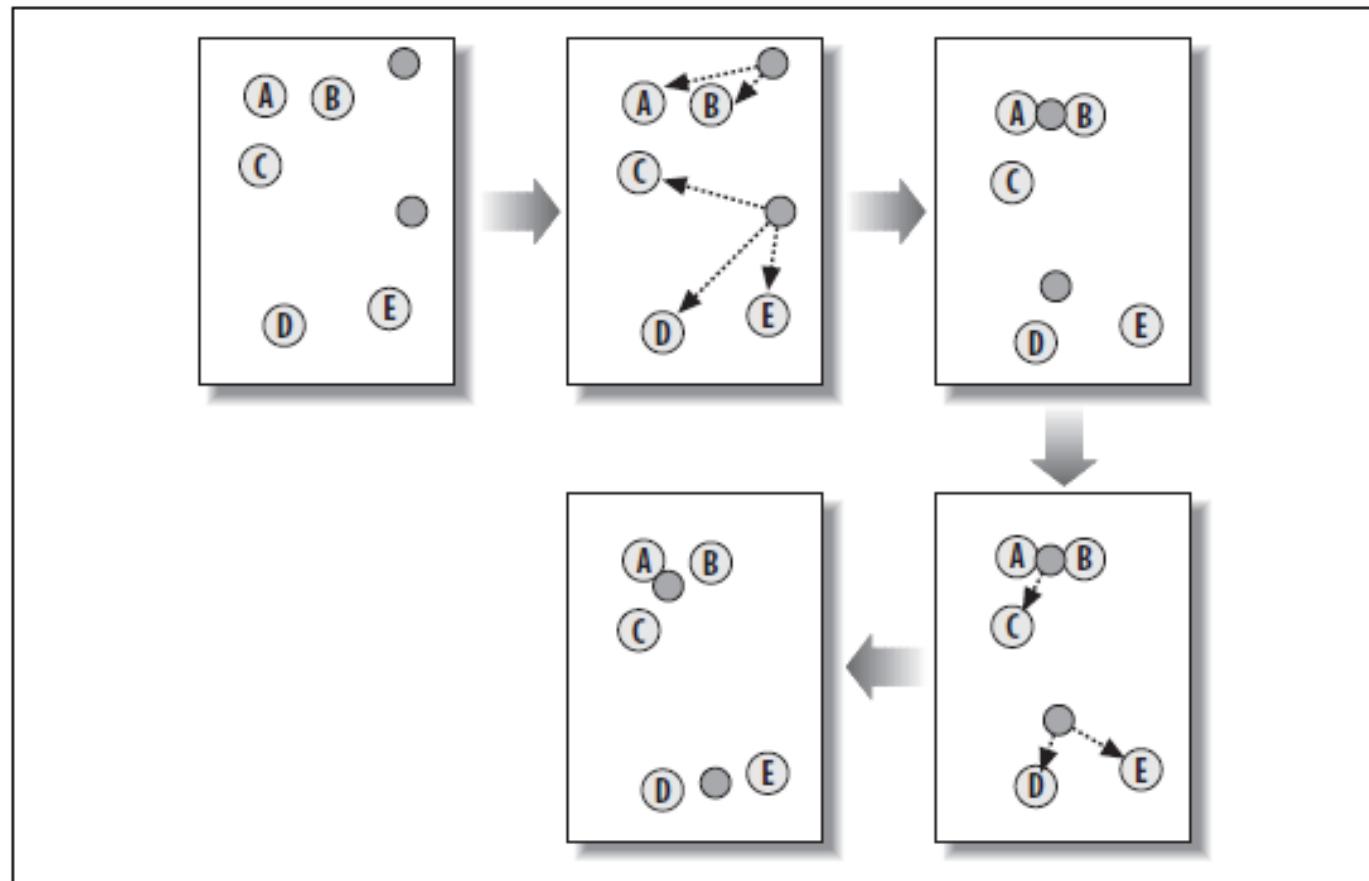


Figure 12-14. Process of K-means clustering



k-means Clustering

- k개의 cluster center를 찾고, 각 중심점에 가까운 데이터들을 묶는 방법

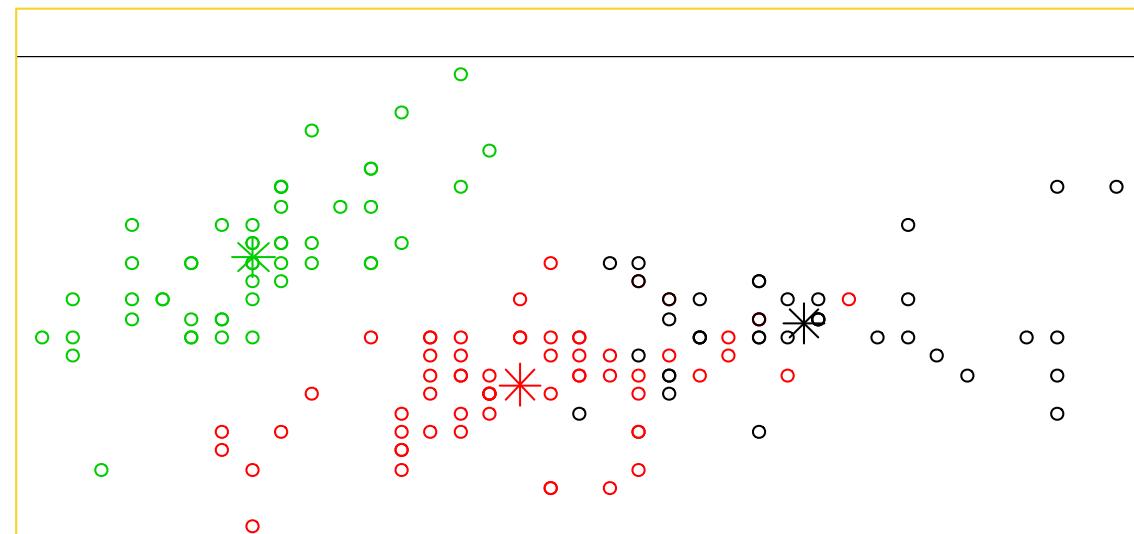
- mean : 평균

- 장점

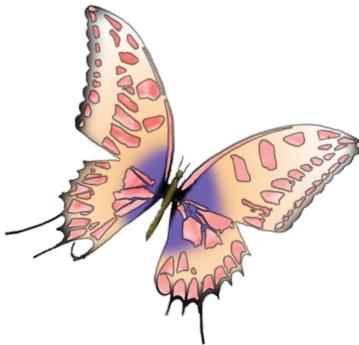
- 구조가 간단하고 많은 환경에서 빠르게 수렴. 대체로 다항 복잡도

- 단점

- k 값을 미리 정해주어야 함
 - 전역 최적값(Global Optimum)을 보장해주지 않음
 - 원형 이외의 데이터 분석이나, outlier, noise에 취약



실습: Machine Learning 기본



- 실습 : K-means 모델 구현
- iris 데이터 셋에서 Species를 제거하고 분류 모델을 만드시오.





모델평가 및 Anomaly Detection을 위한 클러스터링

모델평가를 위한 통계 리뷰
Supervised Model 이해 및 실습(Decision Tree)
Unsupervised Model 이해 및 실습(k-means)



모델평가를 위한 통계 리뷰

통계하면 떠오르는 이미지는?



통계는

인구, 경제 및 사회집단의 어떠한 현상을
주어진 목적에 따라 계량적으로
파악한 숫자

통계가 왜 필요한가?



WHY ?

- 어떤 상태 (Fact)의 파악, 역사기록 자료로서의 역할
- 정부, 기업, 가계부문에서 의사결정의 기초자료로서의 역할
- 지식기반 사회(**Knowledge-based Society**)의 도래로 지식이

중요한 자본이 되었음

모든 지식은 궁극적으로 역사가 된다.

모든 자연과학은 요약하면 수학이 된다.

그러나 지식을 얻는 방법은 통계학에 의해서다.

(C.R.Rao 교수의 1988년 발간 “Statistics and Truth” 서문에서)

→ 통계를 바탕으로 지식을 창출하고, 지식을 기초로 정책이 수립



통계란?

- 통계란 "모든 사회 및 자연현상을 나타내주는 의미를 가진 수치"이다.
- 개별자료로는 잘 알 수 없지만 많은 자료를 모아 서로 비교하면 하나의 현상이 명확하게 파악되어지고 이러한 통계를 이용하면 앞으로 일어날 상황도 예측할 수 있게 된다.
- 예를 들면, 인구, 국토면적, 평균 기온 및 강수량, 실업률, 국민소득, 학생 수 등과 같은 것들이다.



산수? 수학?

- 산수 쉬운 것?

2015 고등수학 교육과정

고등학교 2,3학년(문과): 미적분1, 확률과 통계

- 수학 어려운 것?

미적분1

확률과 통계

- 산수는 수학의 한 분야.

I 수열의 극한

1. 수열의 극한
2. 급수

I 순열과 조합

1. 경우의 수
2. 순열과 조합
3. 분할
4. 이항정리

- 통계 마지막에 배우니까 제일

II 함수의 극한

1. 함수의 극한
2. 함수의 연속

II 확률

1. 확률의 뜻과 활용
2. 조건부확률

III 다항함수의 미분법

1. 미분계수
2. 도함수
3. 도함수의 활용

III 통계

1. 확률분포
2. 통계적 추정

- 모든 것이 깊게 들어가면 다 어렵다.

IV 다항함수의 적분법

1. 부정적분
2. 정적분의 활용

통계학?



- 통계학이란?
 - 미래에 대한 의사결정 결과의 불확실성을 추정하는 문제에 많이 이용 - 통계학은 의사결정의 결과를 사전에 예측하여 의사결정을 도움.
 - 결정에 따른 위험을 최소화하는 것
- 경영학에서 데이터를 분석하는 목적에 따른 통계학의 분류
 - 기술통계 (Descriptive Statistics)
 - 상세한 분석을 수행하기 전에 수집된 데이터를 묘사하고 정리하기 위한 방법
 - 정리된 데이터를 의사결정자에게 보고할 목적으로 수행하는 통계방법
 - 추론통계 (Inferential Statistics)
 - 표본으로부터 획득된 정보(표본 통계량)을 바탕으로 전체 집단의 특성에 대한 결론을 내리는 (추론적 의사결정) 것을 목적으로 하는 통계분석 방법

기본 용어



- 도수(Frequency): 도수는 어떤 특정한 그룹이나 범위 안에 얼마나 많은 항목이 들어 있는지 나타내는 값입니다. 항목의 수를 센 값이라고 하네요.
- 범주적 데이터(Categorical data): 일정한 범주로 나눈 다음 각 범주의 성질이나 특징을 묘사하는 데이터입니다. 정성적 데이터라고도 합니다.
- 수치적 데이터(Numerical data): 숫자를 다루는 데이터로서 값의 측정이나 개수처럼 숫자로서의 의미를 갖는다고 합니다. 정량적 데이터라고도 합니다.
- 도수밀도(Frequency density): 그룹으로 묶인 데이터의 도수가 얼마나 집중되어 있는지 나타냅니다. "Frequency / Group width"로 계산합니다.
- 히스토그램(Histogram): 그룹으로 묶인 데이터를 위한 차트로 각 막대의 높이는 도수가 아니라 도수밀도의 값을 나타냅니다. (막대 사이의 빈 공간이 없는 막대그래프를 생각하면 됩니다.)
- 누적도수(Cumulative frequency): 특정 값에 이르기까지의 도수의 합을 의미합니다. 즉 누적도수는 도수의 누적 합계입니다.



쉽게 수학 공부하는 방법은?

- 용어만이라도 익히자.
 - 통계용어·지표의 이해 : [http://kostat.go.kr/
file_total/2013_korki_1_total.pdf](http://kostat.go.kr/file_total/2013_korki_1_total.pdf)
- 실제 작업은?
 - 통계관련 프로그램으로 해결!! . ex) Excel,
R, Pandas, Numpy



지수

- 개수
- 합계
- 평균
- 표준편차
- 분산
- 최소값
- 중앙값
- 최대값



기본 통계 지식

- 대표값 : 산술평균, 기하평균, 조화평균, 가중평균, 중위수, 최빈수
- 산포도 : 편차, 분산, 표준편차, 변이계수
- 비·비율·율
- 퍼센트와 퍼센트포인트
- 변동률 : 전년동월비, 전월비
- 기여율·기여도
- 표본조사
- 오차 : 표본오차, 비표본오차
- 신뢰도
- 지수 : 기준시점, 가중치, 지수산식



기능에 따른 분류

기술통계

추리통계

관찰, 측정된 잡다한 개개의 현상을 전체적
파악 및 간략하게 기술해 주는 기능

관찰, 측정된 소수의 결과로 관찰되지 않은
전반적인 현상에 대해서
일반적인 결론으로 일반화 시키는 기능

방법: (수량적 자료제시)

빈도분포, 백분율, 표준편차, 평균, 상관관계 등

T검정, 분산분석, 회귀분석 등이 해당
(추리통계는 통계적 방법에 따라
모수적 통계와 비모수적 통계가 있다)

모집단 분포의 가정에 따른 종류

모수적 추리통계

(parametric statistics)

정규 분포 가정

모집단의 특성을 추정,

측정치의 연속성과 등간격성

(선형성), 모집단의 어떤 특성의

정규분포성, 분산의 동질성의 조건

을 가지고 있어야 함

비모수적 추리통계

(nonparametric statistics)

모집단의 특성의 분포에 어떤

가정도 하지 않은 상태

선형성, 정규분포성 등의 조건이

없어도 됨

가설 검정력이 더 정밀함.



변인의 수에 따른 분류

일원적 통계분석

다원적 통계분석

하나의 변인만을
분석하는 통계

둘 이상의 변인을
동시에 분석하는 통계

(2) 표본의 크기

모집단의 크기,
이질성 또는
동질성의 정도
신뢰도 범위 통계
방법 등에
영향을 받는다.

모집단 의 크기	적정표본의 크기							
	95% 신뢰도 수준에서의 허용표집오차				99% 신뢰도 수준에서의 허용표집오차			
	±1%	±2%	±3%	±5%	±1%	±2%	±3%	±5%
1,000	-	-	473	244	-	-	-	360
3,000	-	1,206	690	291	-	-	1,021	470
5,000	-	1,437	760	303	-	2,053	1,182	508
10,000	4,465	1,678	823	313	-	2,584	1,341	527
20,000	5,749	1,832	858	318	8,213	2,967	1,437	542
50,000	6,946	1,939	811	321	10,898	3,257	1,502	551
100,000	7,465	1,977	888	321	12,231	3,367	1,525	554
500,000	7,939	2,009	895	322	13,557	3,460	1,544	557



변인(변수, Variable)

변인

- 연구대상이 되고 있는 집단 구성원이 총성(계층 성격)에 있어서 서로 구별 짜어질 수 있을 때 이 속성을 변인

구성원

- 관심이 되는 분석의 단위를 의미

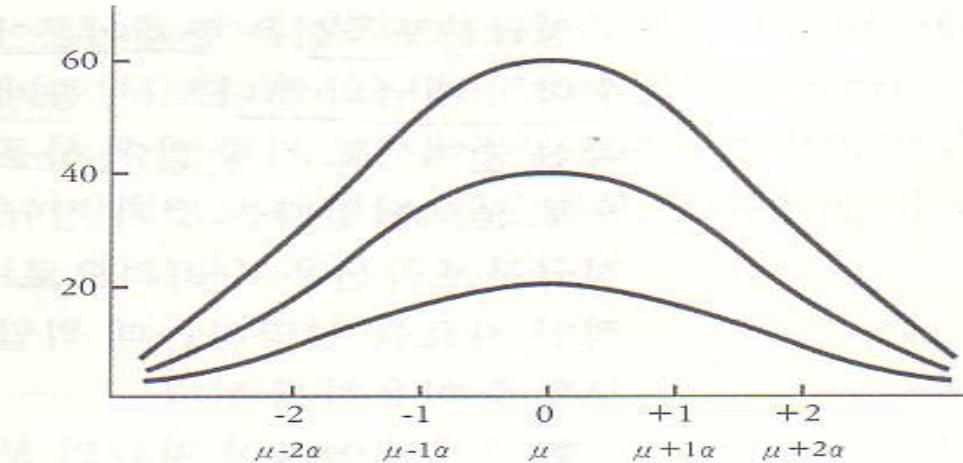
예)) 만약 개인이라면.....

- 성별, 연령, 학교, 종교 등이 변인



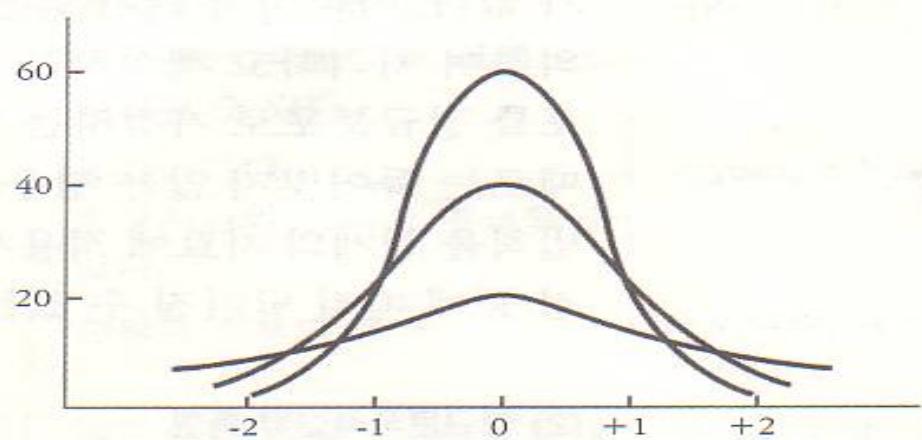
정규분포

평균과 표준편차가 다르고, 곡선
안에 포함된 면적이 다른 경우에
정규분포는 여러 가지 형태를
이루게 됨



(평균과 표준편자는 같으나 사례수가 다른 경우)

사례수 N 이 변하면 이 곡선의
모양은 좌우로 퍼지게 되나 전체적
모양은 변하지 않고, 표준편차가 커지
며
그 곡선의 형태를 변화시켜 곡선의 모양
이 평평하게 된다.



(평균과 사례수는 같으나 표준편자가 다른 경우)



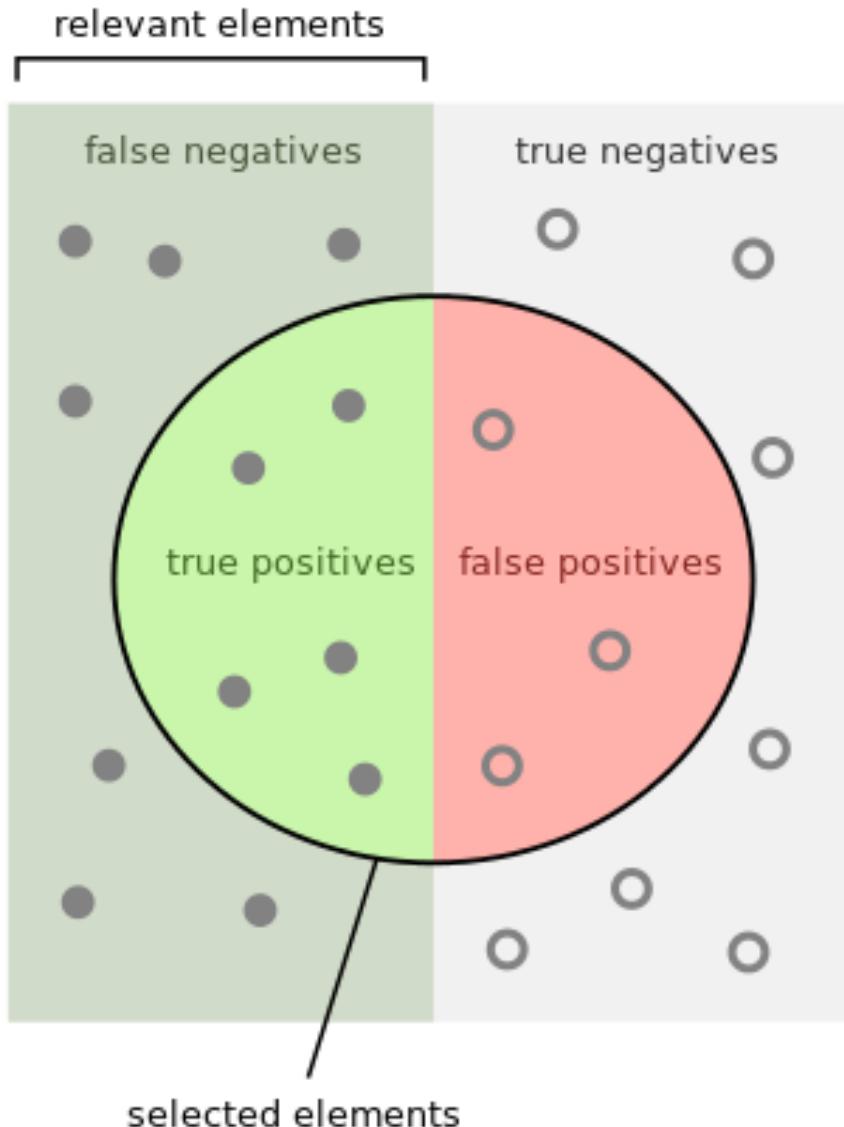
모델 평가

- 훈련 데이터로부터 하나의 함수가 유추되고 나면 해당 함수에 대한 평가를 통해 파라미터를 최적화
- 이러한 평가를 위해 교차 검증(Cross-Validation)이 이용되며 이를 위해 검증 집합을 다음의 세가지로 나눔
 1. 훈련 집합 (A Training Set)
 2. 검증 집합 (A Validation Set)
 3. 테스트 집합 (A Test Set)

		실제 결과 / 분류	
		참	거짓
추론된 결과 / 분류	참	TP (True Positive)	FP (False Positive)
	거짓	FN (False Negative)	TN (True Negative)



Confusion Matrix



How many selected items are relevant?

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

		실제 결과 / 분류	
		참	거짓
추론된 결과 / 분류	참	TP (True Positive)	FP (False Positive)
	거짓	FN (False Negative)	TN (True Negative)

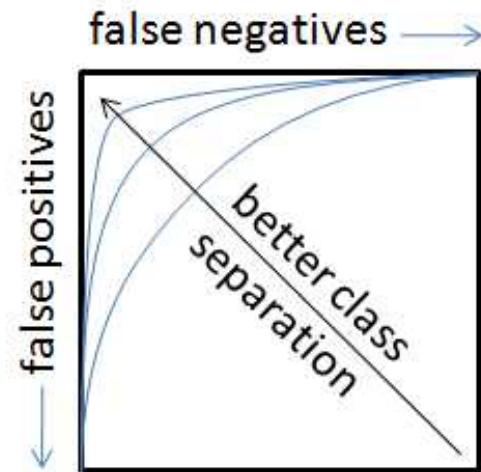


정밀도(Precision), 재현율(Recall)

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

- ROC curve(Receiver Operating Characteristic curve)
 - TPR과 FPR을 각각 x,y축으로 놓은 :





공돌이의 수학정리노트

- 공개 책을 참고 하세요.
- Link : <https://wikidocs.net/book/563>
- Youtube : <https://www.youtube.com/user/AngeloYeo/videos>

wikidocs.net/4050

2) Chebyshev 필터
3) frequency transformation
[7] 헐버트 변환
[8] 시간 주파수 불확정도
4. 선형대수학
[01] 행렬식의 기하학적 의미
[02] 고유값과 고유벡터의 기하학적
[03] Jacobian 행렬
[04] 선형 비동차 미분방정식의 일반
[05] 서포트 벡터 머신(Support Vect...
[06] Linear Discriminant Analysis
[07] 론스키안(Wronskian)에 관한
[08] 상관 계수와 벡터의 내적

“그렇다면, 그 벡터의 크기는 얼마만큼 변했나요?”

3. 예시를 통한 Eigenvector와 Eigenvalue, 그리고 선형 변환에 대한 간략한 이해

다음과 같은 행렬 A를 생각해보자.

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

이 행렬에 대해 Eigenvalue와 Eigenvector를 구해보도록 하자.

