파이썬을 이용한 데이터과학과 머신러닝 입문

Unsupervised Machine Learning
- Clustering & Decomposition

August, 2019

Ein Intelligence

1

CONTENTS

- Intro to Unsupervised ML
- Clustering
- Decomposition

2

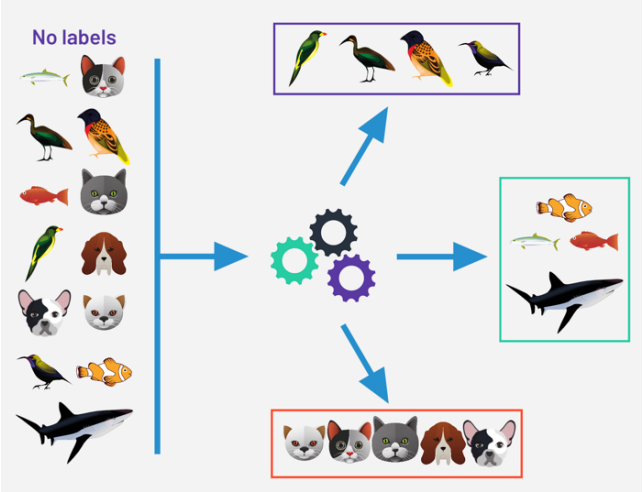
Introduction to Unsupervised

Machine Learning

3

비지도학습(Unsupervised Learning) 이란?

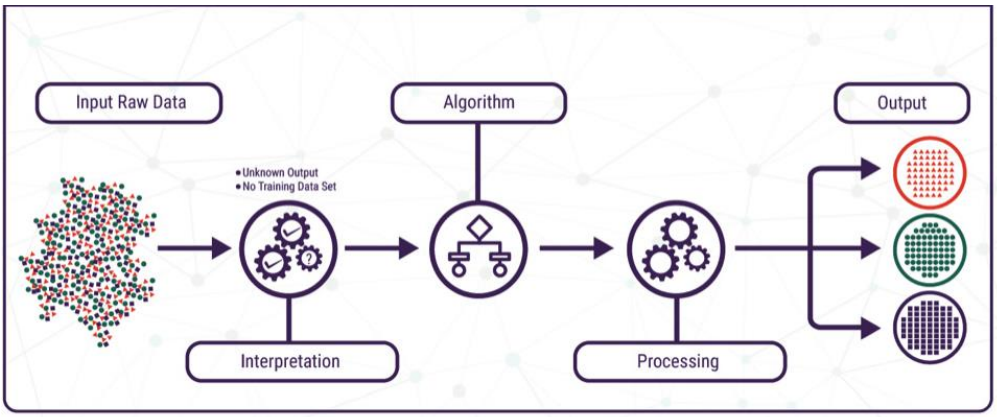
- 주어진 정답(Label)이나 비교의 기준이 되는 절대적인 기준이 없는 경우,
- 주어진 데이터 내에서 상대비교를 통해서 입력데이터의 패턴, 특성 등을 학습하게 됨



4

비지도학습(Unsupervised Learning)의 특징

- 비지도학습은 데이터 구조를 분석하고 특징을 추출하여 처리
- 정답이 없기 때문에 정확도 등에 대한 측정(Metric)이 어려움
- Clustering, Anomaly Detection, Association, Auto-Encoders 등의 방법이 있음



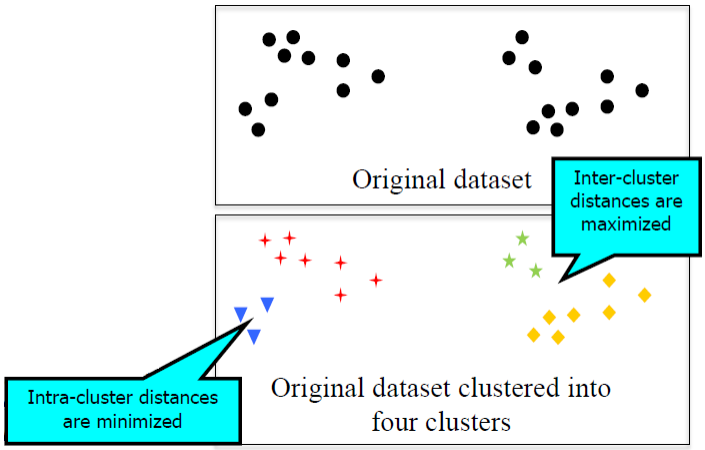
5

Clustering

6

군집분석(Clustering)

- 데이터 세트를 그룹들로 나누어 묶는 방법을 찾는 것
- 데이터 간격이 멀거나 달라야 분리가 가능



7

K-Means Algorithm

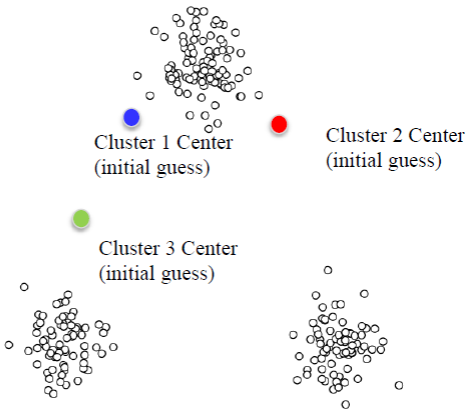
The k-means algorithm

Initialization Pick the number of clusters *k* you want to find. Then pick *k random* points to serve as an initial guess for the cluster centers.

Step A Assign each data point to the nearest cluster center.

Step B Update each cluster center by replacing it with the mean of all points assigned to that cluster (in step A).