정의에 따라, eigenvalue λ 와 eigenvector K 는 다음과 같은 식을 만족한다.

$$AK = \lambda K$$

그러므로, 행렬의 성질에 의해 $(A - \lambda I)K = 0$ 이다. 또한, 벡터 K 가 nontrivial solution을 갖기 위해서는 다음이 만족해야 한다.

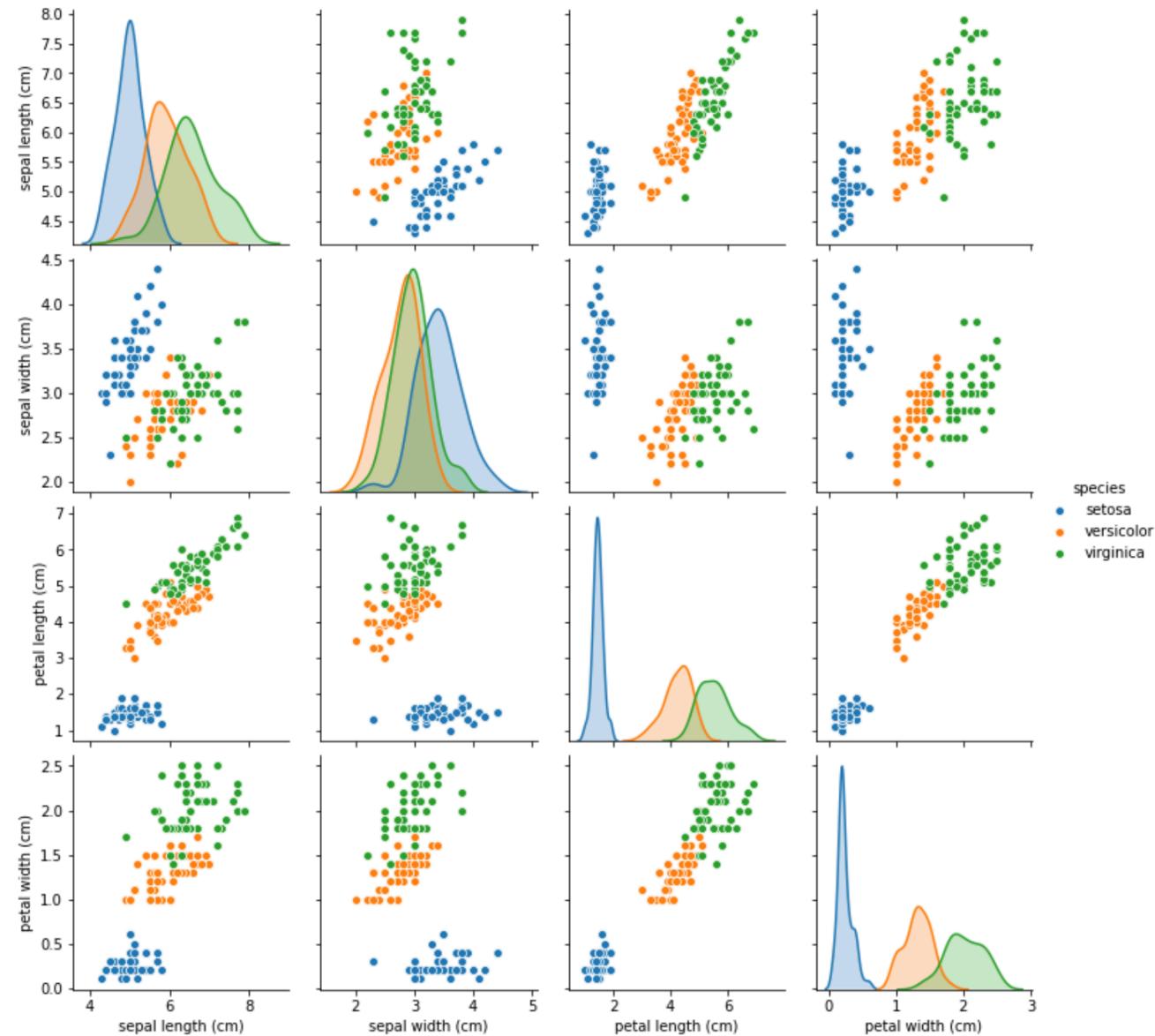
$$\det(A - \lambda I) = 0$$

그러므로,



Supervised Model 이해 및 실습(Decision Tree)

Scatter Plot으로 데이터 분포 확인





의사결정나무분석의 특징

장점	단점
<p>해석의 용이성</p> <ul style="list-style-type: none"> - 나무구조로 표현되어 모형을 사용자가 쉽게 이해할 수 있다. 	<p>비연속성</p> <ul style="list-style-type: none"> - 연속형 변수를 비연속적 값으로 취급하기 때문에 분리의 경계점 부근에서 예측오류가 클 가능성이 있다.
<p>교호작용효과의 해석</p> <ul style="list-style-type: none"> - 두 개 이상의 변수가 결합하여 목표변수에 어떻게 영향을 주는지 쉽게 알 수 있다. 	<p>선형성 또는 주효과의 결여</p> <ul style="list-style-type: none"> - 선형모형에서는 주효과는 다른 예측변화와 관련시키지 않아도 각 변수의 영향력을 해설 할 수 있는데 의사나무는 그렇지 않다.
<p>비모수적 모형</p> <ul style="list-style-type: none"> - 선형성(linearity), 정규성(normality) 또는 등분산성 (equal variance) 등의 가정이 필요하지 않다. 	<p>비안정성</p> <ul style="list-style-type: none"> - 분석용 자료(training data)에만 의존하므로 새로운 자료의 예측에서는 불안정(unstable) 할 가능성이 높다.



Decision Tree 장점

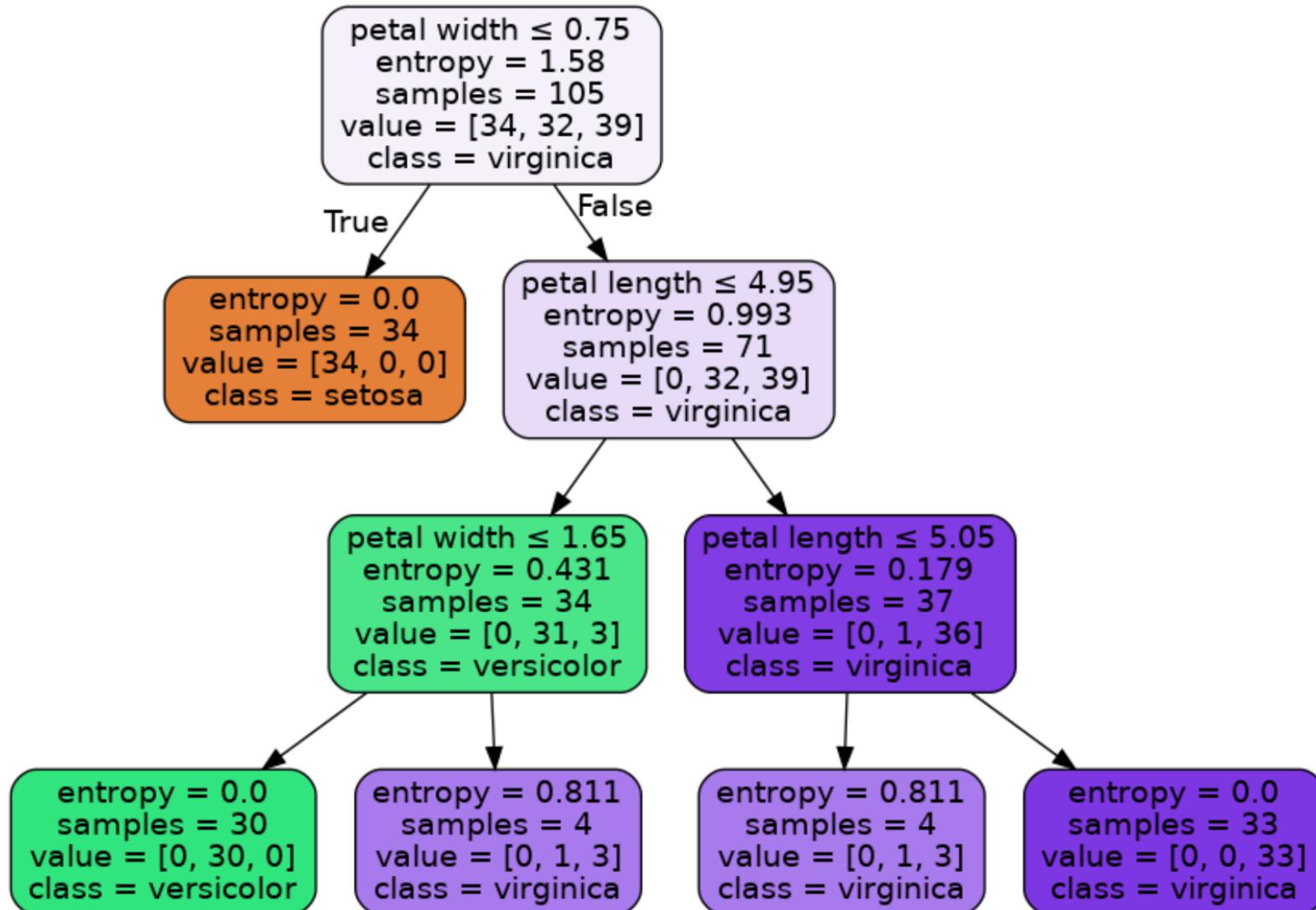
- 결과를 해석하고 이해하기 쉽다
 - 간략한 설명만으로 결정 트리를 이해하는 것이 가능하다.
- 자료를 가공할 필요가 거의 없다
 - 다른 기법들의 경우 자료를 정규화하거나 임의의 변수를 생성하거나 값이 없는 변수를 제거해야 하는 경우가 있다.
- 수치 자료와 범주 자료 모두에 적용할 수 있다
 - 다른 기법들은 일반적으로 오직 한 종류의 변수를 갖는 데이터 셋을 분석하는 것에 특화되어 있다.
- 화이트박스 모델을 사용한다.
 - 모델에서 주어진 상황이 관측 가능하다면 불 논리를 이용하여 조건에 대해 쉽게 설명할 수 있다.
- 안정적이다.
 - 해당 모델 추리의 기반이 되는 명제가 다소 손상되었더라도 잘 동작한다.
- 대규모의 데이터 셋에서도 잘 동작한다.
 - 방대한 분량의 데이터를 일반적인 컴퓨터 환경에서 합리적인 시간 안에 분석할 수 있다.



Decision Tree 한계

- 결정 트리 학습 알고리즘은 각 노드에서의 부분 최적값을 찾아내는 탐욕 알고리즘 같은 휴리스틱 기법을 기반으로 하고 있다.
 - 최적 결정 트리를 알아낸다고 보장할 수는 없다.
- 결정 트리 학습자가 훈련 데이터를 제대로 일반화하지 못할 경우 너무 복잡한 결정 트리를 만들 수 있다.
 - 이 문제를 해결하기 위해서 가지치기 같은 방법을 사용하여야 한다.
- 결정 트리로는 배타적 논리합이나 패리티, 멀티플렉서와 같은 문제를 학습하기 어렵다.
 - 표현 방법을 바꾸거나 통계 관련 학습법이나 귀납 논리 프로그래밍처럼 더 많은 것을 표현할 수 있는 학습 알고리즘을 사용하여야 한다.
- 각각 서로 다른 수의 단계로 분류가 가능한 변수를 포함하는 데이터에 대하여 더 많은 단계를 가지는 속성 쪽으로 정보 획득량이 편향되는 문제가 있다.
 - 이 문제는 조건부 추론을 통해 해결이 가능하다.
- 데이터의 특성이 특정 변수에 수직/수평적으로 구분되지 못할 때 분류율이 떨어지고, 트리가 복잡해지는 문제가 발생한다.
 - 신경망 등의 알고리즘이 여러 변수를 동시에 고려하지만 결정트리는 한 개의 변수만을 선택하기 때문에 발생하는 당연한 문제이다.
- 약간의 차이에 따라 (레코드의 개수의 약간의 차이) 트리의 모양이 많이 달라질 수 있다.

Decision Tree 적용 예





Unsupervised Model 이해 및 실습(k-means)

Jupyter Lab의 Interact란?

- 사용자로부터 입력을 받을 수 있게 해주는 Jupyter UI Component
- 사용해야 하는 이유 : 잘은 Hyper parameter

변경

```
from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
def f(x):
    return x
interact(f, x=10);
```



19

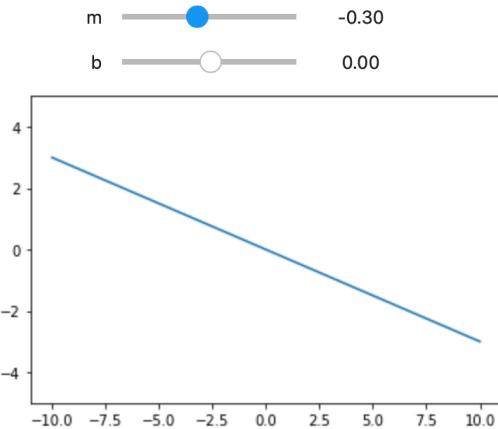
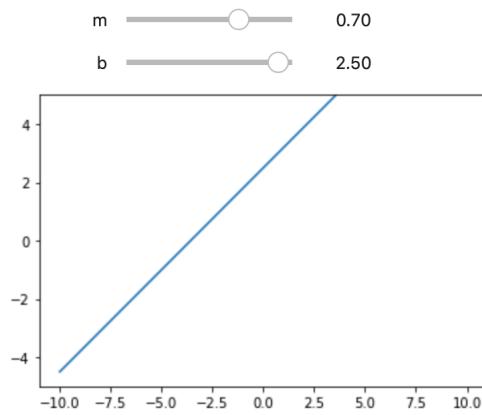
19

Jupyter Lab의 Interact 사용 예

```
%matplotlib inline
from ipywidgets import interactive
import matplotlib.pyplot as plt
import numpy as np

def f(m, b):
    plt.figure(2)
    x = np.linspace(-10, 10, num=1000)
    plt.plot(x, m * x + b)
    plt.ylim(-5, 5)
    plt.show()

interactive_plot = interactive(f, m=(-2.0, 2.0), b=(-3, 3, 0.5))
output = interactive_plot.children[-1]
output.layout.height = '350px'
interactive_plot
```





Jupyter Lab의 Interact 설치 방법

nodejs 설치

```
curl -sL https://deb.nodesource.com/setup_10.x -o nodesource_setup.sh  
bash nodesource_setup.sh  
apt-get install -y nodejs
```

Jupyter widgets 설치

```
pip install ipywidgets  
jupyter nbextension enable --py widgetsnbextension  
jupyter labextension install @jupyter-widgets/jupyterlab-manager  
initctl restart jupyter-server --no-wait
```

- 그밖에

1. 아래 명령 실행 후 브라우저 Refresh 필요.
2. Menu → Settings → Extension Manager 클릭해서 Extension Manager 활성화 필요.

Jupyter Lab의 Interact의 Reference

- <https://ipywidgets.readthedocs.io/en/latest/examples/Using%20Interact.html>

The screenshot shows a web browser displaying the Jupyter Widgets documentation. The URL in the address bar is ipywidgets.readthedocs.io/en/latest/examples/Using%20Interact.html. The page title is "Using Interact". On the left, there is a sidebar with a search bar and a navigation tree under "User Guide". The "Using Interact" section is expanded, showing sub-sections like "Basic interact", "Fixing arguments using fixed", "Widget abbreviations", etc. The main content area starts with a heading "Using Interact" and a paragraph explaining that the `interact` function automatically creates UI controls for exploring code and data interactively. It includes a code snippet:

```
[1]: from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
```

Below this, there is a section titled "Basic `interact`" with a paragraph explaining how it generates UI controls for function arguments. It includes another code snippet:

```
[2]: def f(x):
    return x
```

Finally, there is a note about passing a function to `interact` with an integer keyword argument, which generates a slider.

K-means 알고리즘



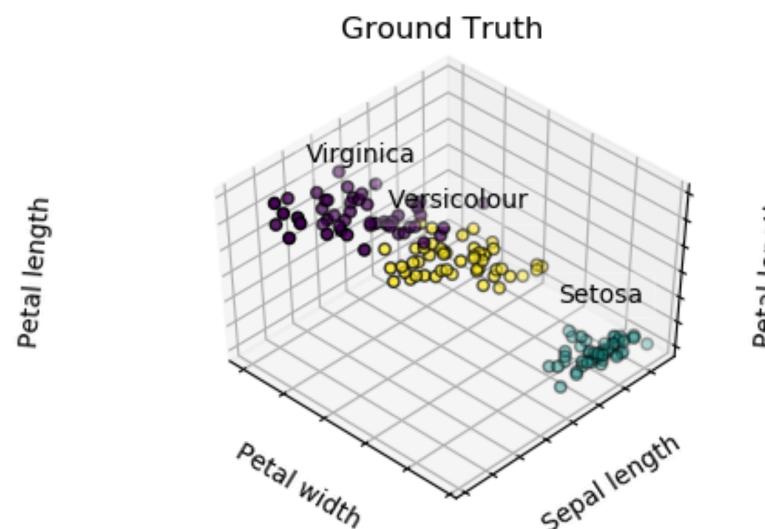
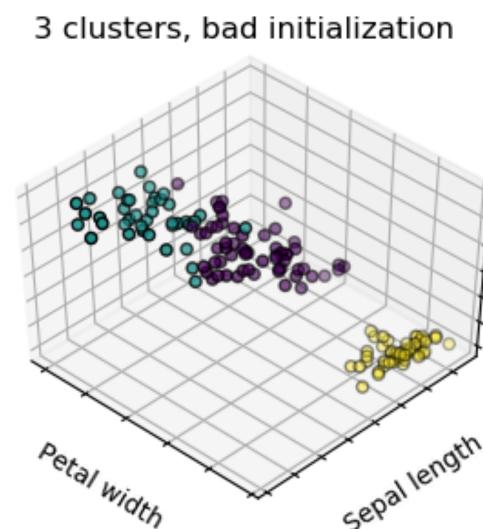
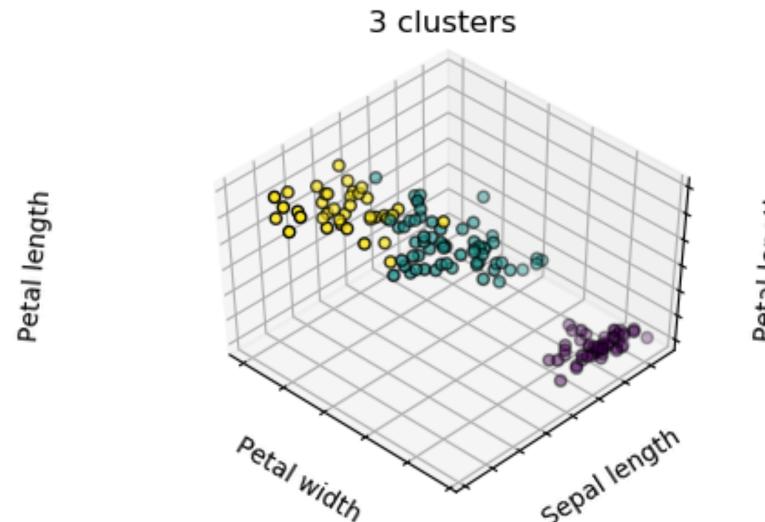
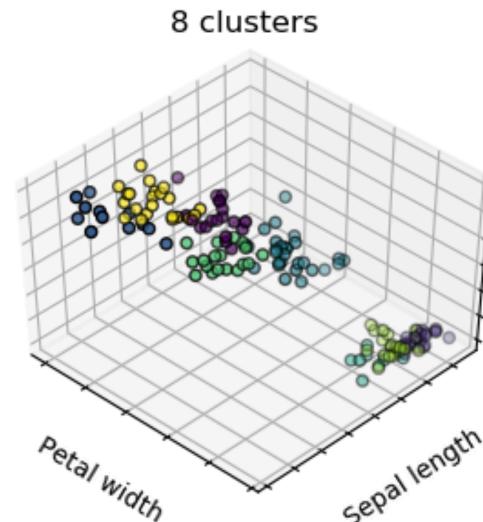
1. 임의의 중심위치 $\mu_k (k=1, \dots, K)$ 를 고른다. 보통 데이터 표본 중에서 K 개를 선택한다.
2. 모든 데이터 $x_i (i=1, \dots, N)$ 에서 각각의 중심위치 μ_k 까지의 거리를 계산한다.
3. 각 데이터에서 가장 가까운 중심위치를 선택하여 각 데이터가 속하는 군집을 정한다.
4. 각 군집에 대해 중심위치 μ_k 를 다시 계산한다.
5. 2 ~ 4를 반복한다.
 - K-평균 군집화란 명칭은 각 군집의 중심위치를 구할 때 해당 군집에 속하는 데이터의 평균(mean)값을 사용하는데서 유래하였다. 만약 평균 대신 중앙값(median)을 사용하면 K-중앙값(K-Median) 군집화라 한다.

scikit-learn의 KMeans 클래스

- **n_clusters**: 군집의 갯수
- **init**: 초기화 방법. "random"이면 무작위, "k-means++"이면 K-평균++ 방법. 또는 각 데이터의 군집 라벨
- **n_init**: 초기 중심위치 시도 횟수. 디폴트는 10이고 10개의 무작위 중심위치 목록 중 가장 좋은 값을 선택
- **max_iter**: 최대 반복 횟수
- **random_state**: 시드값



k-means에서 k개 구하기-1

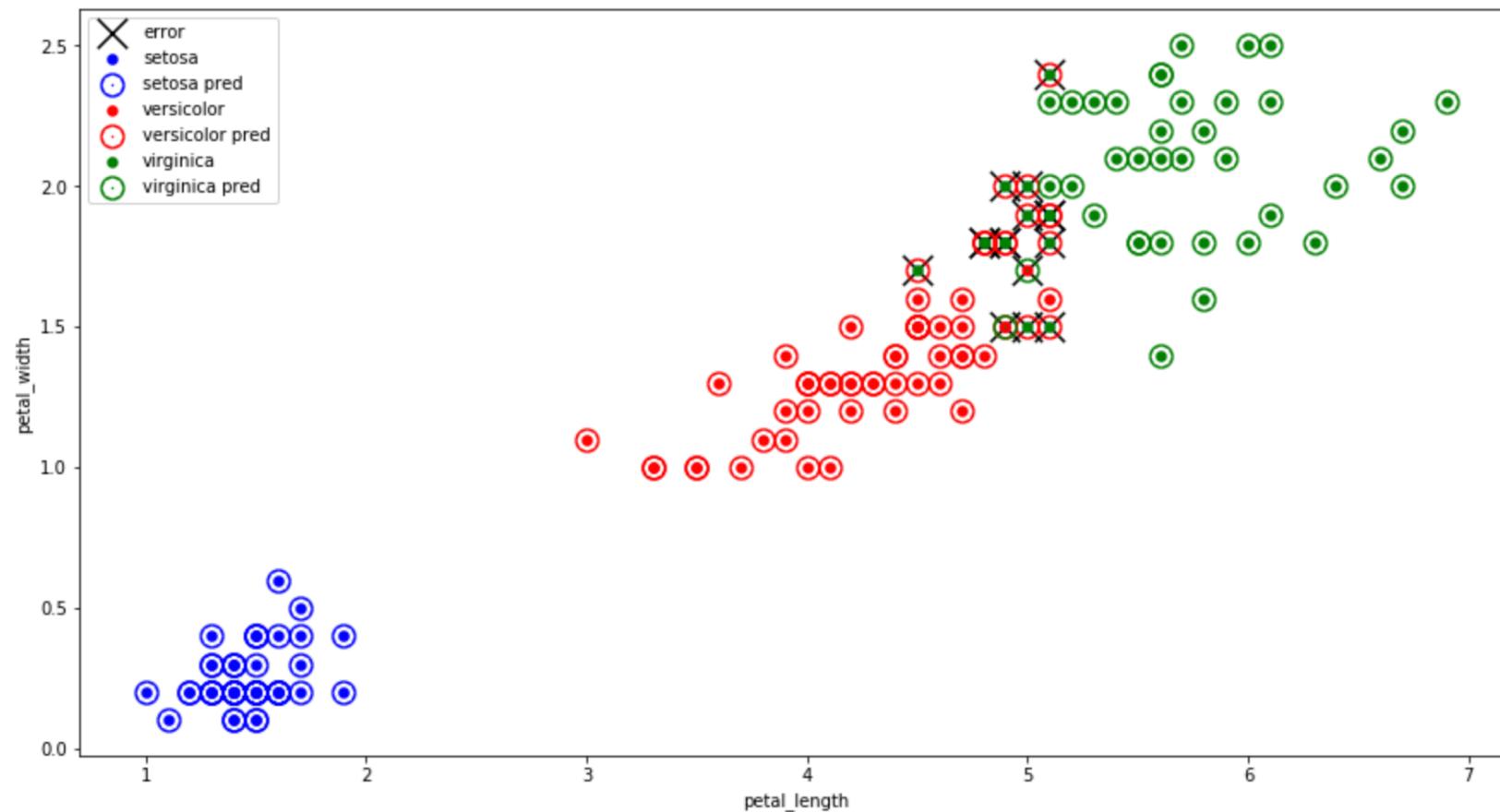


Interact를 통한 k-means의 k 구하기

```
interact(render_plot, x_col=x_columns, y_col=x_columns, k=ks)
```

x_col petal_length
y_col petal_width
k 3

accuracy 0.89





데이터 전처리

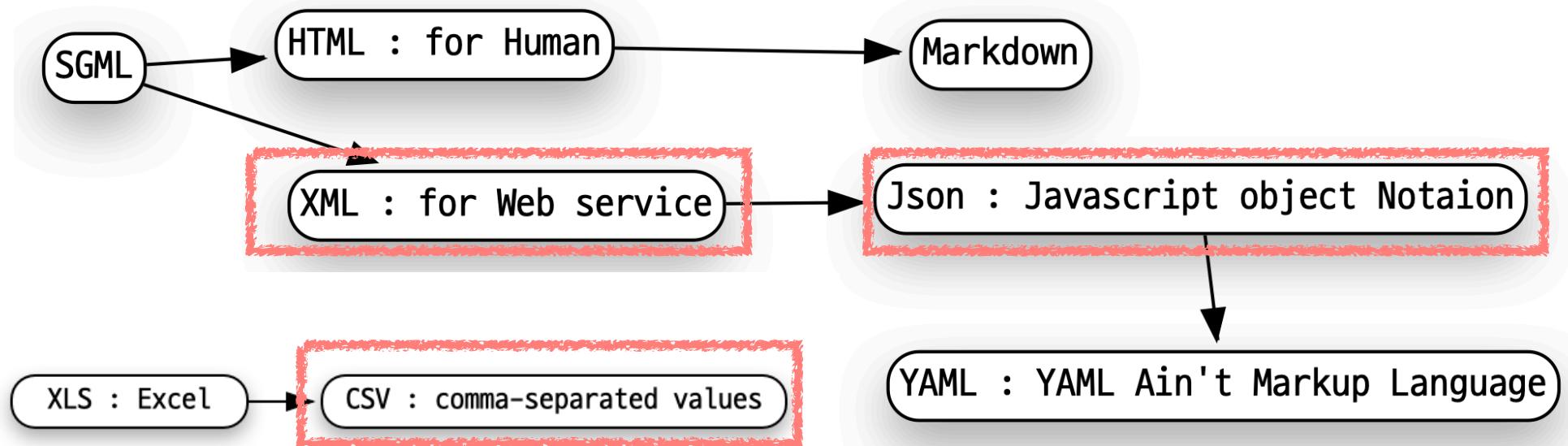
데이터 파일 종류별 이해
파싱 실습
인코더 만들기 실습



데이터 파일 종류별 이해



File Format





XML

- eXtention Markup Language.
- Web Service에서 데이터 교환용
- 단점 : 작성 불편하고, 크기도 큼
- Example

```
<xml>
<raw><id>1</id><name>aaa</name></raw>
<raw><id>2</id><name>bbb</name></raw>
</xml>
```



JSON

- JavaScript Object Notation
- Web Service에서 데이터 교환용
- 장점 : XML에 비해 크기 작음
Python이나 Javascript에서 별도의 파싱과정 없이 사용
- Example:

```
[  
  {"id":1 , "name":"aaa"},  
  {"id":2 , "name":"bbb"}  
]
```



CSV

- Comma-Separated Values
- 일반적으로 많이 사용되는 데이터 파일
- 단점 : ,(comma)가 포함된 문자열에 취약
- 장점 : Excel에서 CSV내보내기 가능
- Example

```
id, name
1, aaa
2, bbb
```



그밖의 Data file format

- TSV : Tab-Separated Values
- Parquet :
 - 나무조각을 붙여넣은 마룻바닥이라는 뜻
 - Binary format
 - 속도, 용량 이득
- ORC :
 - Optimized Row Columnar
 - for Hive 속도
- libsvm:
 - ex) aa,null,cc → 1:aa,3:cc
 - Sparse Matrix를 처리시 유리함
 - Machine Learning에서 많이 사용



Pandas IO tools (text, CSV, HDF5, ...)

- https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html

Format Type	Data Description	Pandas Reader	Pandas Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	Fixed-Width Text File	read_fwf	(tsv)
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
-	MS Excel	read_excel	to_excel
binary	OpenDocument	read_excel	-
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	ORC Format	read_orc	-
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	-
binary	SPSS	read_spss	-
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google BigQuery	read_gbq	to_gbq



파상 실습

Scikit-Learn의 전처리 기능

- 스케일링(Scaling): 자료의 크기 조정
 - scale, StandardScaler
 - normalize, Normalizer
 - minmax_scale, MinMaxScaler
 - maxabs_scale, MaxAbsScaler
 - robust_scale, RobustScaler
- 인코딩(Encoding): 카테고리 값의 정수 표현
 - binarize, Binarizer
 - label_binarize, LabelBinarizer
 - LabelEncoder
 - OneHotEncoder
 - DictVectorizer
- Imputation: 결손 데이터(missing data) 처리
 - Imputer
- Transform: 데이터 변환
 - PolynomialFeatures
 - FunctionTransformer



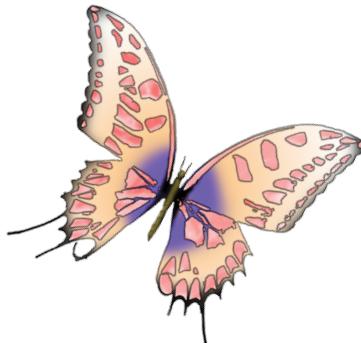
Json Parsing Example

```
>>> x="""[  
...     {"id":1 , "name":"aaa"},  
...     {"id":2 , "name":"bbb"}  
... ]  
... """  
>>> xx=eval(x)  
>>> for i in xx:  
...     print("%d,%s"%(i["id"],i["name"]))  
...  
1,aaa  
2,bbb
```

File Download Example by Python

```
>>> import requests  
>>> url = 'http://j.fintra.com/_file/dept.json'  
>>> response = requests.get(url)  
>>> print(response.text)  
[  
{"deptno":10 , "dname":"ACCOUNTING" , "loc":"NEW YORK" },  
 {"deptno":20 , "dname":"RESEARCH" , "loc":"DALLAS" },  
 {"deptno":30 , "dname":"SALES" , "loc":"CHICAGO" },  
 {"deptno":40 , "dname":"OPERATIONS" , "loc":"BOSTON" }  
]
```

파일 받아서 파싱하기 실습



- http://j.fintra.com/_file/dept.json 의 파일 받아서 dname 컬럼을 모두 출력 하시오.
- 이전 페이지의 2개 예제를 참고 하시오.





인코더 만들기 실습



Encoder vs. Decoder

- Encoder란?
 - 일반적 의미 : 물리 환경으로부터 피드백을 제공하는 센서 장치
 - 데이터 과학에서의 의미 : 분석을 위해 데이터를 **부호화** 함.
- Decoder란?
 - Encoding된 데이터를 원래의 형태로 복원함(**복호화**).

Encoder가 만들어지는 과정

["ACCOUNTING", "RESEARCH", "SALES", "RESEARCH", "RESEARCH"]

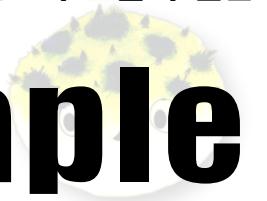
↓ set operation

['ACCOUNTING', 'RESEARCH', 'SALES']

[0, 1, 2]

↓

{ 'ACCOUNTING':0, 'RESEARCH':1, 'SALES':2 } **인코더**



Encoder vs. Decoder Example

- Encoder

```
{'ACCOUNTING':0, 'RESEARCH':1, 'SALES':2}
```

- Decoder

```
{0:'ACCOUNTING', 1:'RESEARCH', 2:'SALES'}
```

Encoder Example Coder

```
>>> data=["ACCOUNTING", "RESEARCH", "SALES", "RESEARCH",
"RESEARCH"]
>>> enc= { k:v for v,k in enumerate( set(data)) }
>>> enc
{'RESEARCH': 0, 'SALES': 1, 'ACCOUNTING': 2}
>>> dec= { v:k for v,k in enumerate( set(data)) }
>>> dec
{0: 'RESEARCH', 1: 'SALES', 2: 'ACCOUNTING'}
>>> enc['SALES']
1
>>> dec[0]
'RESEARCH'
```



Scikit-Learn 기반 Encoding, Decoding

```
>>> from sklearn import preprocessing
>>> le = preprocessing.LabelEncoder()
>>> data=["ACCOUNTING", "RESEARCH", "SALES", "RESEARCH", "RESEARCH"]
>>> le.fit(data)
LabelEncoder()
>>> le.classes_
array(['ACCOUNTING', 'RESEARCH', 'SALES'], dtype='<U10')
>>> le.transform(data)
array([0, 1, 2, 1, 1])
>>> le.inverse_transform([0, 0, 1, 2])
array(['ACCOUNTING', 'ACCOUNTING', 'RESEARCH', 'SALES'], dtype='<U10')
```



딥러닝 모듈 실습

Keras(Tensorflow) 사용

딥러닝을 위한 Keras와 TensorFlow 환경 준비
옵티마이저 이해
MLP 실습
AutoEncoder 실습

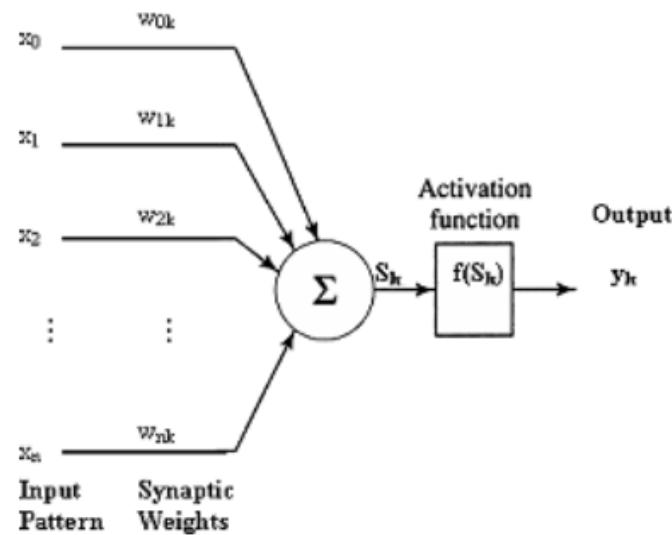
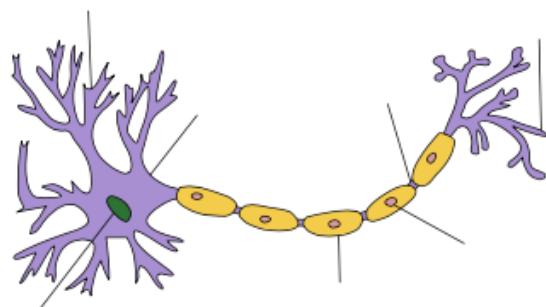


딥러닝을 위한 Keras와 TensorFlow 환경 준비



Deep Learning이란?

- 일반적 정의 : 여러 비선형 변환기법의 조합을 통해 높은 수준의 추상화(abstractions, 다양한 데이터나 복잡한 자료들 속에서 핵심적인 내용 또는 기능을 요약하는 작업)를 시도하는 **기계학습**(machine learning) 알고리즘의 집합
- 빠른 의미 : 사람의 사고방식을 컴퓨터에게 가르치는 기계학습의 한 분야
- 더 빠른 의미 : GPU컴퓨팅, 알파고?



Deep Learning 성공 원인

- 알고리듬 개선

- pre-training: layer 별로 pre-training을 시켜서 initialization을 잘해서 과적응을 방지
- drop-out: training 시 drop-out시킴으로써 over-fitting을 감소 시킴
- rectified unit function: 기존의 sigmoid 함수를 rectified unit function로 대체함으로써, vanishing gradient effect를 줄임

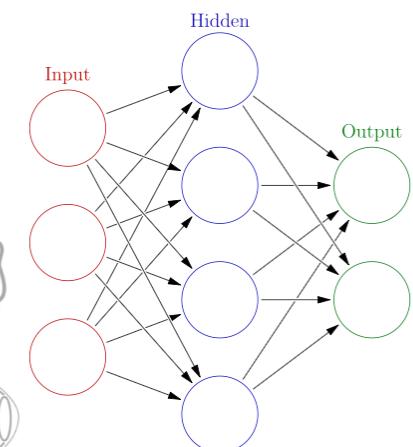
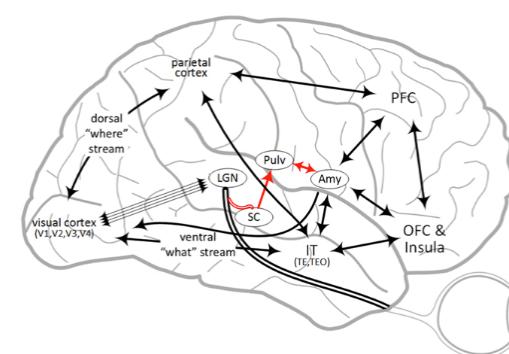
- 빅 데이터 (Big Data)

SNS 사용자들에 의해 생산되는 다양한 자료와 태그 정보

- 하드웨어 성능 개선

병렬 시스템: GPU

분산 시스템: 클라우드 컴퓨팅, Spark, Hadoop

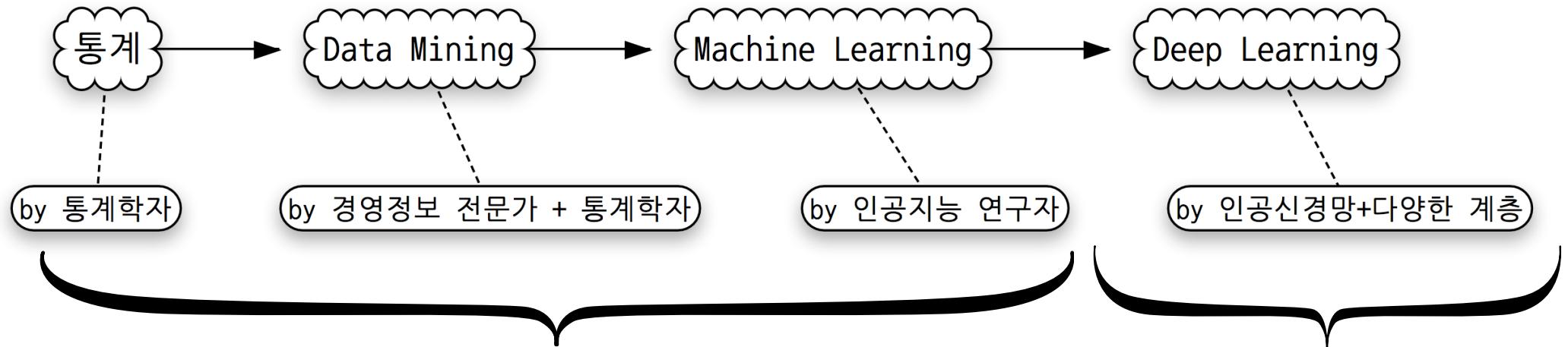




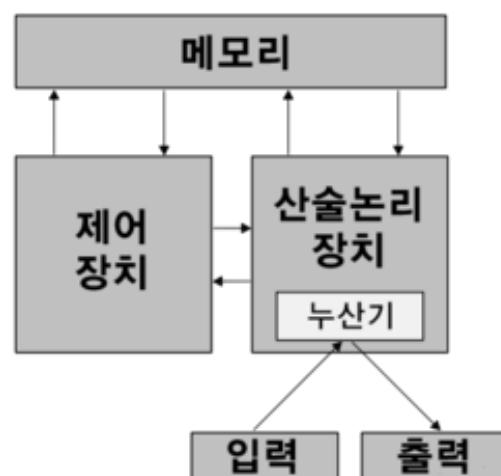
Deep Learning 알고리즘

- 심층 신경망(Deep Neural Network, DNN)
 - ÷ 합성곱 신경망(Convolutional Neural Network, CNN)
- 순환 신경망(Recurrent Neural Network, RNN)
- 제한 볼츠만 머신 (Restricted Boltzmann Machine, RBM)
 - 심층 신뢰 신경망 (Deep Belief Network, DBN)
 - AutoEncoder
- 심층 Q-네트워크(Deep Q-Networks or Reinforcement Learning)

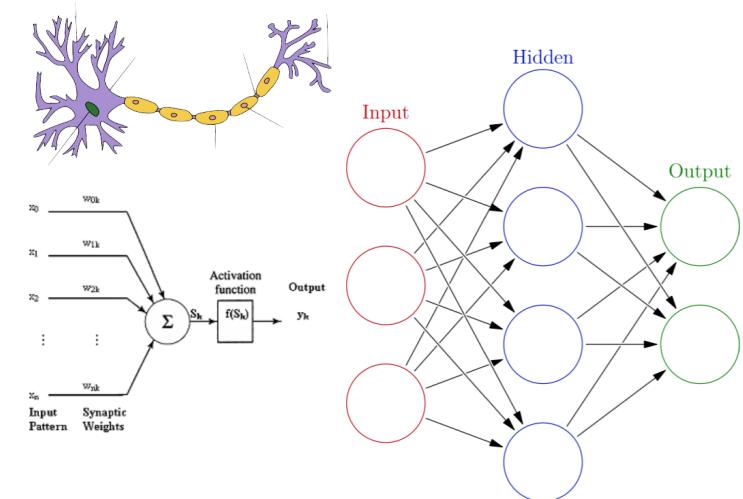
What is DeepLearning?



폰 노이만 구조

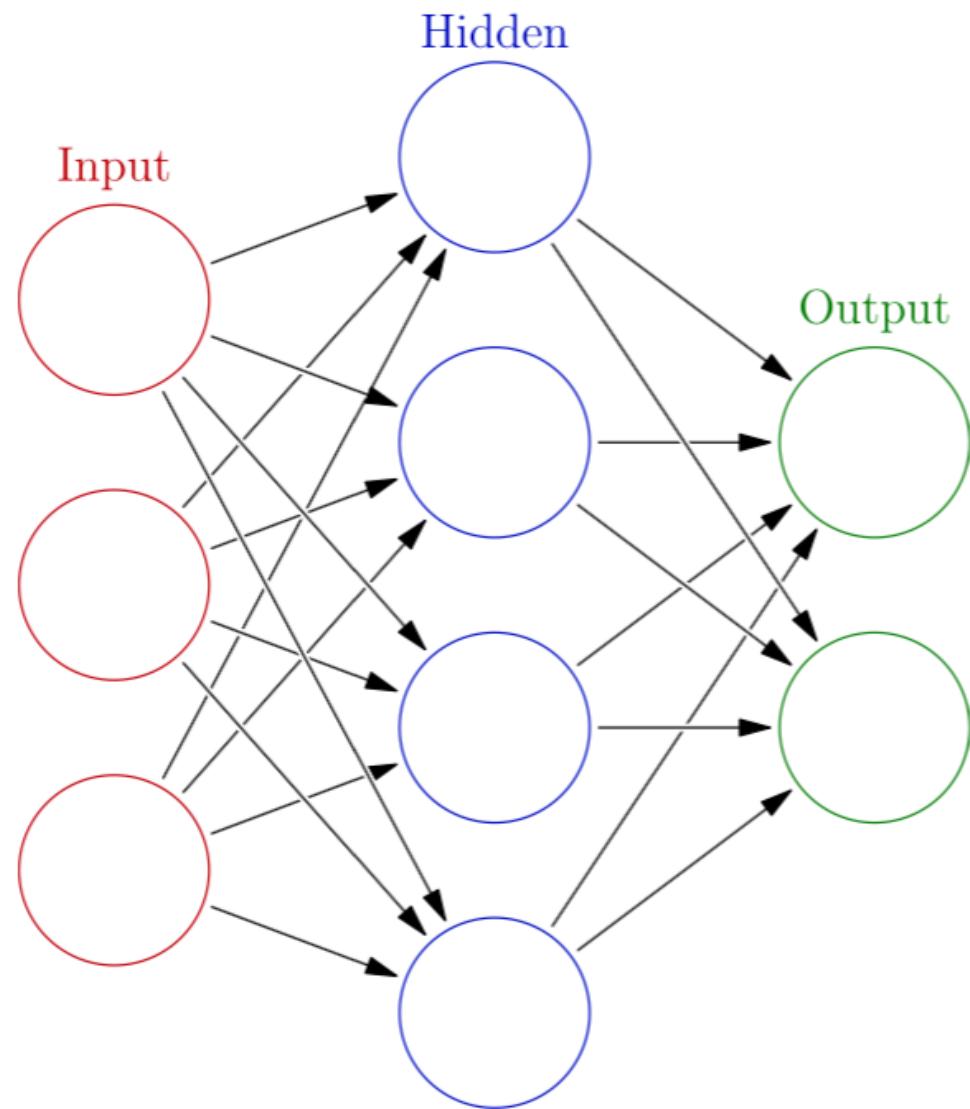
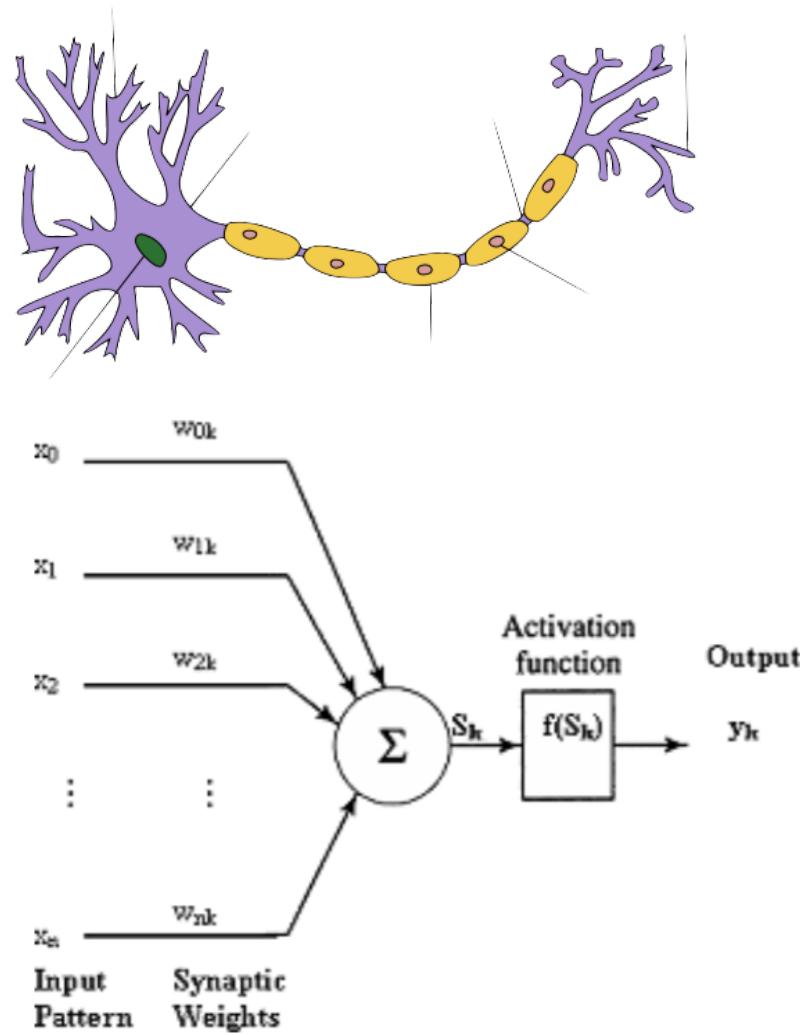


인공신경망 구조



What is DeepLearning?

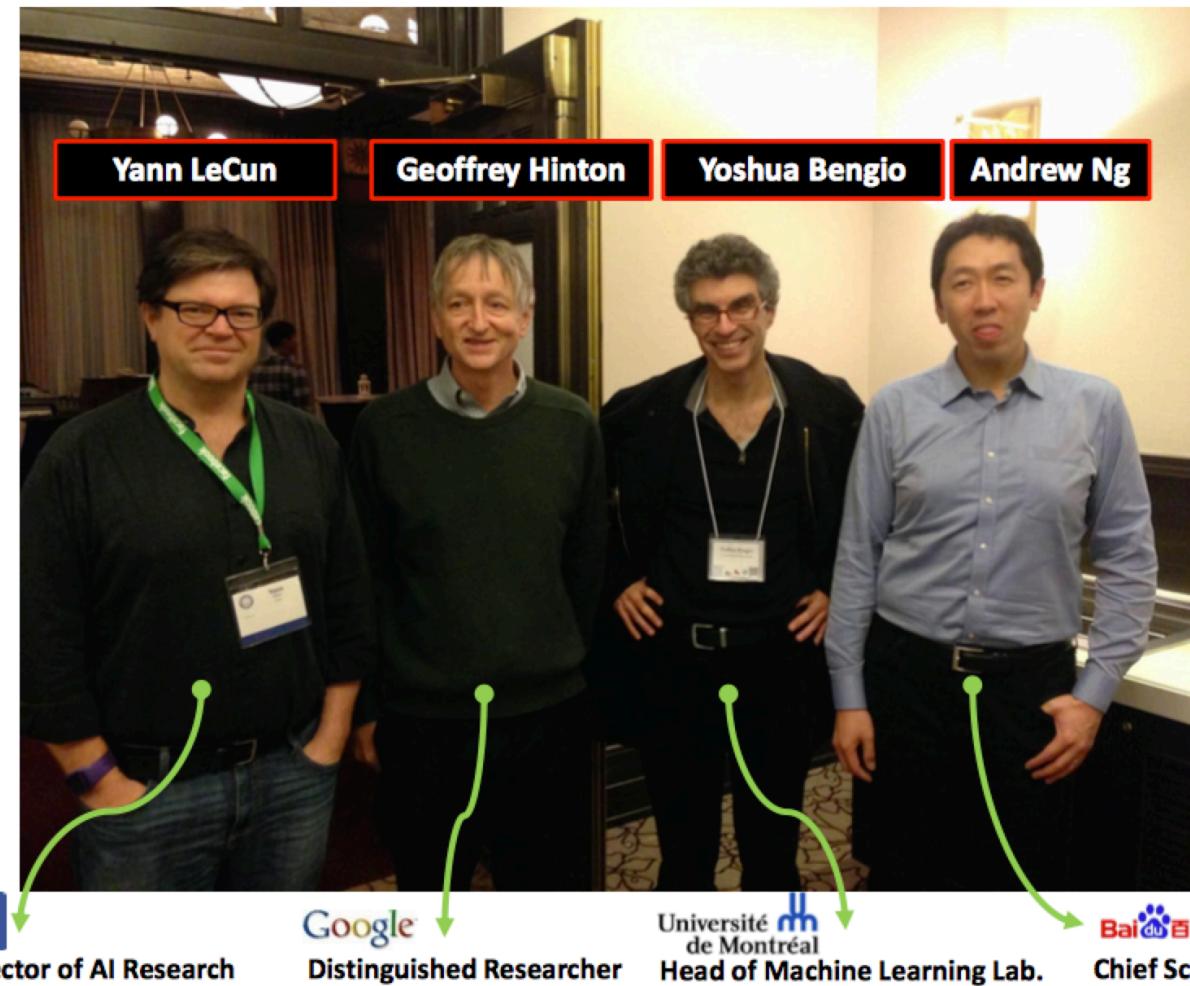
인공신경망 구조





AI in the past

- Many Failures(Neural Network).
- A Few Successes

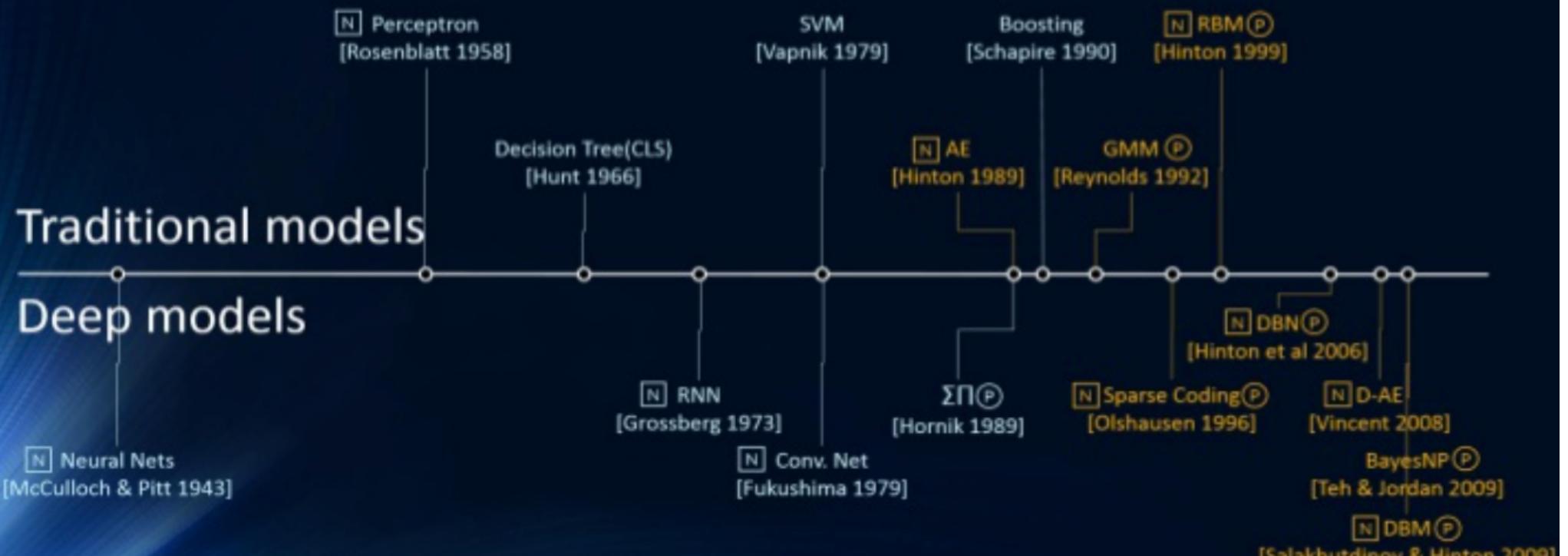




DeepLearning evolution

Deep Learning evolution

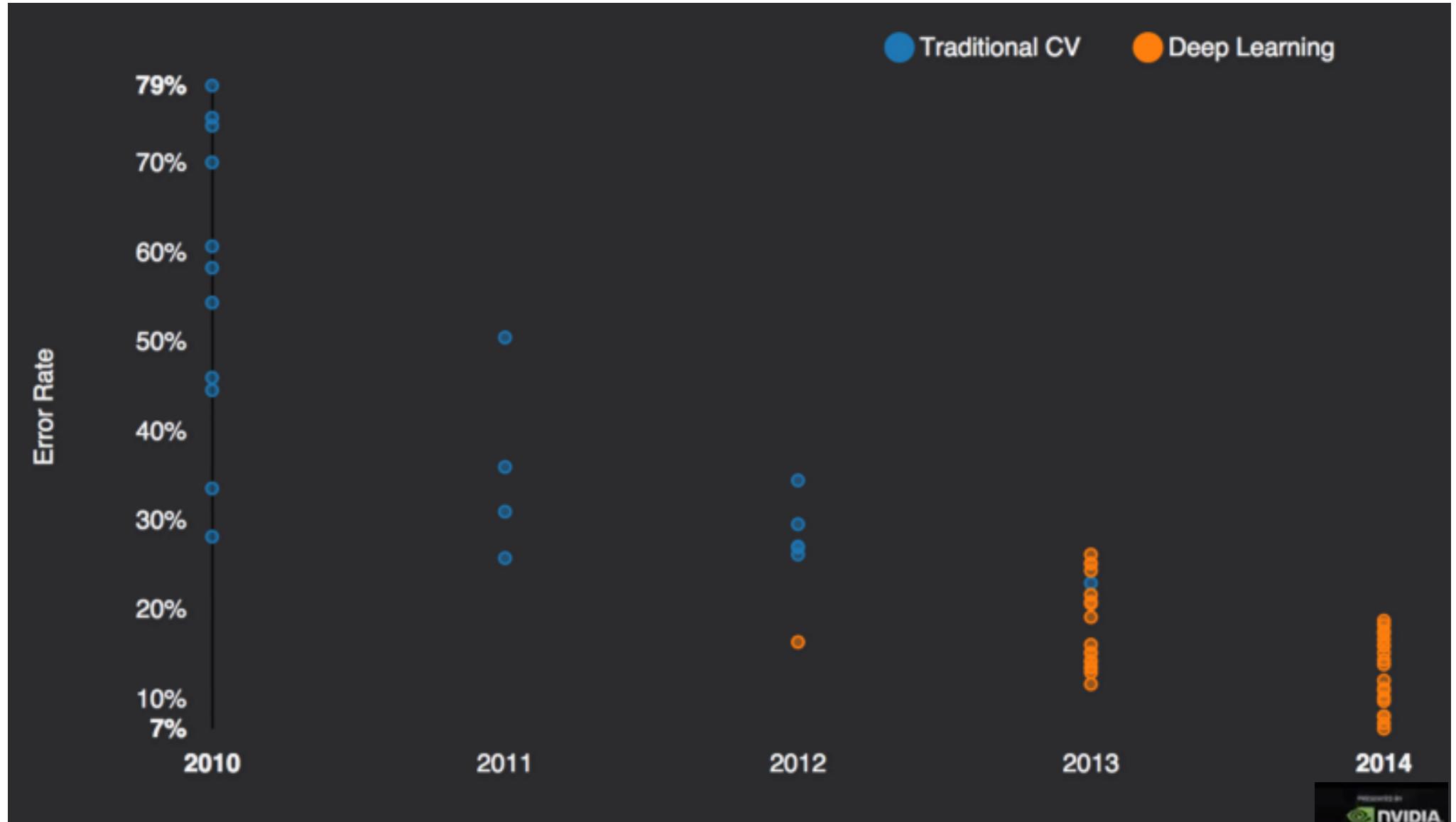
- [N] Neural Network
- (P) Probabilistic Model
- (●) Supervised learning
- (○) Unsupervised learning



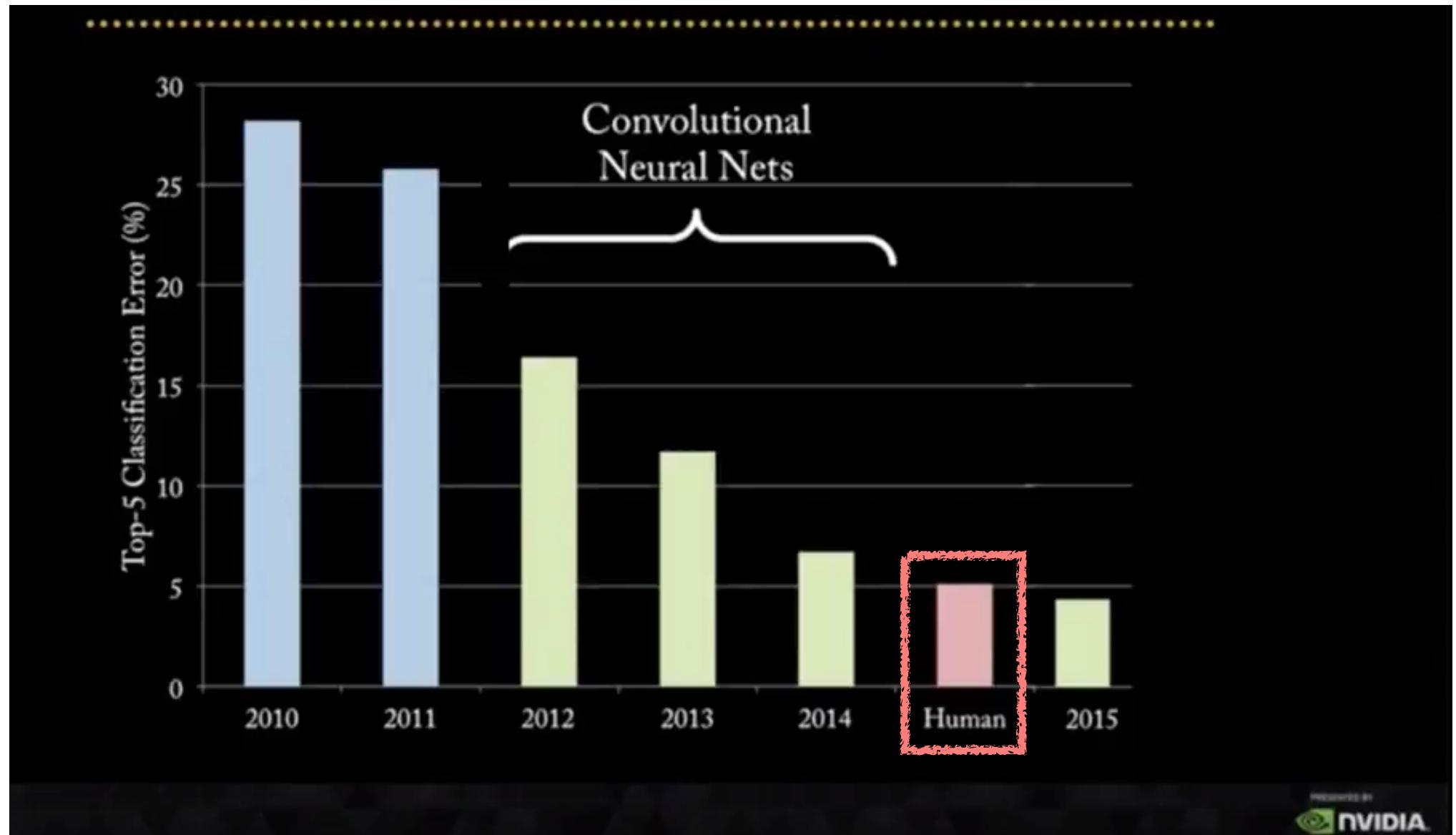
Algorithms authors and dates often unclear. Oldest citations were assumed.
Classifications based on Yann LeCun's Deep Learning class at NYU – spring 2014



ImageNet : Traditional CV .vs. Deep Learning



ImageNet Classification(2010~2015)



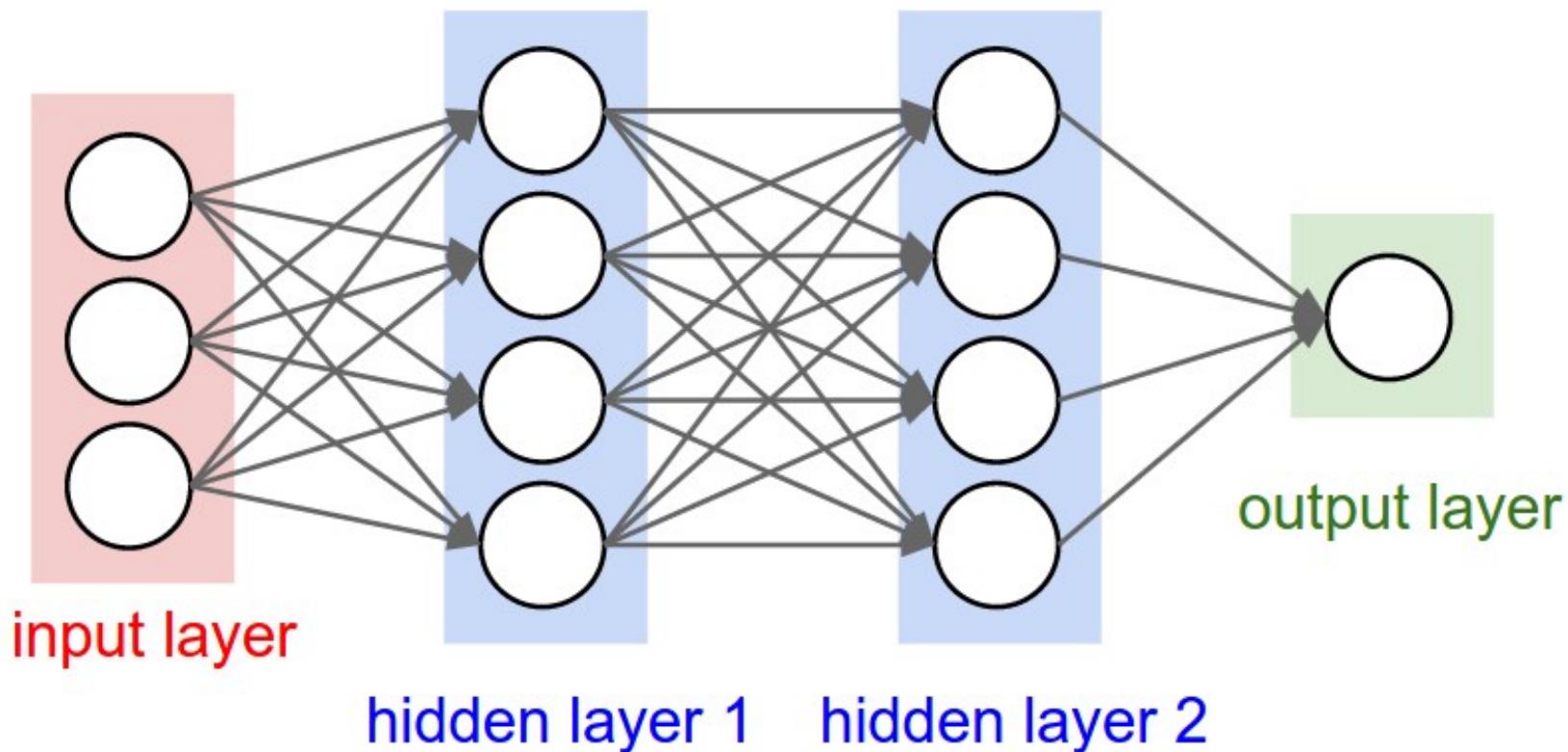


딥러닝 관련 용어

- Perceptron : 뇌세포를 구현
- Weight : 기억 자체
- Learning Rate : 학습률
- Activation Function : ex) Sigmoid, ReLu
- Loss Function : 실제값 - 결과값 중에 결과값
- Layer : MLP에서 Perceptron을 쌓은 단
- 학습이 일어난다 : Error Function을 미분해서 Learning Rate 곱해서 에러가 최소화되는 Weight을 찾음.
- Back Propagation : 학습(여러층)
- forward Propagation : 적용(여러층)
- Epoch : 학습의 한 단계
- Training/Test : 학습/검증
- Overfitting : 과도학습
- Hyper parameter : 분석을 위한 옵션들 ex) Learning Rate, Activation Function 등
- Test Error : (Overfitting이 많이 된 상태)
- Regularization : Test Error를 줄이기 위해 결과를 일반화
- Dropout : 임의로 perceptron 일부를 작동 안시키시고 학습



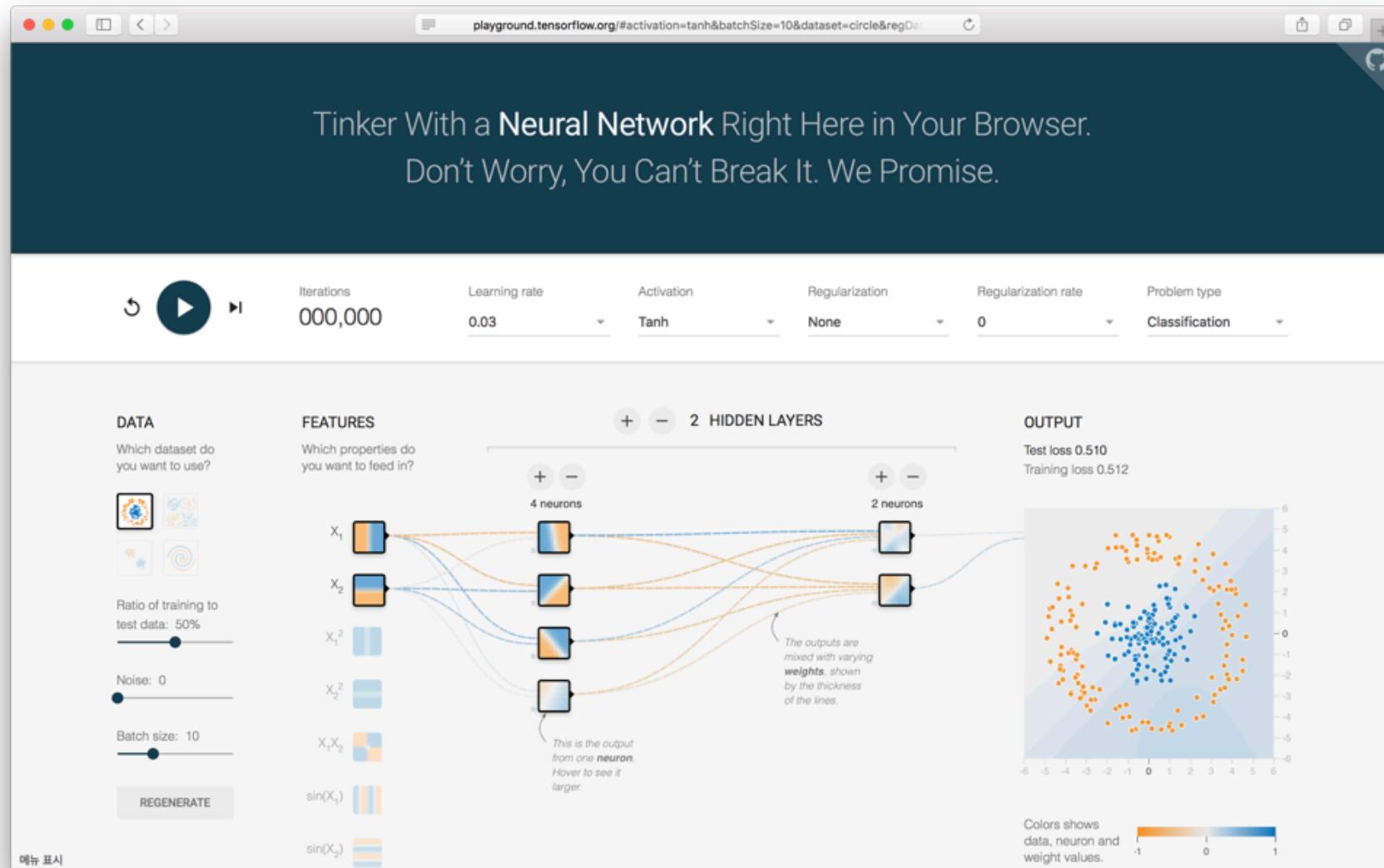
다층 신경망(=다층퍼셉트론 신경망, =MLP)



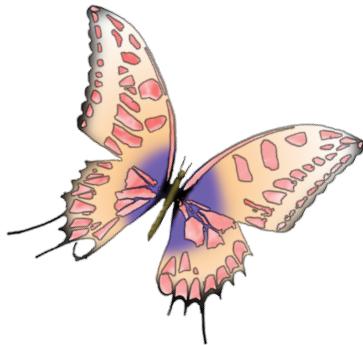


MLP: playground

- <http://playground.tensorflow.org>



실습 : 시뮬레이터 통한 딥러닝 개념과 용어 익히기



- 딥러닝 용어들에 해당하는 시뮬레이터의 옵션들을 바꿔서 Training하여 딥러닝 용어들을 익혀 봅시다.





The engines of modern AI

- Tensorflow - Google
 - Keras
- Caffe - UC Berkeley
- Torch - NYU, Facebook
 - PyTorch
- CNTK - Microsoft
- Watson - IBM
- Theano - Universite de Motreal
- and so on



TensorFlow 소개

- TensorFlow is not a tool for Deep learning
- but is a tool for numerical computation using data flow graphs
- By Google





텐서플로우 특징

- 데이터 플로우 그래프를 통한 풍부한 표현력
- 코드 수정 없이 CPU/GPU 모드로 동작
- 아이디어 테스트에서 서비스 단계까지 이용 가능
- 계산 구조와 목표 함수만 정의하면 자동으로 미분 계산을 처리
- Python/C++를 지원하며, SWIG를 통해 다양한 언어 지원 가능

"구글이 텐서플로우를 오픈소스로 한 것은, 기계 학습이 앞으로 제품과 기술을 혁신하는데 가장 필수적인 요소라고 믿기 때문입니다." - *Google Brain Team*



Hello World

```
import tensorflow as tf  
# Create a Constant op  
hello = tf.constant('Hello, TensorFlow!')  
# Start tf session  
sess = tf.Session()  
# Run the op  
print(sess.run(hello))  
sess.close()
```

Tensorflow Basic Example

```
import tensorflow as tf

# 변수를 0으로 초기화
state = tf.Variable(0, name="counter")

# state에 1을 더할 오퍼레이션 생성
one = tf.constant(1)
new_value = tf.add(state, one)
update = tf.assign(state, new_value)

# 그래프는 처음에 변수를 초기화해야 합니다. 아래 함수를 통해 init 오퍼레이션을 만듭니다.
init_op = tf.initialize_all_variables()

# 그래프를 띄우고 오퍼레이션들을 실행
with tf.Session() as sess:
    # 초기화 오퍼레이션 실행
    sess.run(init_op)
    # state의 초기 값을 출력
    print(sess.run(state))
    # state를 갱신하는 오퍼레이션을 실행하고, state를 출력
    for _ in range(3):
        sess.run(update)
        print(sess.run(state))
```

결과

0
1
2
3

Tensorflow 라이브러리 탐색

python

```
>>> import tensorflow as tf
```

```
>>> dir(tf)
```

```
['AggregationMethod', 'Assert', 'AttrValue', 'ConfigProto', 'DType', 'DeviceSpec', 'Dimension', 'Event', 'FIFOQueue', 'FixedLenFeature', 'FixedLenSequenceFeature', 'FixedLengthRecordReader', 'GPUOptions', 'GRAPH_DEF_VERSION', 'GRAPH_DEF_VERSION_MIN_CONSUMER', 'GRAPH_DEF_VERSION_MIN_PRODUCER', 'Graph', 'GraphDef', 'GraphKeys', 'GraphOptions', 'HistogramProto', 'IdentityReader', 'IndexedSlices', 'InteractiveSession', 'LogMessage', 'NameAttrList', 'NoGradient', 'NodeDef', 'OpError', 'Operation', 'OptimizerOptions', 'PaddingFIFOQueue', 'Print', 'QUANTIZED_DTYPES', 'QueueBase', 'RandomShuffleQueue', 'ReaderBase', 'RegisterGradient', 'RegisterShape', 'RunMetadata', 'RunOptions', 'Session', 'SessionLog', 'SparseTensor', 'SparseTensorValue', 'Summary', 'TFRecordReader', 'Tensor', 'TensorArray', 'TensorShape', 'TextLineReader', 'VarLenFeature', 'Variable', 'VariableScope', 'WholeFileReader', '__builtins__', '__doc__', '__file__', '__name__', '__package__', '__path__', '__version__', 'abs', 'absolute_import', 'accumulate_n', 'acos', 'add', 'add_check_numerics_ops', 'add_n', 'add_to_collection', 'all_variables', 'app', 'arg_max', 'arg_min', 'argmax', 'argmin', 'as_dtype', 'as_string', 'asin', 'assert_equal', 'assert_integer', 'assert_less', 'assert_less_equal', 'assert_negative', 'assert_non_negative', 'assert_non_positive', 'assert_positive', 'assert_proper_iterable', 'assert_rank', 'assert_rank_at_least', 'assert_type', 'assert_variables_initialized', 'assign', 'assign_add', 'assign_sub', 'atan', 'audio_summary', 'batch_cholesky', 'batch_matrix_solve', 'batch_fft', 'batch_fft2d', 'batch_fft3d', 'batch_ifft2d', 'batch_ifft3d', 'batch_matmul', 'batch_matrix_band_part', 'batch_matrix_determinant', 'batch_matrix_diag', 'batch_matrix_diag_part', 'batch_matrix_inverse', 'batch_matrix_set_diag', 'batch_matrix_solve', 'batch_matrix_solve_ls', 'batch_matrix_transpose', 'batch_matrix_triangular_solve', 'batch_self_adjoint_eig', 'batch_self_adjoint_eigvals', 'batch_svd', 'batch_to_space', 'bfloating16', 'bfloating16_ref', 'bitcast', 'bool', 'bool_ref', 'boolean_mask', 'bytes', 'case', 'cast', 'ceil', 'check_numerics', 'cholesky', 'cholesky_solve', 'clip_by_average_norm', 'clip_by_global_norm', 'clip_by_norm', 'clip_by_value', 'compat', 'complex', 'complex128', 'complex128_ref', 'complex64', 'complex64_ref', 'complex_abs', 'concat', 'conj', 'constant', 'constant_initializer', 'container', 'contrib', 'control_dependencies', 'convert_to_tensor', 'convert_to_tensor_or_indexed_slices', 'core', 'cos', 'count_up_to', 'create_partitioned_variables', 'cross', 'cumprod', 'cumsum', 'decode_csv', 'decode_json_example', 'decode_raw', 'delete_session_tensor', 'depth_to_space', 'deserialize_many_sparse', 'device', 'diag', 'diag_part', 'digamma', 'div', 'division', 'double', 'double_ref', 'dynamic_partition', 'dynamic_stitch', 'edit_distance', 'equal', 'erf', 'erfc', 'errors', 'exp', 'expand_dims', 'extract_image_patches', 'fft', 'fft2d', 'fft3d', 'fill', 'flags', 'float16', 'float16_ref', 'float32', 'float32_ref', 'float64', 'float64_ref', 'floor', 'floordiv', 'foldl', 'foldr', 'gather', 'gather_nd', 'get_collection', 'get_collection_ref', 'get_default_graph', 'get_default_session', 'get_seed', 'get_session_handle', 'get_session_tensor', 'get_variable', 'get_variable_scope', 'gfile', 'global_norm', 'gradients', 'greater', 'greater_equal', 'group', 'half', 'half_ref', 'histogram_fixed_width', 'histogram_summary', 'identity', 'ifft', 'ifft2d', 'ifft3d', 'igamma', 'igammac', 'imag', 'image', 'image_summary', 'import_graph_def', 'initialize_all_tables', 'initialize_all_variables', 'initialize_local_variables', 'initialize_variables', 'int16', 'int16_ref', 'int32', 'int32_ref', 'int64', 'int64_ref', 'int8', 'int8_ref', 'inv', 'invert_permutation', 'is_finite', 'is_inf', 'is_nan', 'is_non_decreasing', 'is_numeric_tensor', 'is_strictly_increasing', 'is_variable_initialized', 'lbeta', 'less', 'less_equal', 'lgamma', 'lin_space', 'linspace', 'list_diff', 'listdiff', 'load_file_system_library', 'load_op_library', 'local_variables', 'log', 'logging', 'logical_and', 'logical_not', 'logical_or', 'logical_xor', 'make_template', 'map_fn', 'matching_files', 'matmul', 'matrix_determinant', 'matrix_inverse', 'matrix_solve', 'matrix_solve_ls', 'matrix_triangular_solve', 'maximum', 'merge_all_summaries', 'merge_summary', 'meshgrid', 'min_max_variable_partitioner', 'minimum', 'mod', 'moving_average_variables', 'mul', 'multinomial', 'name_scope', 'neg', 'newaxis', 'nn', 'no_op', 'no_regularizer', 'not_equal', 'one_hot', 'ones', 'ones_initializer', 'ones_like', 'op_scope', 'pack', 'pad', 'parse_example', 'parse_single_example', 'parse_single_sequence_example', 'placeholder', 'placeholder_with_default', 'polygamma', 'pow', 'print_function', 'py_func', 'python', 'python_io', 'qint16', 'qint16_ref', 'qint32', 'qint32_ref', 'qint8', 'qint8_ref', 'quint16', 'quint16_ref', 'quint8', 'quint8_ref', 'random_crop', 'random_gamma', 'random_normal', 'random_normal_initializer', 'random_shuffle', 'random_uniform', 'random_uniform_initializer', 'range', 'rank', 'read_file', 'real', 'reduce_all', 'reduce_any', 'reduce_join', 'reduce_max', 'reduce_mean', 'reduce_min', 'reduce_prod', 'reduce_sum', 'register_tensor_conversion_function', 'report_uninitialized_variables', 'reset_default_graph', 'reshape', 'resource_loader', 'reverse', 'reverse_sequence', 'round', 'rsqrt', 'saturate_cast', 'scalar_mul', 'scalar_summary', 'scan', 'scatter_add', 'scatter_sub', 'scatter_update', 'segment_max', 'segment_mean', 'segment_min', 'segment_prod', 'segment_sum', 'select', 'self_adjoint_eig', 'self_adjoint_eigvals', 'serialize_many_sparse', 'serialize_sparse', 'set_random_seed', 'shape', 'shape_n', 'sigmoid', 'sign', 'sin', 'size', 'slice', 'space_to_batch', 'space_to_depth', 'sparse_add', 'sparse_concat', 'sparse_fill_empty_rows', 'sparse_mask', 'sparse_matmul', 'sparse_maximum', 'sparse_merge', 'sparse_minimum', 'sparse_placeholder', 'sparse_reduce_sum', 'sparse_reorder', 'sparse_reset_shape', 'sparse_reshape', 'sparse_retain', 'sparse_segment_mean', 'sparse_segment_mean_grad', 'sparse_segment_sqrt_n', 'sparse_segment_sqrt_n_grad', 'sparse_segment_sum', 'sparse_softmax', 'sparse_split', 'sparse_tensor_dense_matmul', 'sparse_tensor_to_dense', 'sparse_to_dense', 'sparse_to_indicator', 'split', 'sqrt', 'square', 'squared_difference', 'squeeze', 'stop_gradient', 'strided_slice', 'string', 'string_join', 'string_ref', 'string_to_hash_bucket', 'string_to_hash_bucket_fast', 'string_to_hash_bucket_strong', 'string_to_number', 'sub', 'summary', 'svd', 'sysconfig', 'tan', 'tanh', 'test', 'tile', 'to_bfloat16', 'to_double', 'to_float', 'to_int32', 'to_int64', 'trace', 'train', 'trainable_variables', 'transpose', 'truediv', 'truncated_normal', 'truncated_normal_initializer', 'tuple', 'uint16', 'uint16_ref', 'uint8', 'uint8_ref', 'uniform_unit_scaling_initializer', 'unique', 'unique_with_counts', 'unpack', 'unsorted_segment_sum', 'user_ops', 'variable_axis_size_partitioner', 'variable_op_scope', 'variable_scope', 'verify_tensor_all_finite', 'where', 'while_loop', 'zeros', 'zeros_initializer', 'zeros_like', 'zeta']
```



Tensorflow : 연산 category

연산 카테고리	연산 예
Maths	Add, Sub, Mul, Div, Exp, Log, Greater, Less ,Equal
Array	Concat, Slice, Split, Constant, Rank, Shape, Shuffle
Matrix	MatMul, MatrixInverse, MatrixDeterminant
Neuronal	Network SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool
Checkpointing	Save, Restore
Queues	and syncronizations Enqueue, Dequeue, MutexAcquire, MutexRelease
Flow	control Merge, Switch, Enter, Leave, NextIteration

Tensorflow : General



함수,	설명
tf.add,	덧셈
tf.sub,	뺄셈
tf.mul,	곱셈
tf.div,	나눗셈의 몫
tf.mod,	나눗셈의 나머지
tf.abs,	절대값을 리턴합니다.
tf.neg,	음수를 리턴합니다.
tf.sign,	부호를 리턴합니다.(역주: 음수는 -1, 양수는 1, 0 일땐 0 을 리턴합니다)
tf.inv,	역수를 리턴합니다.(역주: 3의 역수는 1/3 입니다)
tf.square,	제곱을 계산합니다.

함수,	설명
tf.round,	반올림 값을 리턴합니다.
tf.sqrt,	제곱근을 계산합니다.
tf.pow,	거듭제곱 값을 계산합니다.
tf.exp,	지수 값을 계산합니다.
tf.log,	로그 값을 계산합니다.
tf.maximum	최대값을 리턴합니다.
tf.minimum	최소값을 리턴합니다.
tf.cos,	코사인 함수 값을 계산합니다.
tf.sin,	사인 함수 값을 계산합니다.



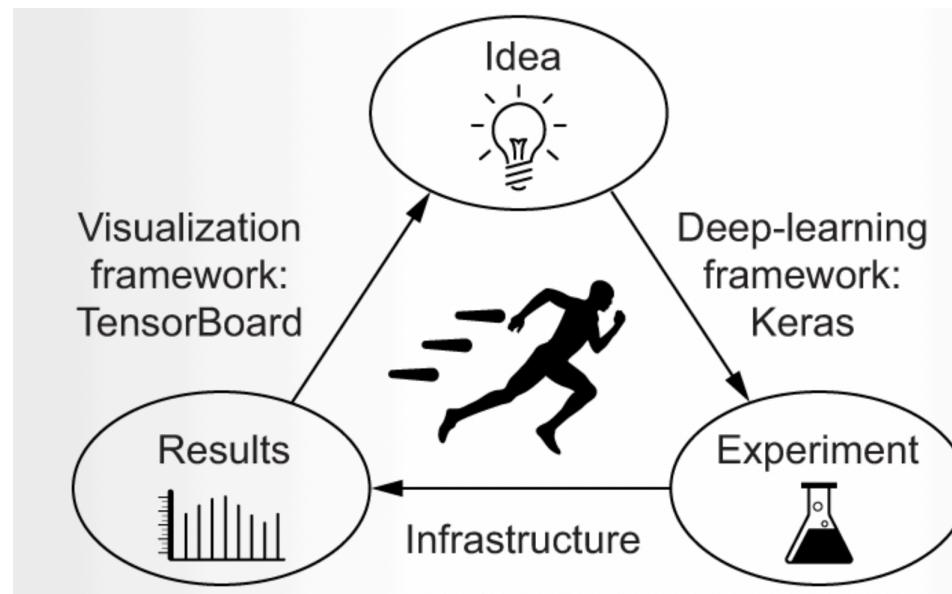
Tensorflow : Matrix

함수	설명
<code>tf.diag</code>	대각행렬을 리턴합니다.
<code>tf.transpose</code>	전치행렬을 리턴합니다.
<code>tf.matmul</code>	두 텐서를 행렬곱셈하여 결과 텐서를 리턴합니다.
<code>tf.matrix_determinant</code>	정방행렬의 행렬식 값을 리턴합니다.
<code>tf.matrix_inverse</code>	정방행렬의 역행렬을 리턴합니다.



Keras

- 파이썬으로 구현된 쉽고 간결한 딥러닝 라이브러리
- 딥러닝 비전문가라도 각자 분야에서 손쉽게 딥러닝 모델을 개발하고 활용할 수 있도록 케라스는 직관적인 API를 제공
- Tensorflow에 기본 내장 되어 있음
- 내부적으로는 텐서플로우(TensorFlow), 티아노(Theano), CNTK 등의 딥러닝 전용 엔진이 구동되지만 케라스 사용자는 복잡한 내부 엔진을 알 필요는 없음
- Home Page : <https://keras.io>





Keras의 주요 특징

- 모듈화 (Modularity)
 - 케라스에서 제공하는 모듈은 독립적이고 설정 가능
 - 가능한 최소한의 제약사항으로 서로 연결
 - 모델은 시퀀스 또는 그래프로 이러한 모듈들을 구성한 것
- 최소주의 (Minimalism)
 - 각 모듈은 짧고 간결
 - 모든 코드는 한 번 훑어보는 것으로도 이해 가능
- 쉬운 확장성
 - 새로운 클래스나 함수로 모듈을 아주 쉽게 추가 가능
 - 따라서 고급 연구에 필요한 다양한 표현 가능
- 파이썬 기반
 - Caffe 처럼 별도의 모델 설정 파일이 필요없으며 파이썬 코드로 모델들이 정의됨

Keras vs Tensorflow



Keras

```

1. import tensorflow as tf
2. from keras.models import Sequential
3. import pandas as pd
4. from keras.layers import Dense
5.
6. cols = ['Usercountry', 'Nr. reviews', 'Nr. hotel reviews', 'Helpful votes',
7.          'Score', 'Period of stay', 'Traveler type', 'Pool', 'Gym', 'Tennis court',
8.          'Spa', 'Casino', 'Free internet', 'Hotel name', 'Hotel stars', 'Nr. rooms',
9.          'User continent', 'Member years', 'Review month', 'Review weekday']
10. data = pd.read_csv('/home/ubuntu/Downloads/tripAdvisorFL.csv', delimiter=',', names=cols)
11. import numpy as np
12. from keras.utils import np_utils
13. from sklearn.model_selection import train_test_split
14. labels = data['Score']
15. features = data.drop(['Score'], axis=1)
16. X=features
17. y=np_utils.to_categorical(labels)
18. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
19. from keras.models import Sequential
20. from keras.layers import Dense
21. model = Sequential()
22. model.add(Dense(8, input_dim=19, activation='relu'))
23. model.add(Dense(6, activation='softmax'))
24. model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
25. model.fit(X_train, y_train, epochs=4, batch_size=1, verbose=1)

```

TensorFlow

```

1. import tensorflow as tf
2. import numpy as np
3. feature_names = ['Usercountry', 'Nrreviews', 'Nrhotelreviews', 'Helpfulvotes', 'Periodofstay',
   'Travelertype', 'Pool', 'Gym', 'Tenniscourt', 'Spa', 'Casino', 'Freeinternet', 'Hotelname', 'Hotelstars', 'Nrrooms', 'Usercontinent', 'Memberyears',
   'Reviewmonth', 'Reviewweekday']
4. FIELD_DEFAULTS = [[0], [0], [0], [0],
   [0], [0], [0], [0],
   [0], [0], [0], [0],
   [0], [0], [0], [0],
   [0], [0], [0], [0]]
5. def parse_line(line):
6.     parsed_line = tf.decode_csv(line, FIELD_DEFAULTS)
7.     features = parsed_line
8.     d = dict(zip(feature_names, features))
9.     print("dictionary", d, "label =", label)
10.    return d, label
11. def csv_input_fn(csv_path, batch_size):
12.     dataset = tf.data.TextLineDataset(csv_path)
13.     dataset = dataset.map(parse_line)
14.     dataset = dataset.shuffle(1000).repeat().batch(batch_size)
15.     return dataset
16. Usercountry = tf.feature_column.numeric_column("Usercountry")
17. Nrreviews = tf.feature_column.numeric_column("Nrreviews")
18. Nrhotelreviews = tf.feature_column.numeric_column("Nrhotelreviews")
19. Helpfulvotes = tf.feature_column.numeric_column("Helpfulvotes")
20. Periodofstay = tf.feature_column.numeric_column("Periodofstay")
21. Travelertype = tf.feature_column.numeric_column("Travelertype")
22. Pool = tf.feature_column.numeric_column("Pool")
23. Gym = tf.feature_column.numeric_column("Gym")
24. Tenniscourt = tf.feature_column.numeric_column("Tenniscourt")
25. Spa = tf.feature_column.numeric_column("Spa")
26. Casino = tf.feature_column.numeric_column("Casino")
27. Freeinternet = tf.feature_column.numeric_column("Freeinternet")
28. Hotelname = tf.feature_column.numeric_column("Hotelname")
29. Hotelstars = tf.feature_column.numeric_column("Hotelstars")
30. Nrrooms = tf.feature_column.numeric_column("Nrrooms")
31. Usercontinent = tf.feature_column.numeric_column("Usercontinent")
32. Memberyears = tf.feature_column.numeric_column("Memberyears")
33. Reviewmonth = tf.feature_column.numeric_column("Reviewmonth")
34. Reviewweekday = tf.feature_column.numeric_column("Reviewweekday")
35. feature_columns = [Usercountry, Nrreviews, Nrhotelreviews, Helpfulvotes, Periodofstay,
   Travelertype, Pool, Gym, Tenniscourt, Spa, Casino, Freeinternet, Hotelname, Hotelstars, Nrrooms, Usercontinent, Memberyears, Reviewmonth, Reviewweekday]
36.
37. classifier=tf.estimator.DNNClassifier(
38.     feature_columns=feature_columns,
39.     hidden_units=[10, 10],
40.     n_classes=6,
41.     model_dir='/tmp')
42. batch_size = 100
43. classifier.train(
44.     steps=100,
45.     input_fn=lambda : csv_input_fn("/home/ubuntu/Downloads/tripAdvisorFL.csv", batch_size))

```

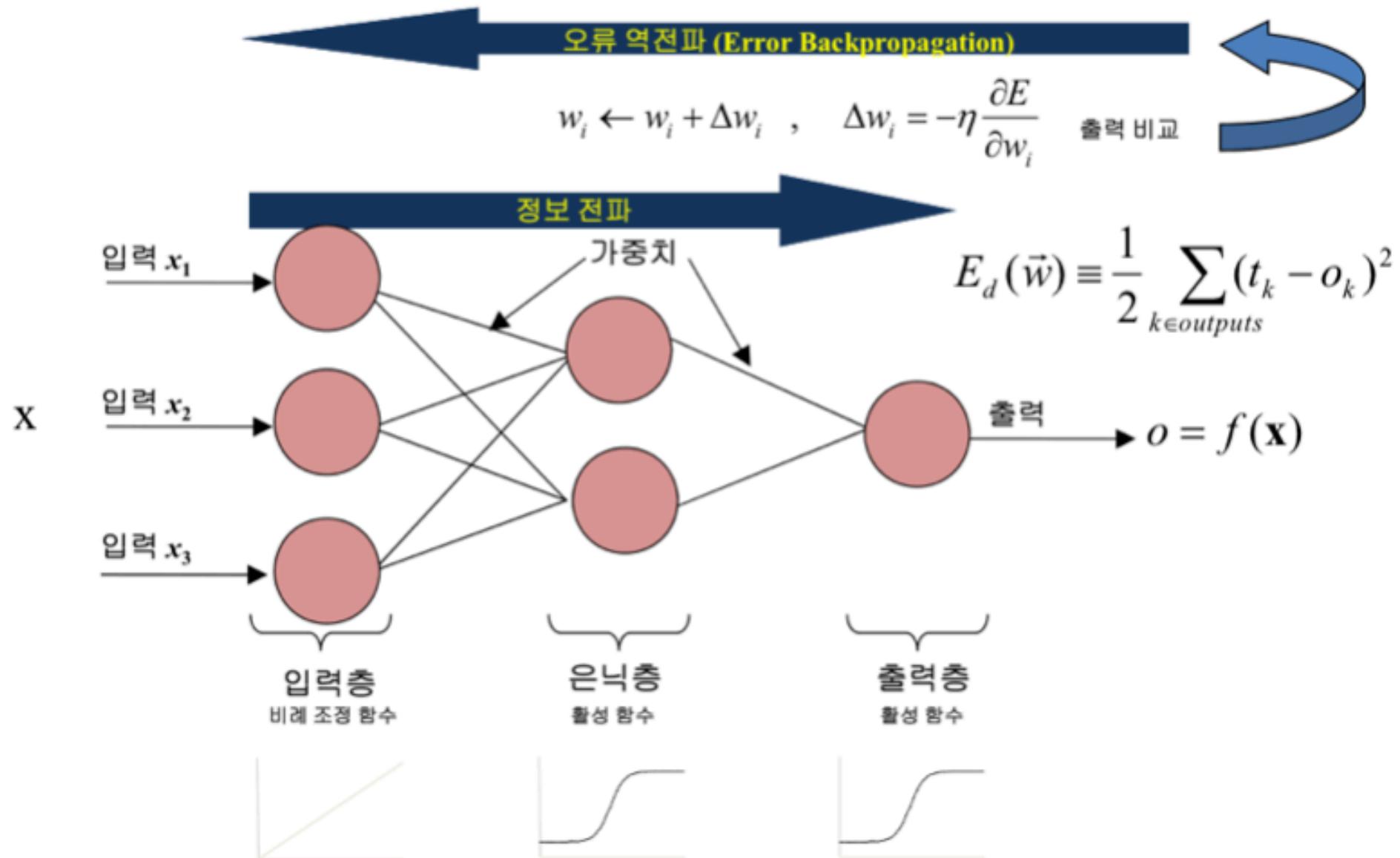
Keras 관련 한글 참고 자료

- Keras 공식 한글 문서 : <https://keras.io/ko/>
- Keras 강좌 : <https://tykimos.github.io/lecture/>
- Tensorflow Blog의 Keras : <https://tensorflow.blog/케라스-딥러닝/3-2-케라스-소개/>
- Keras 이용 간단한 방정식 풀이 : <https://needjarvis.tistory.com/426>



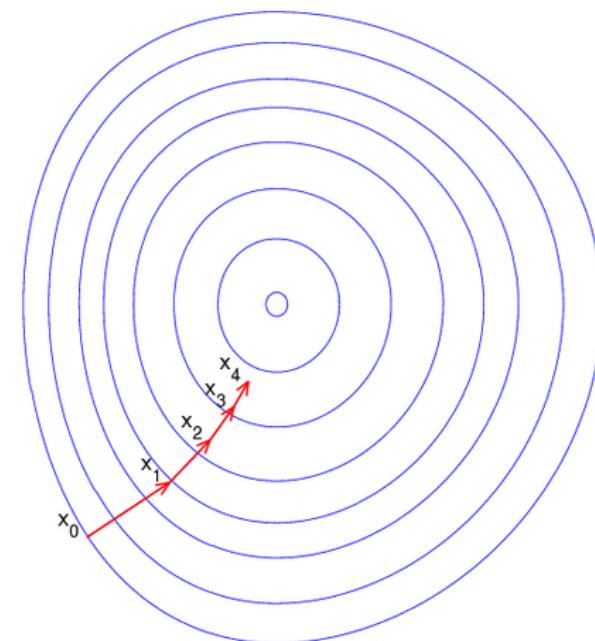
올티마이저 이해

신경망 학습원리

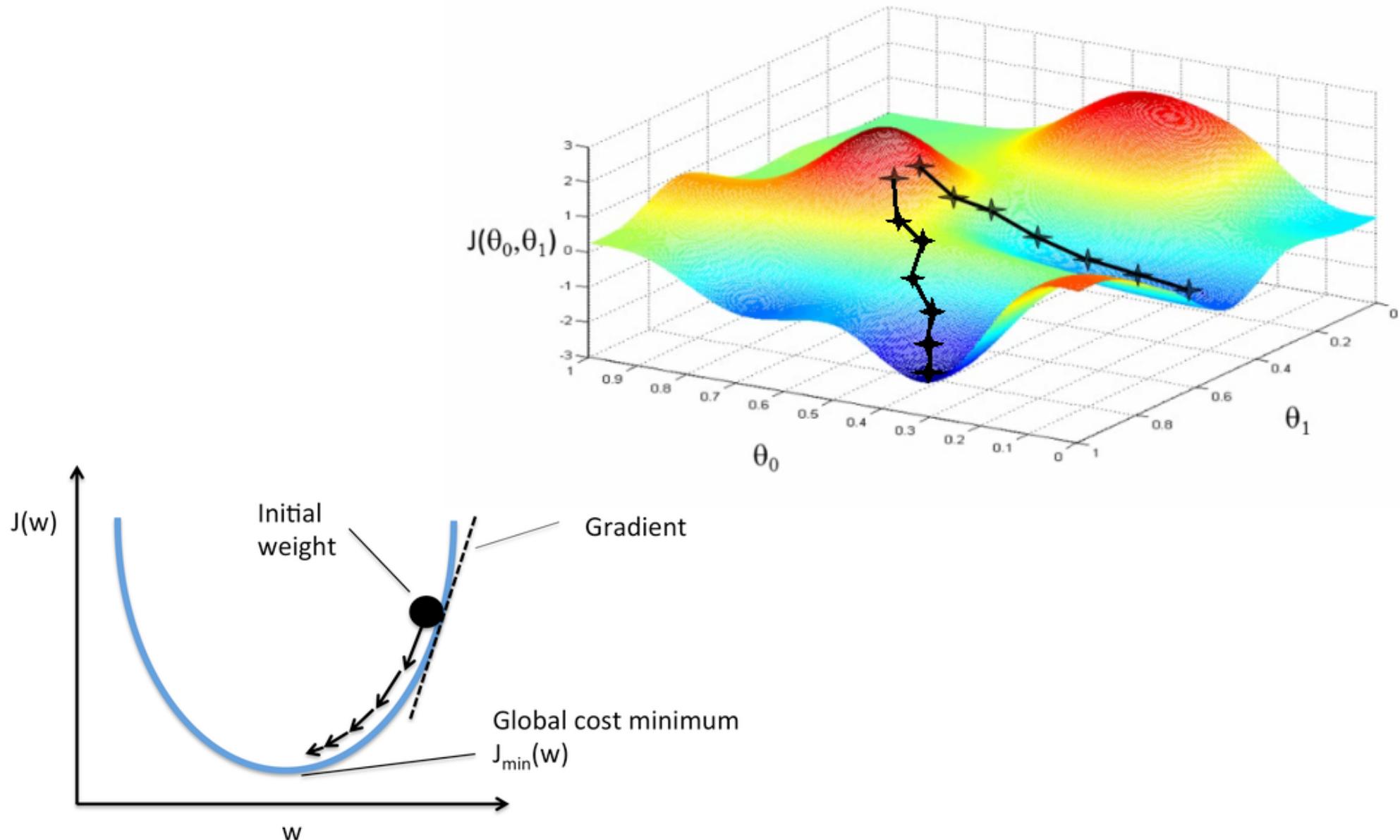


경사 하강법(gradient descent)

- 경사 하강법은 1차 근삿값 발견용 최적화 알고리즘이다. 기본 아이디어는 함수의 기울기(경사)를 구하여 기울기가 낮은 쪽으로 계속 이동시켜서 극값에 이를 때까지 반복시키는 것이다.



경사 하강법(gradient descent)





경사하강법 설명

- 최적화할 함수 $f(x)$ 에 대하여, 먼저 시작점 x_0 를 정한다. 현재 x_i 가 주어졌을 때, 그 다음으로 이동할 점인 x_{i+1} 은 다음과 같이 계산된다.

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i \nabla f(\mathbf{x}_i)$$

- 이때 γ_i 는 이동할 거리를 조절하는 매개변수이다.
- 이 알고리즘의 수렴 여부는 f 의 성질과 γ_i 의 선택에 따라 달라진다. 또한, 이 알고리즘은 지역 최적해로 수렴한다. 따라서 구한 값이 전역적인 최적해라는 것을 보장하지 않으며 시작점 x_0 의 선택에 따라서 달라진다. 이에 따라 다양한 시작점에 대해 여러 번 경사 하강법을 적용하여 그 중 가장 좋은 결과를 선택할 수도 있다.

실습 : Multi-linear Regression 구현



- Multi-linear Regression 구현 예제를 통해
경사 하강법을 이해해 봅시다.





MLP 실습



MLP 학습 원리

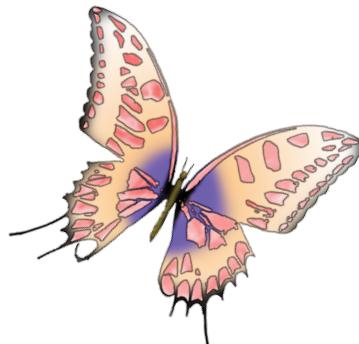
- 대부분의 multilayer perceptron 은 역전파 (Back-propagation) 학습 알고리즘을 사용하여 학습시킬 수 있다.
- 입력층의 각 unit 에 입력 데이터를 제시하면 이 신호는 각 unit에서 변환되어 중간층에 전달되고 최종적으로 출력층으로 나오게 된다. 이 출력값과 원하는 출력값을 비교하여 그 차이를 감소시키는 방향으로 연결 강도를 조정하는 것이다. 그러나 중간층이 있으면 학습은 어려워 진다. 왜냐하면 어떤 연결강도가 오차를 유발 시키는지 알 수 없기 때문이다.
- 단층, 2층, 3층 perceptron 들의 구조와 결정 구역을 나타내었다. 이 그림에서 두 번째 열은 각 네트워크가 형성하는 결정 구역을 나타내고 다음 두 개의 열은 Exclusive-or 와 mesh 구역 문제에 대한 결정 구역을 예시한다. 그리고 맨 오른쪽 열은 일반적인 결정 구역을 예시한다



MLP 학습 원리

Structure	Regions	XOR	Meshed regions
single layer	Half plane bounded by hyperplane		
two layer	Convex open or closed regions		
three layer	Arbitrary (limited by # of nodes)		

실습 : Iris 데이터 셋으로 MLP모델 학습



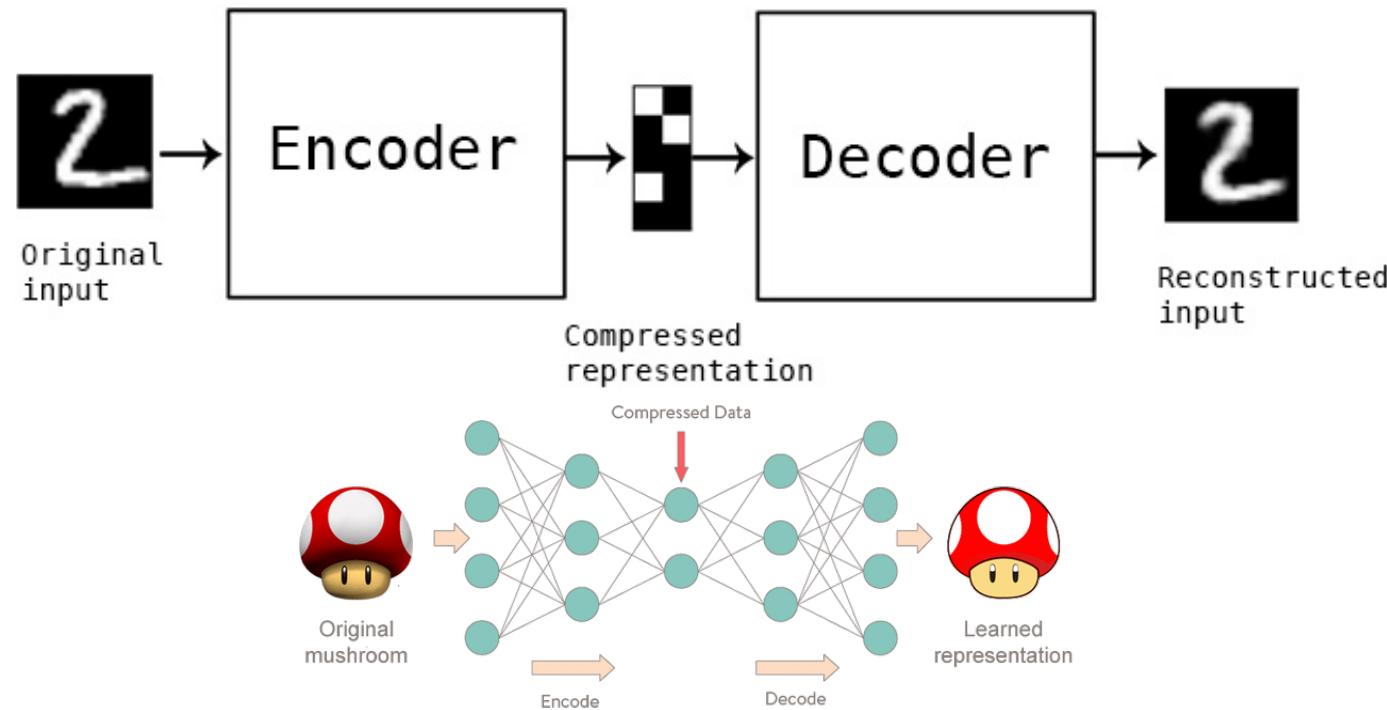
- Iris 데이터 셋으로 MLP를 구현해 봅니다.
 - soft-max란? 합계가 1이 되는 확률값의 집합
 - one-hot encoding : 예) 3→ 0,0,0,1
2→0,0,1,0





AutoEncoder 실습

AutoEncoder의 구조



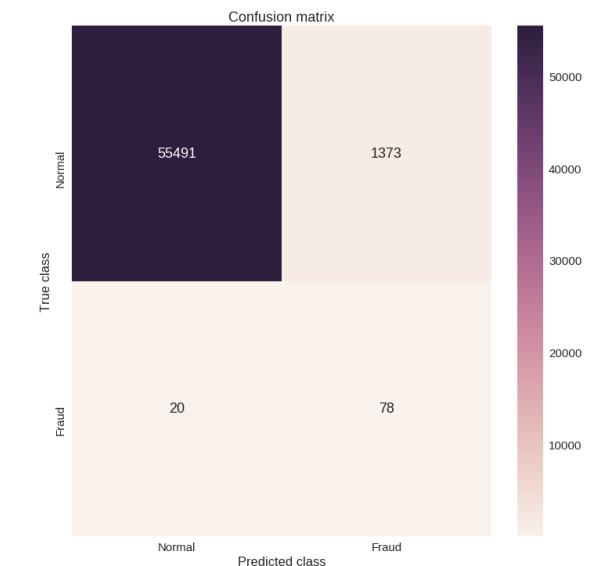
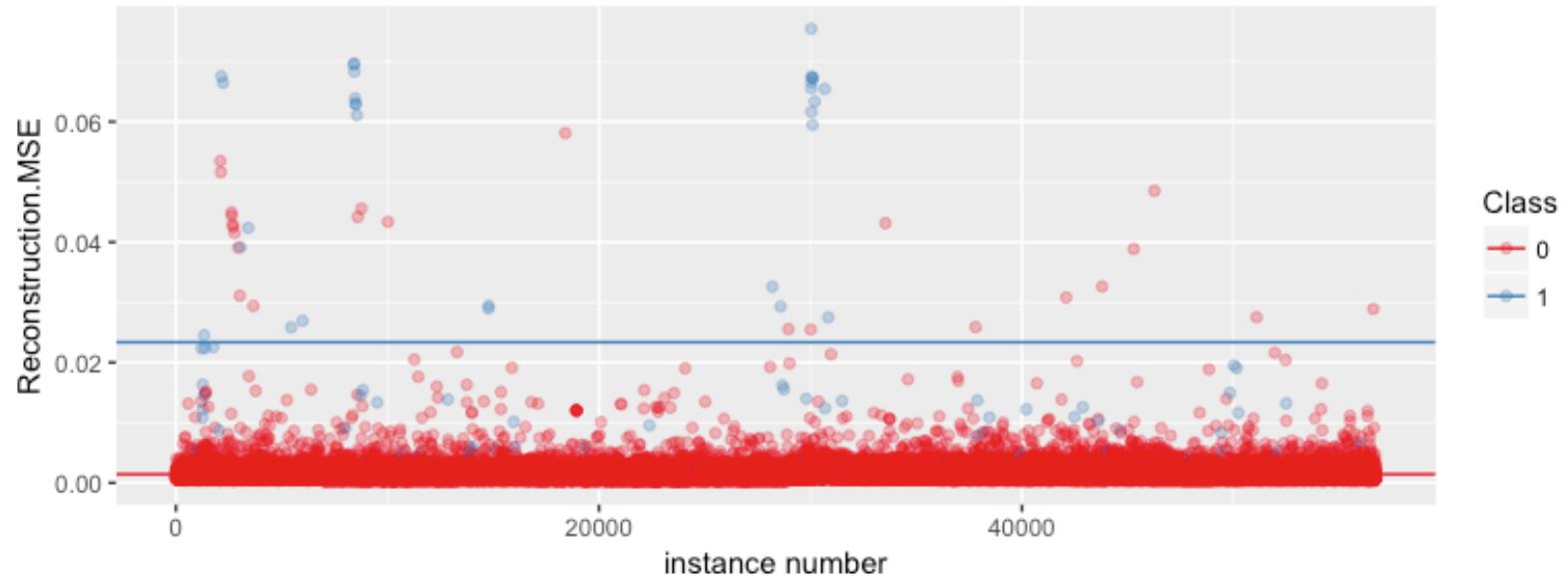
- 자기 자신을 학습하는 Unspervised Learning.
- Encoder - Comporessed Representation(BottleNeck) - Decoder로 이루어져 있음.



AutoEncoder의 활용

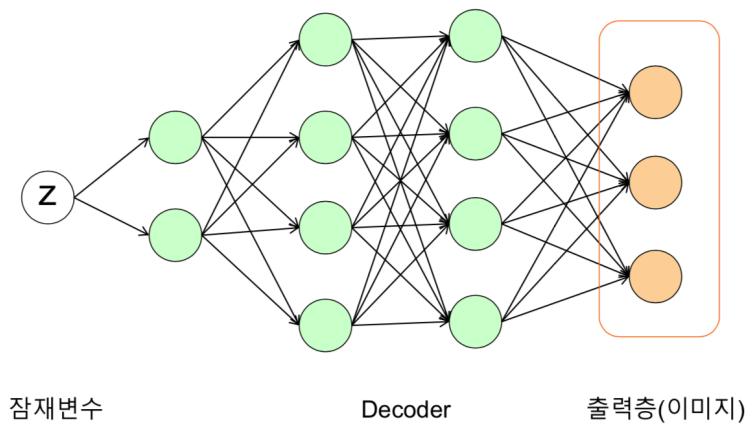
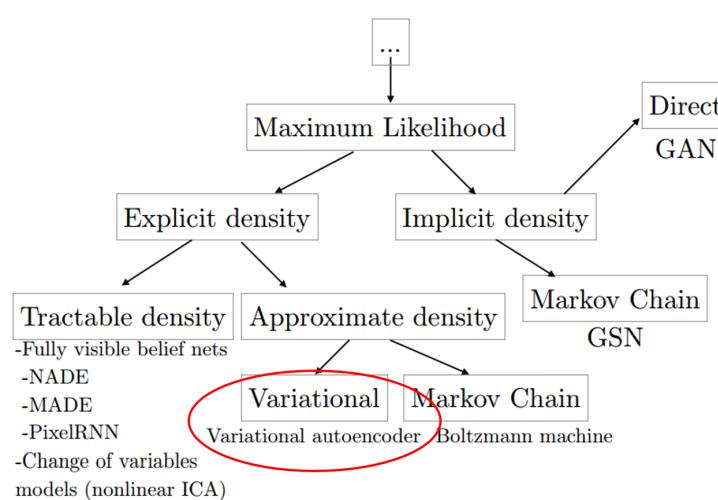
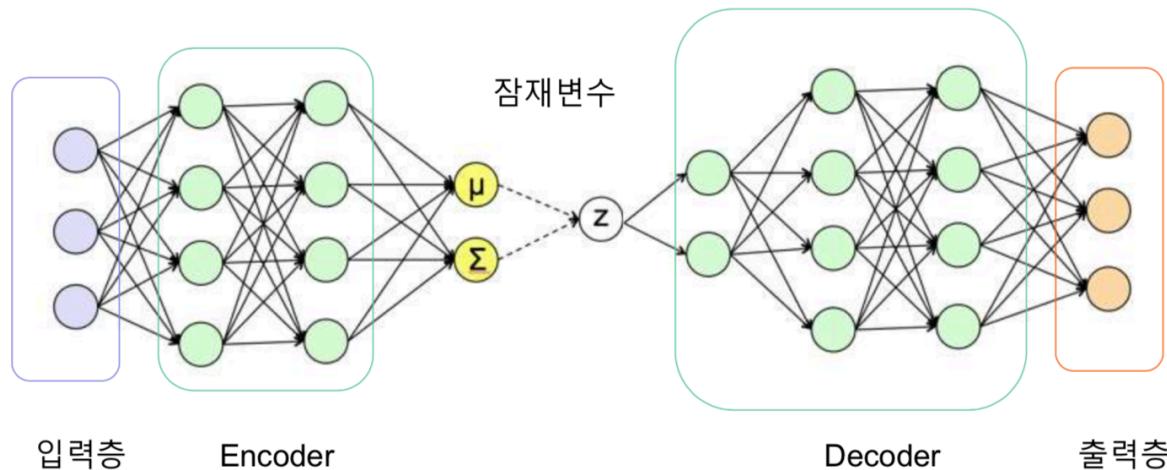
- Denoising autoencoder : 손상된 일부의 input 를 가지고 training해서 복구된 original input 을 output으로 만듦
- Unsupervised Pretraining : SDAE(Stacked Denoising Autoencoder)로 pretraining한 후 supervised DBN에 적용
- Variational AutoEncoder : AutoEncoder를 생성 모델로 활용

AutoEncoder의 활용 예

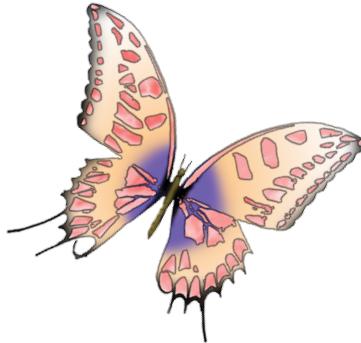


- Auto Encoder로 정상을 학습 시킨 모델에 비정상 데이터를 입력했을때 생성된 데이터와 입력데이터간 Error가 더 크게 발생함.

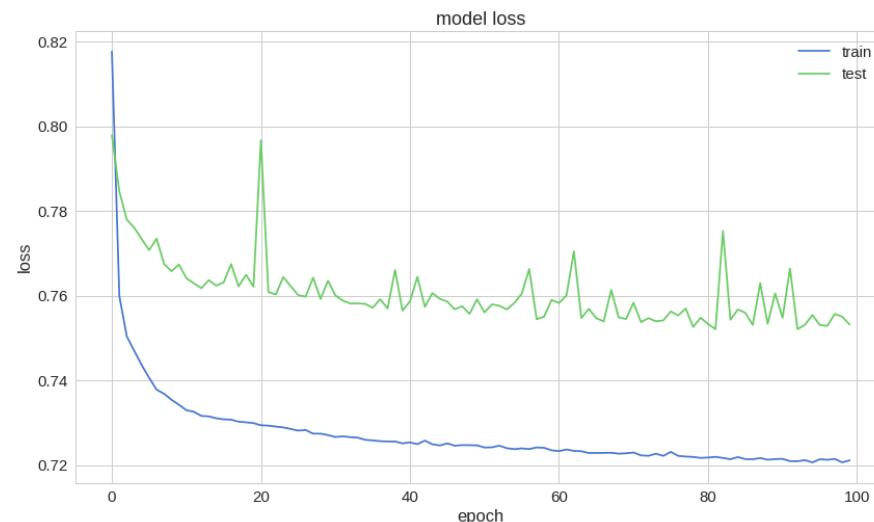
생성모델의 이해



AutoEncoder 실습



1. AutoEncoder로 mnist 이미지를 트레이닝 해보고, Original 이미지와 비교해 본다.
2. Fraud Detection 예제를 통해 AutoEncoder의 Reconstruction Error가 어떻게 활용되는지 알아본다.





Anomaly Detection

AutoEncoder 작동 원리 이해(심화)
각종 Auto Encoder 소개 및 적용 방법
모델 성능 개선을 위한 인코딩 방법 고도화

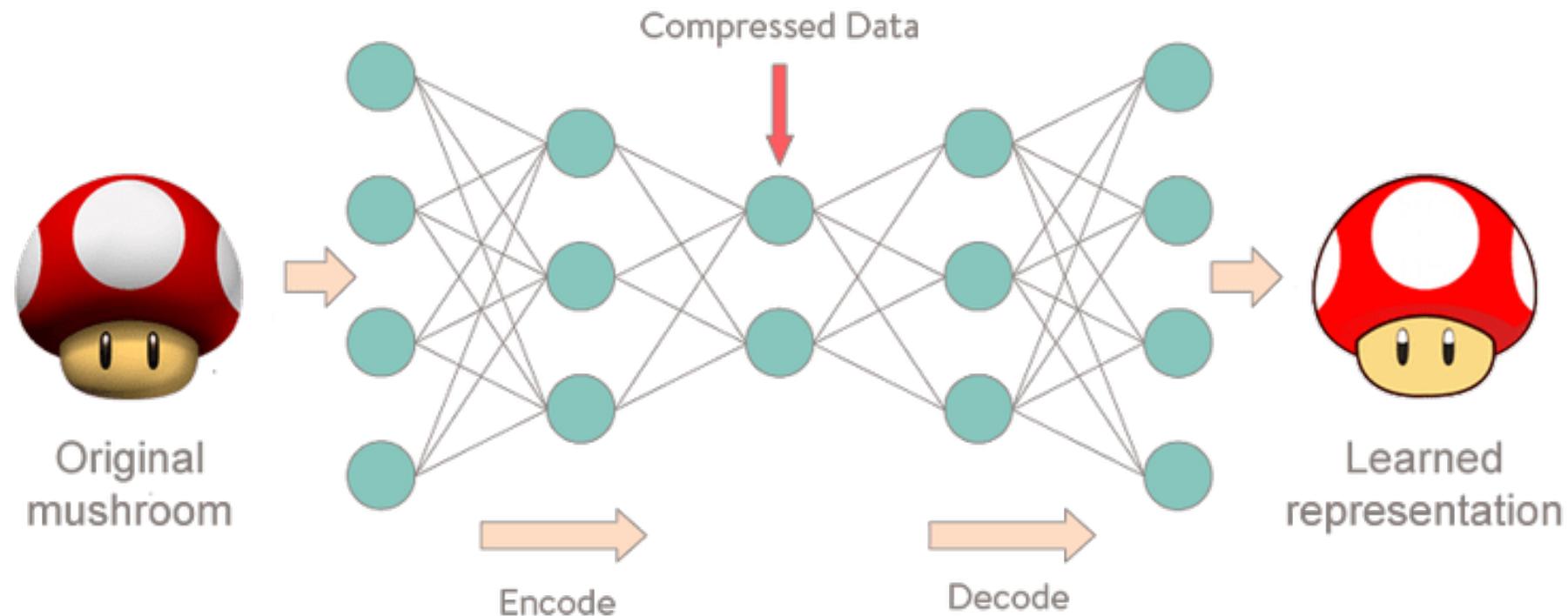


AutoEncoder 작동 원리 이해(심화)



Autoencoder 특성

- 1. 입력과 출력은 같다. (MLP와 네트워크 비슷)
- 2. 구성 : 인코더(encoder)와 디코더(decoder), 두 부분으로 구성





Autoencoder 작동 원리

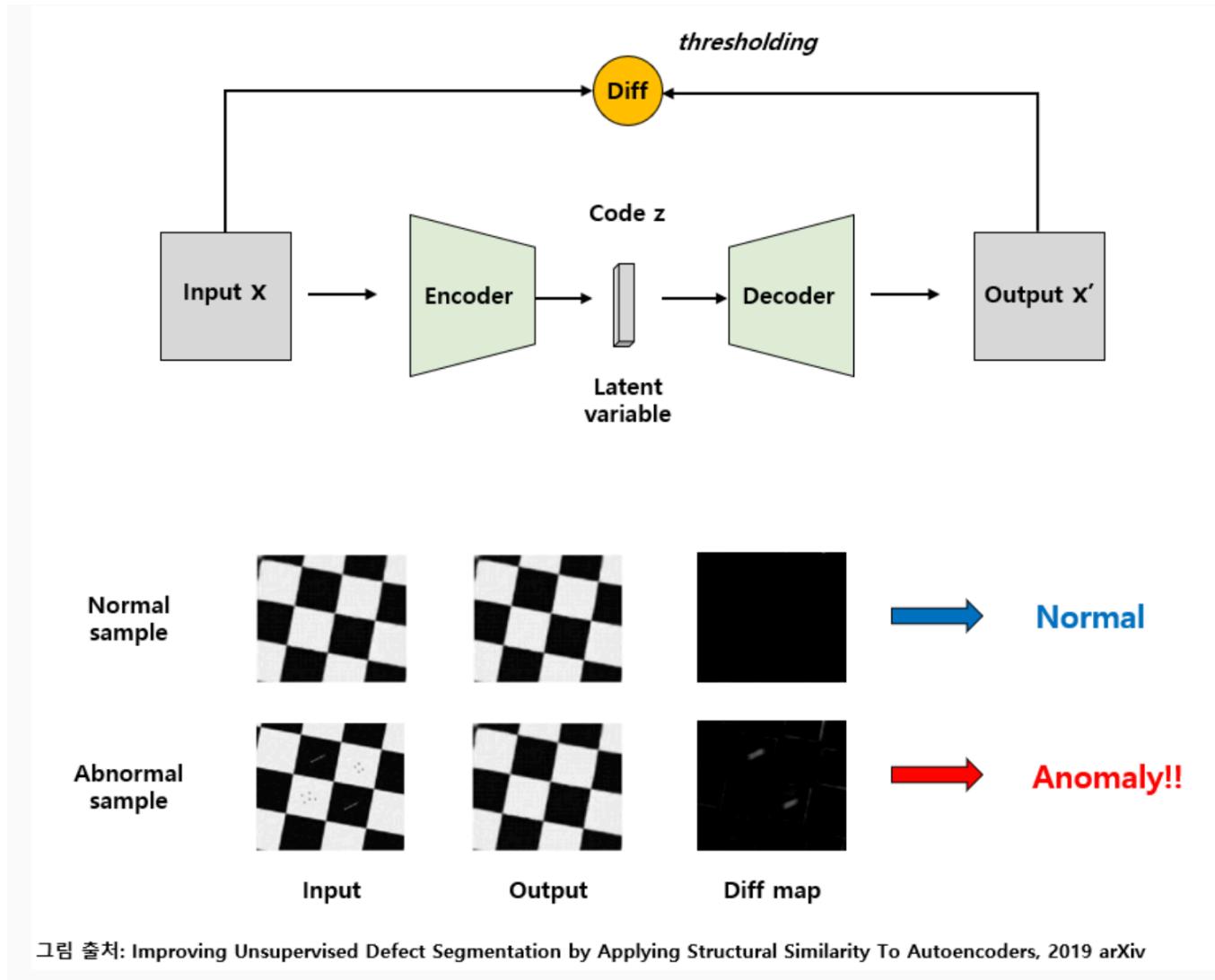
- 인코딩 과정에서 입력된 데이터의 핵심 Feature 정보만 Hidden Layer에서 학습하고, 나머지 정보는 손실시킴
- 디코딩 과정에서 Hidden Layer의 출력값을 뽑았을 때 완벽한 값 복사가 아닌 입력값의 근사치가 됨
- 출력값이 입력값과 최대한 같아지도록 튜닝함으로써, Feature를 잘 추출할 수 있게 하는 것이 오토인코더의 원리(입력값과 출력값이 최대한 비슷하게 만드는 가중치를 찾아냄)

Autoencoder의 학습 과정

1. 입력값과 Hidden Layer의 가중치를 계산해 Sigmoid 함수를 통과시킨다.
2. 1번의 결과물과 출력 레이어의 가중치를 계산해 Sigmoid 함수를 통과시킨다.
3. 2번의 값을 이용해 MSE(Mean Squared Error)를 계산한다.
4. 3번의 결과로 나온 Loss 값을 SGD로 최적화
5. 오류역전파를 사용하여 가중치를 갱신



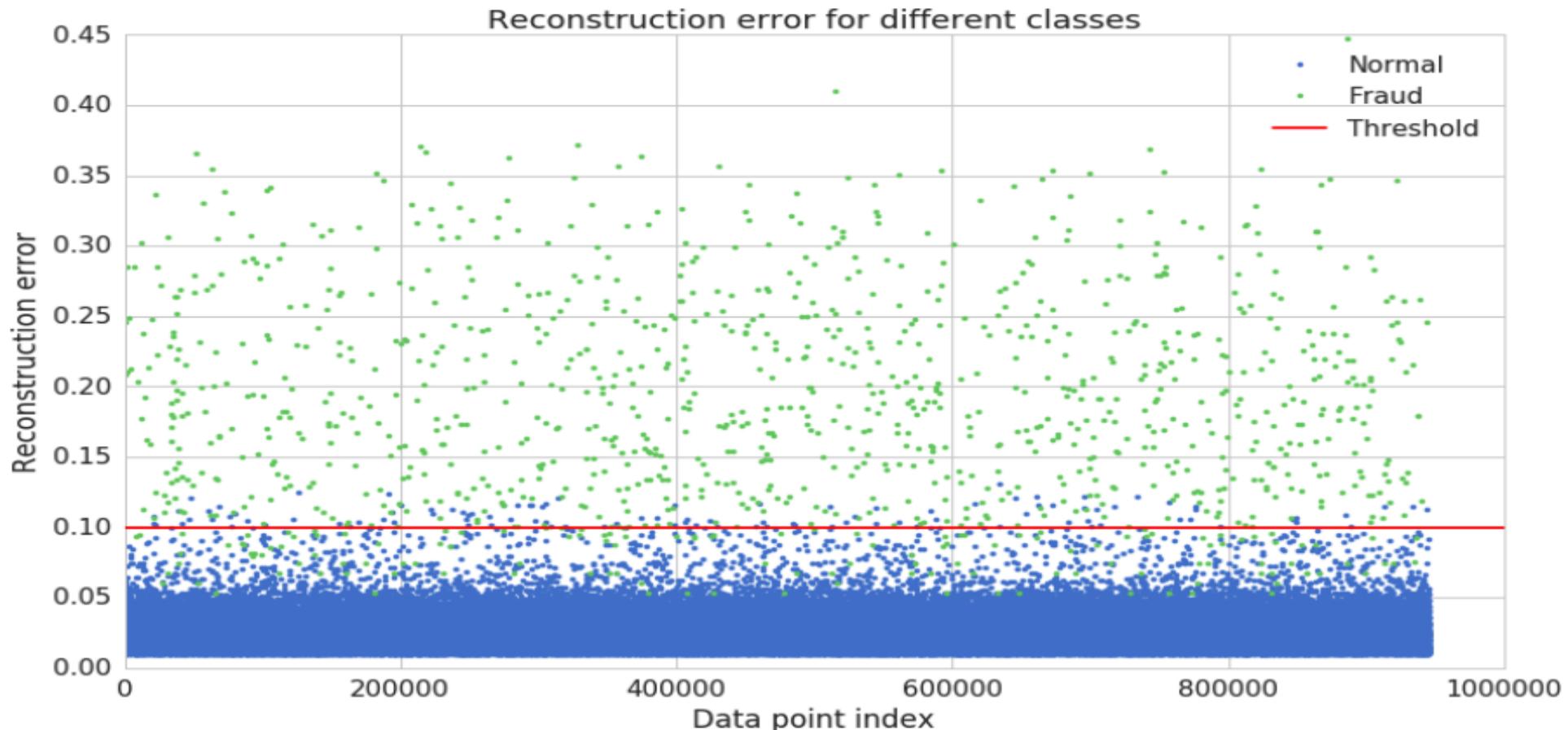
Autoencoder와 Anomaly Detection





Autoencoder와 Anomaly Detection

- 정상을 학습
- 구현시 재구성 Error가 크면 비정상이라고 간주함.

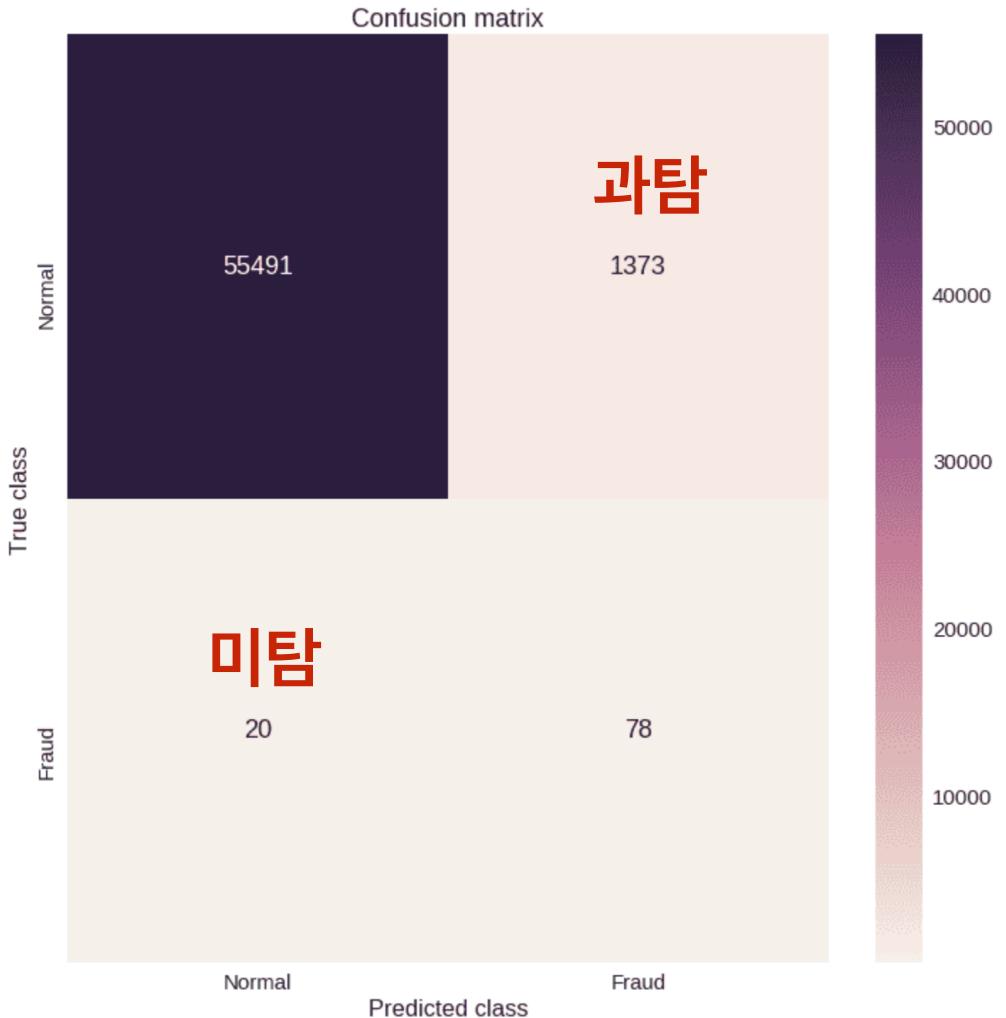




미탐과 과탐

- 전제에 따라 다름 : Normal이 Positive라면?

- FP(False Positive)가 미탐
- FN(False Negertive)가 과탐

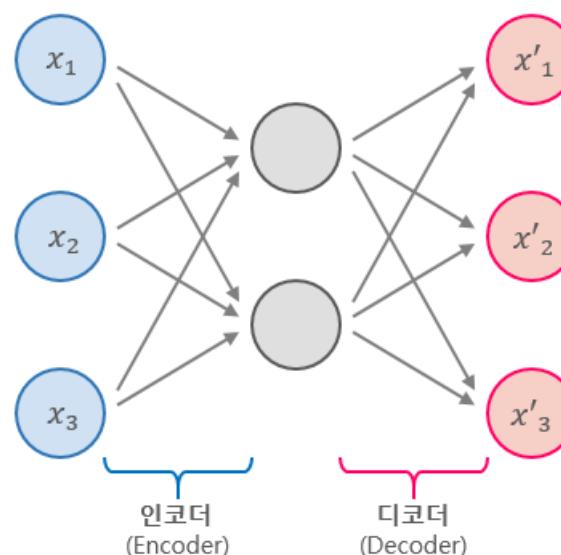




각종 Auto Encoder 소개 및 적용 방법

Uncomplete autoencoder

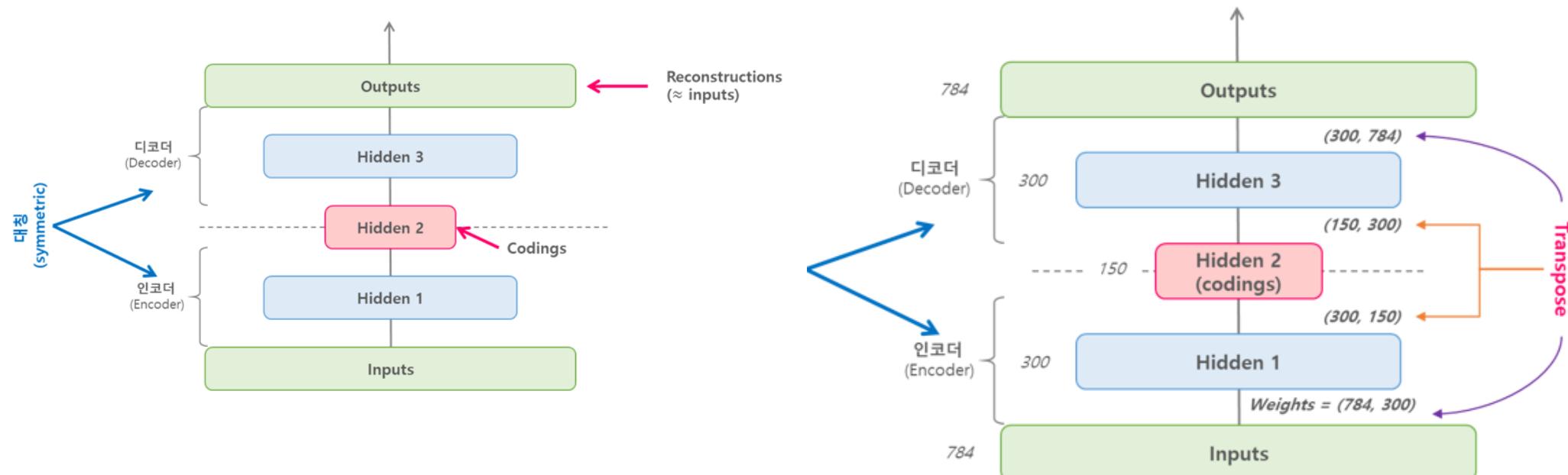
- 오토인코더는 입력을 재구성하기 때문에 출력을 재구성(reconstruction)이라고도 함
- 인코더(encoder) : 인지 네트워크(recognition network)라고도 하며, 입력을 내부 표현으로 변환
- 디코더(decoder) : 생성 네트워크(generative nework)라고도 하며, 내부 표현을 출력으로 변환
- 입력과 출력층의 뉴런 수가 동일
- 저차원을 가지는 하든 레이어에 의해 입력을 그대로 출력으로 복사할 수 없기 때문에, 출력이 입력과 같은 것을 출력하기 위해 학습
- 통해 undercomplete 오토인코더는 입력 데이터에서 가장 중요한 특성(feature)을 학습





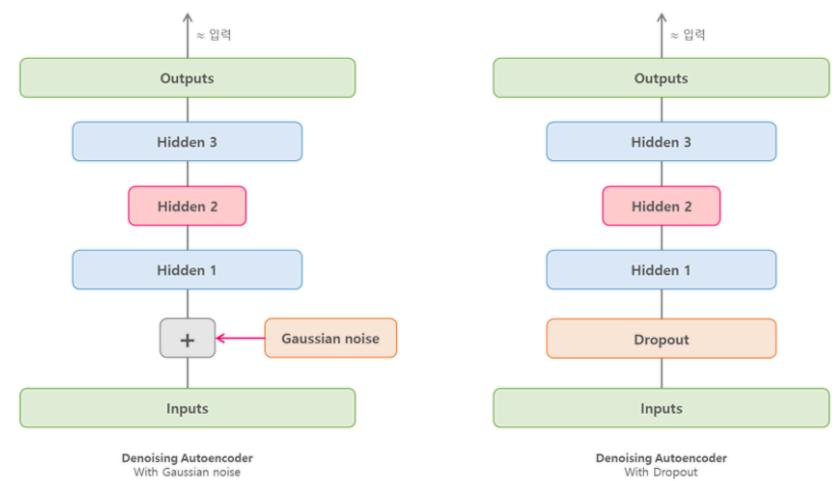
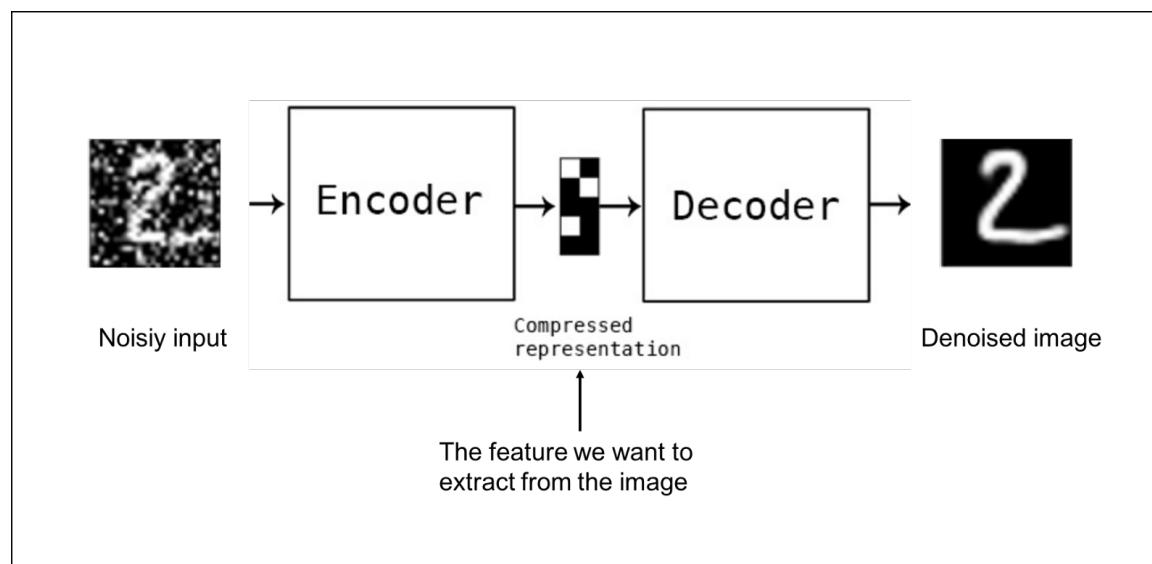
Stacked(Deep) Autoencoder

- Stacked 오토인코더 또는 deep 오토인코더는 **여러개의 히든 레이어**
- 레이어를 추가할수록 오토인코더가 더 복잡한 코딩(부호화)을 학습 가능
- stacked 오토인코더의 구조는 아래의 그림과 같이 가운데 히든레이어(코딩층)을 기준으로 대칭 구조



Denoising Autoencoder

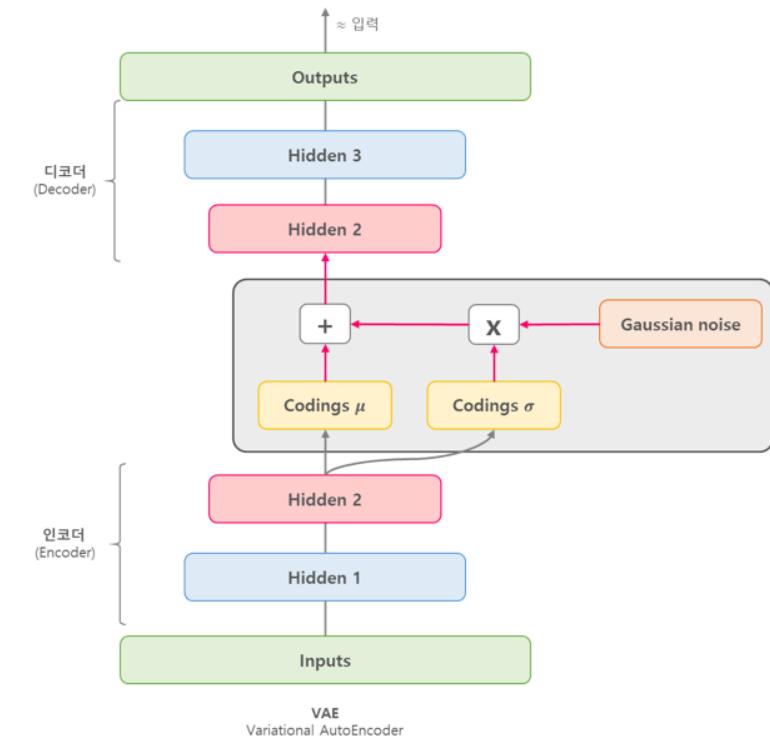
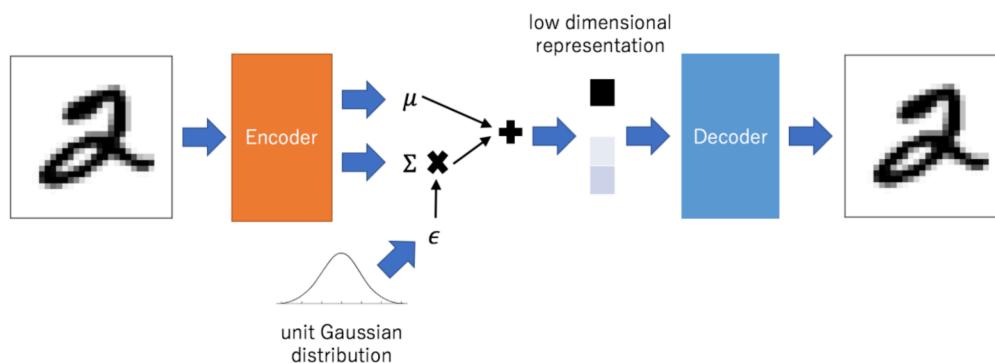
- 오토인코더가 의미있는 특성(feature)을 학습하도록 제약을 주는 다른 방법은 입력에 노이즈(noise, 잡음)를 추가
- 노이즈가 없는 원본 입력을 재구성하도록 학습시키는 것
- 노이즈 발생 방법(아래의 그림)
 - 입력에 가우시안(Gaussian) 노이즈를 추가
 - 드롭아웃(dropout)처럼 랜덤하게 입력 유닛(노드)를 끔





Variational AutoEncoder (VAE)

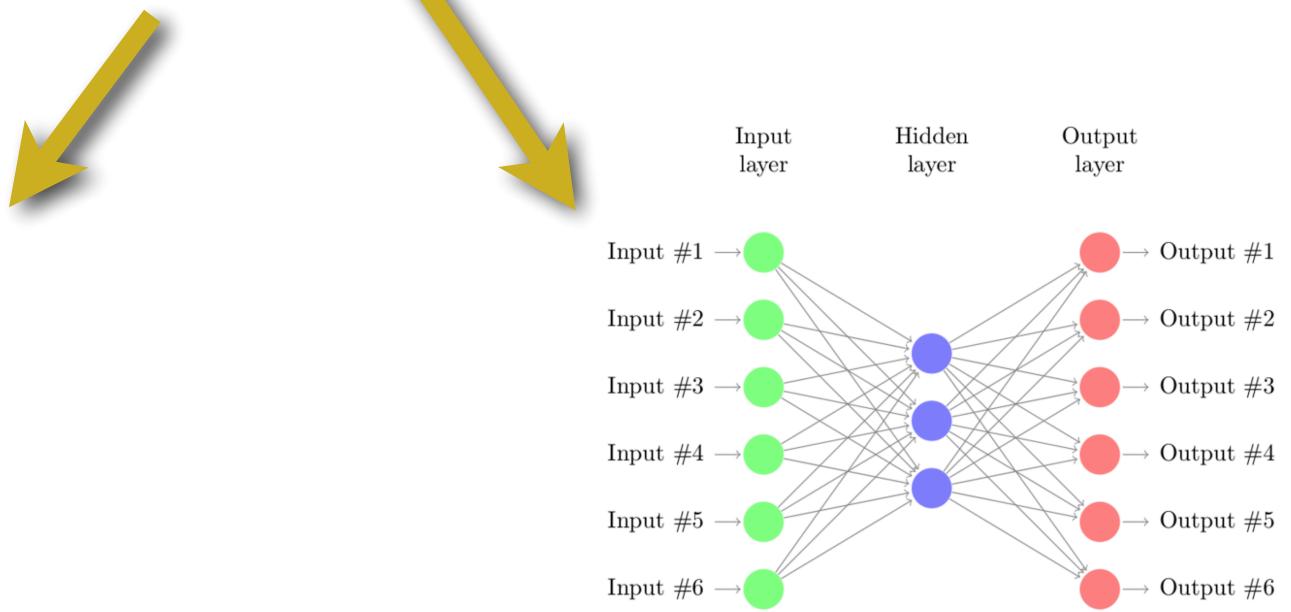
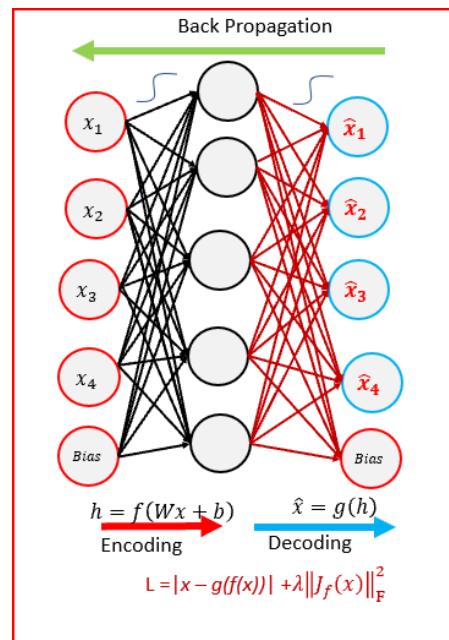
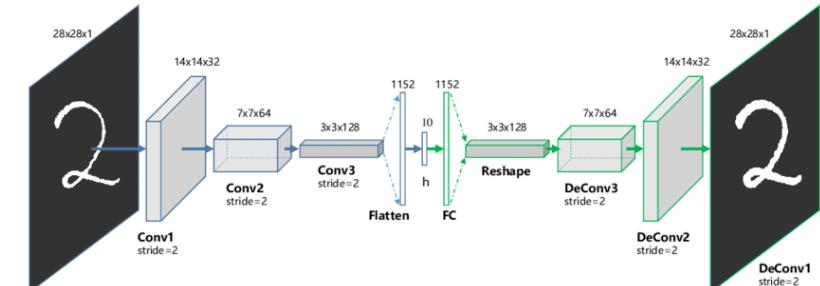
- VAE는 확률적 오토인코더(probabilistic autoencoder)
 - 학습이 끝난 후에도 출력이 부분적으로 우연에 의해 결정된다.
- VAE는 생성 오토인코더(generative autoencoder)
 - 학습 데이터셋에서 샘플링된 것과 같은 새로운 샘플을 생성할 수 있다.
- 평균이 μ 이고 표준편차가 σ 인 가우시안 분포(gaussian distribution)에서 랜덤하게 샘플링되며, 이렇게 샘플링된 코딩을 디코더(decoder)가 원본 입력으로 재구성
- 즉, Decoder를 Generator로 사용면 생성모델의 성능이 좋아진다.



그 밖의 Autoencoder



- Convolutional Autoencoder
- Sparse Autoencoder
- Contractive Autoencoder



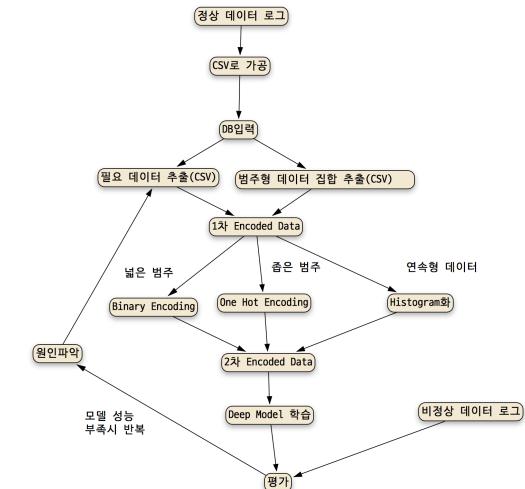


모델 성능 개선을 위한 인코딩 방법 고도화



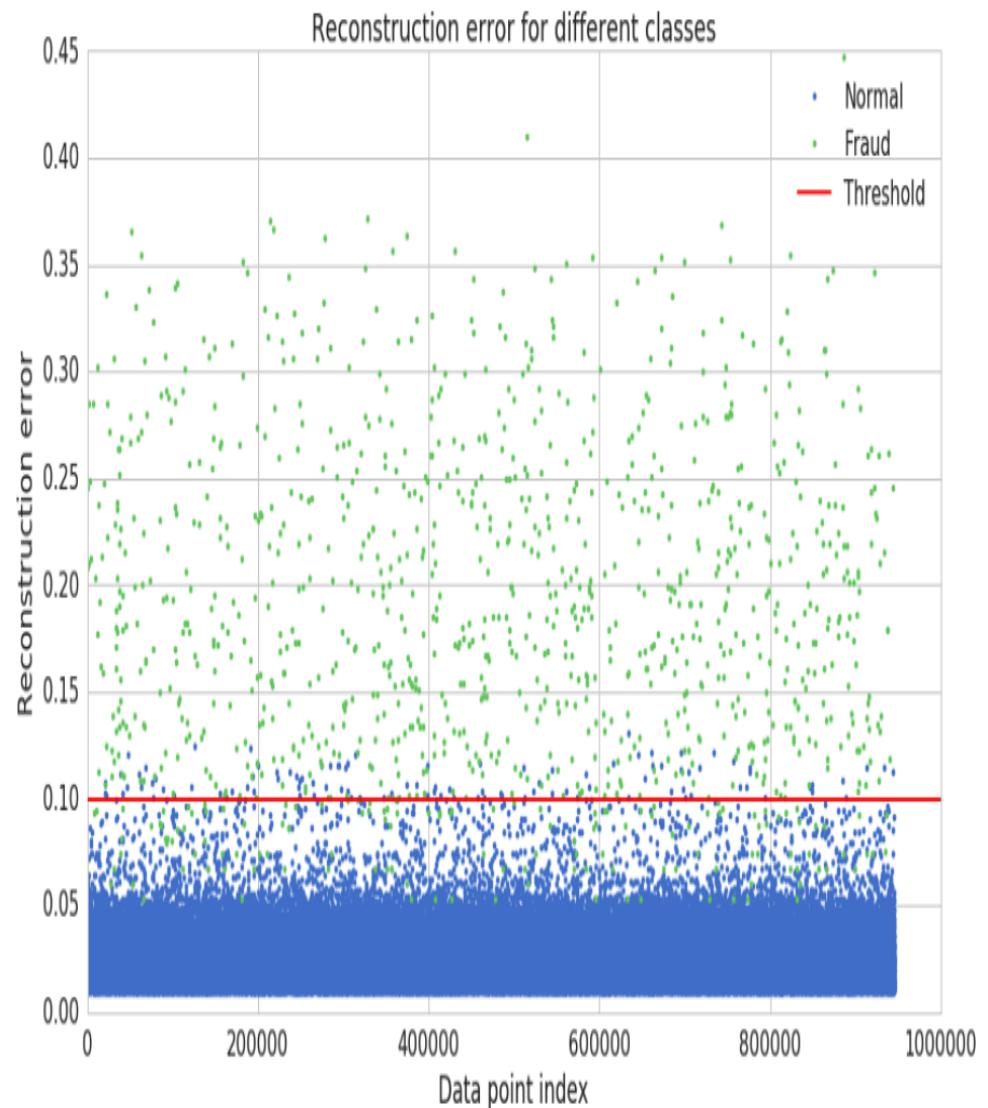
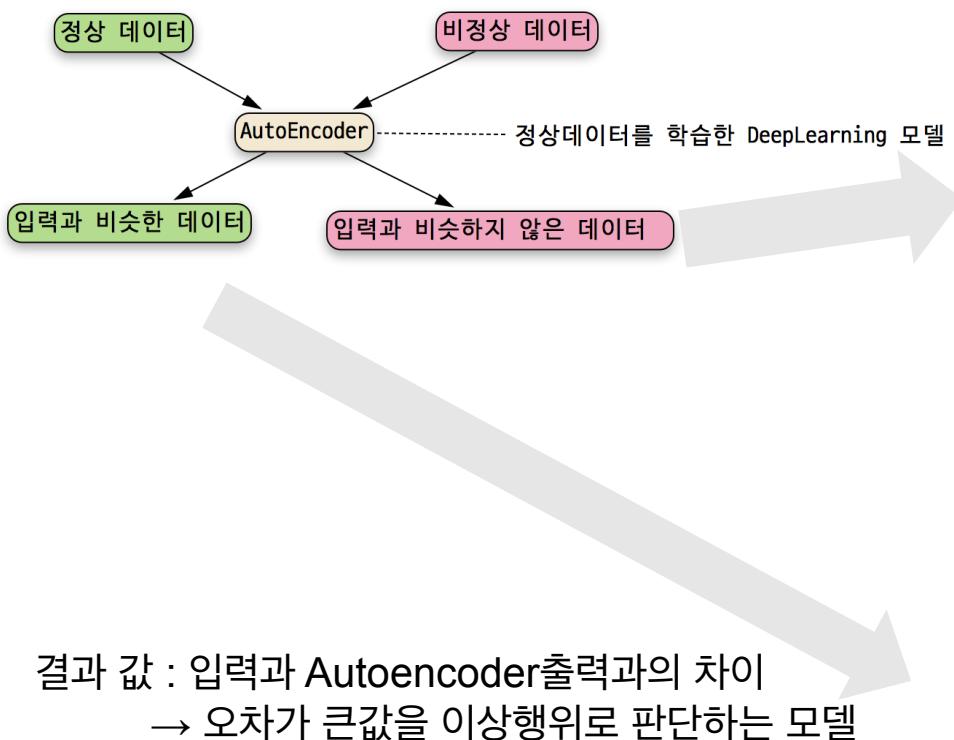
데이터 분석 흐름

1. 정상 데이터 로그 : DB입력을 위해 CSV형태로 가공
2. DB입력 : 변환된 CSV파일을 DB에 입력
 - a. 필요 데이터 추출 : Feature 값 추출 (연속형 데이터)
 - b. 범주형 데이터 집합 추출 : 범주형 데이터를 인코딩하기 위한 중복 제거 리스트
3. 1차 Encoded Data : 정수와 실수만으로 이루어진 데이터
 - a. Binary Encoding : 넓은 범주의 경우 ex) 0x0F → 00001111 과 같은 형태로 인코딩
 - b. One Hot Encoding : 데이터가 몇개 안되는 경우 ex) 03 → 00100과 같은 형태로 인코딩
 - c. Histogram화 : 편차가 심한 데이터의 경우 Histogram으로 표현하고 이를 입력으로 사용
4. 2차 Encoded Data : 위 3가지 인코딩 방식으로 Deep Learning Model이 학습할 수 있는 형태로 변
5. Deep Learning 모델 학습 : Unsupervised Model을 통한 학습
6. 평가 : 실제 값(비정상 로그들)과 비교하여 모델의 성능(정합성)을 평가함.
7. 원인 파악 : 모델의 성능이 나오지 않을 경우 데이터의 추가적으로 필요한 피쳐 값을 찾음.





모델 적용





데이터 전처리 예

2		50612	17	8 80	GET	-1 -1	http://h
2		49306	17	8 80	GET	-1 -1	http://h
2		33256	17	8 80	GET	-1 -1	http://h
2		53952	17	8 80	GET	-1 -1	http://h
1		54450	17	9 80	POST	-1 -1	http://h
1		54452	17	9 80	POST	-1 -1	http://h
2		54475	17	4 80	PUT	-1 -1	http://h
1		42958	17	4 80	PUT	-1 -1	http://h
1		32997	17	4 80	PUT	-1 -1	http://h
1		62923	17	4 80	PUT	-1 -1	http://h



범주형 데이터 집합 추출 : { 0: "GET" , 1:"POST", 2:"PUT"}]

One Hot Encoding : 0 → [1, 0, 0] 1 → [0, 1, 0] 2 → [0, 0, 1]

Binary Encoding : 3 → [0, 1, 1] 7 → [0, 0, 0] 1024 → [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Range Encoding : x<10 → [1, 0, 0], x<20 → [0, 1, 0], 20>=x → [0, 0, 1]

0	50612	1	8 80	0 -1 -1 http:
1	49306	1	8 80	0 -1 -1 http:
2	33256	1	8 80	0 -1 -1 http:
3	53952	1	8 80	0 -1 -1 http:
4	54450	1	9 80	1 -1 -1 http:
5	54452	1	9 80	1 -1 -1 http:
6	54475	1	4 80	2 -1 -1 http:
7	42958	1	4 80	2 -1 -1 http:

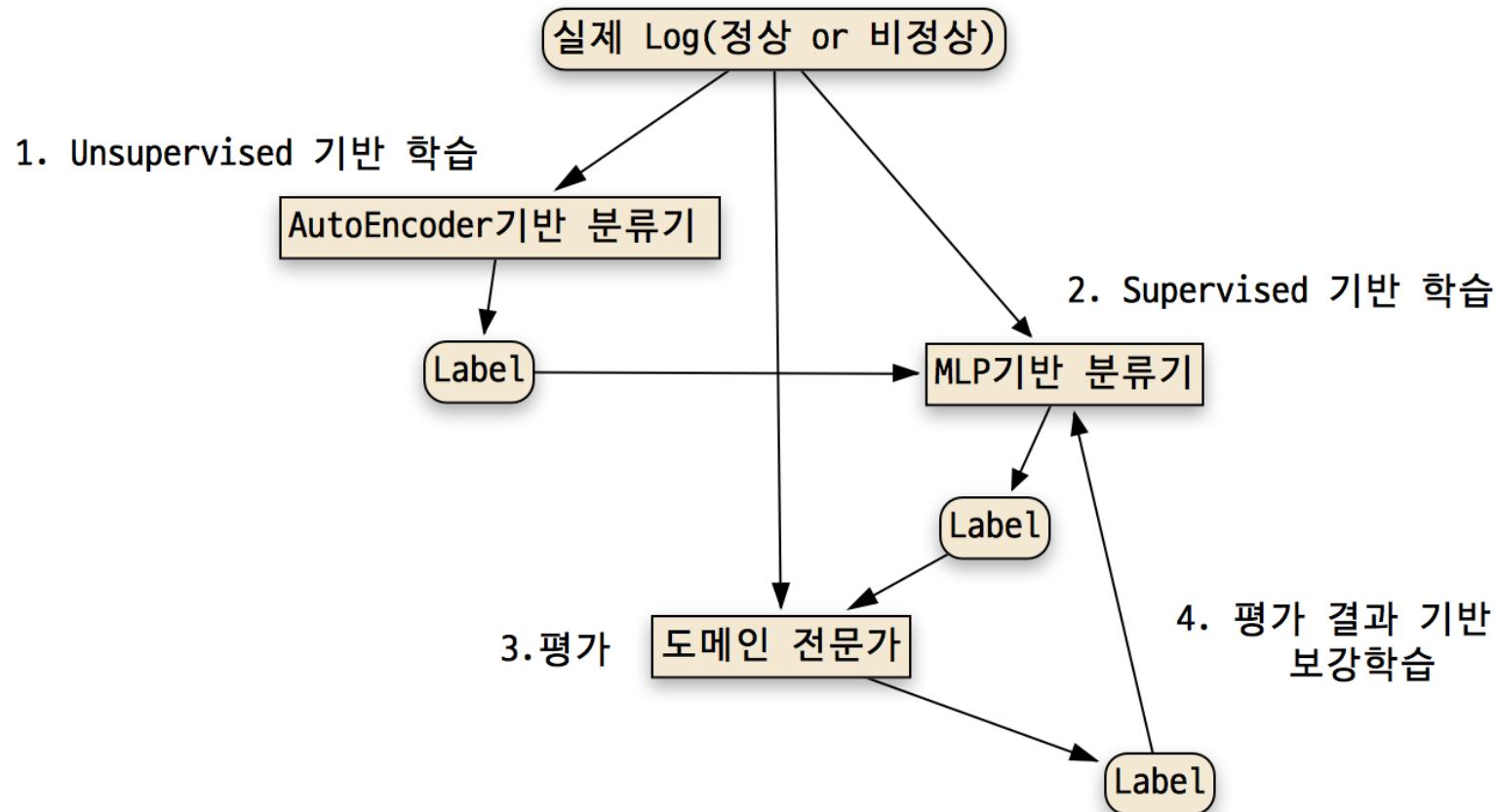
최종 Input 데이터

0	1	2	3	4	5	6	7	8	9	...	134	135	136	137	138	139	140	141	142	143
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0



모델 고도화 예

- 실제 데이터를 본 Unsupervised Model의 결과에 대입하여 1차 레이블된 데이터 셋을 만들고 이를 기반으로 MLP모델을 1차 학습시키고, 이 결과를 도메인 전문가가 오류 레이블을 찾아 내어 이를 기반으로 MLP모델을 2차 학습시켜 이를 반복적으로 시행하여 모델의 정합성을 올려 가는 방법.

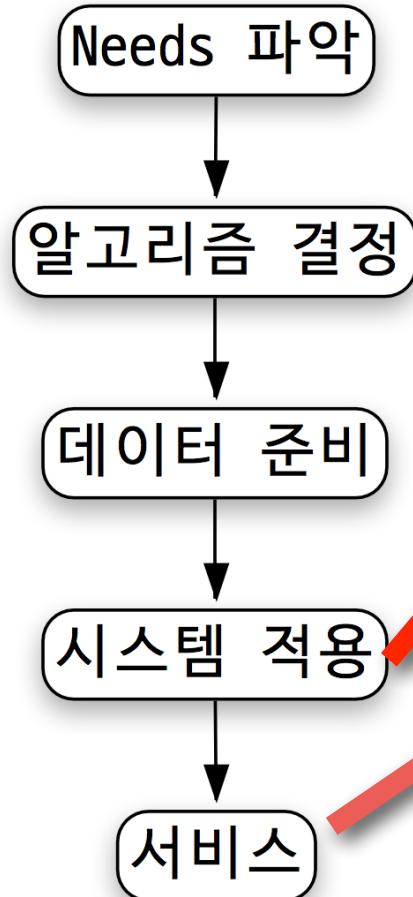




Wrap-Up



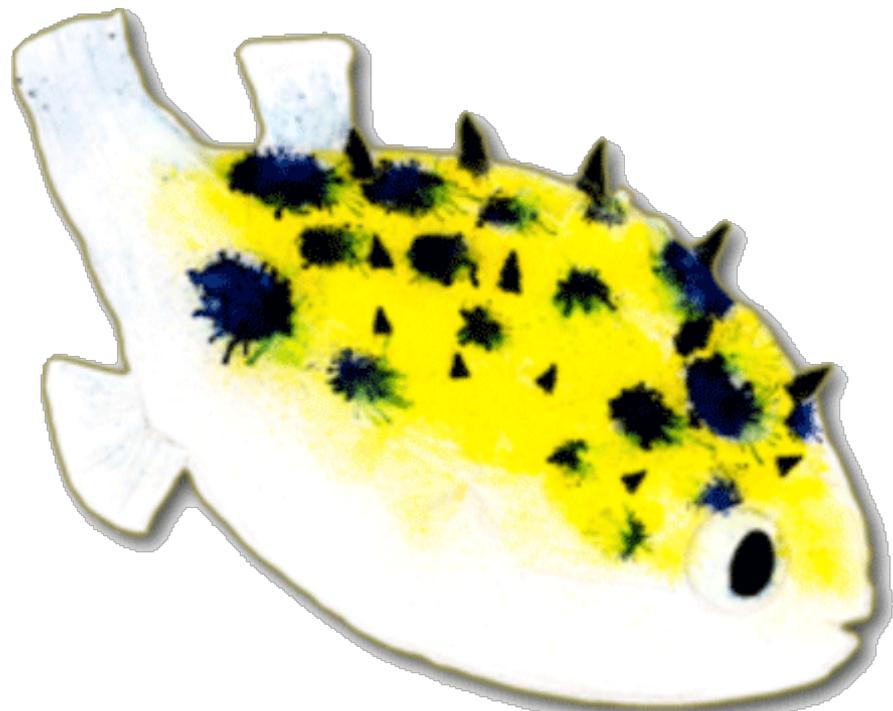
Analytic Process



- 1단계 : 데이터 로딩
- 2단계 : 학습 데이터/ 평가 데이터로 분리
- 3단계 : 학습(Training)
- 4단계 : 평가
- 5단계 : 모델 저장
- 6단계 : 서비스 활용

Decision Tree Classifier
Bayesian Classifier
Logistic regression
Support Vector Machines
MLP
CNN
RNN
K-Means Clustering
Hierarchical Clustering
Association Rule
Collaborative Filtering
Autoencoder

Thank You !!



Question!!

Solution!!

The End

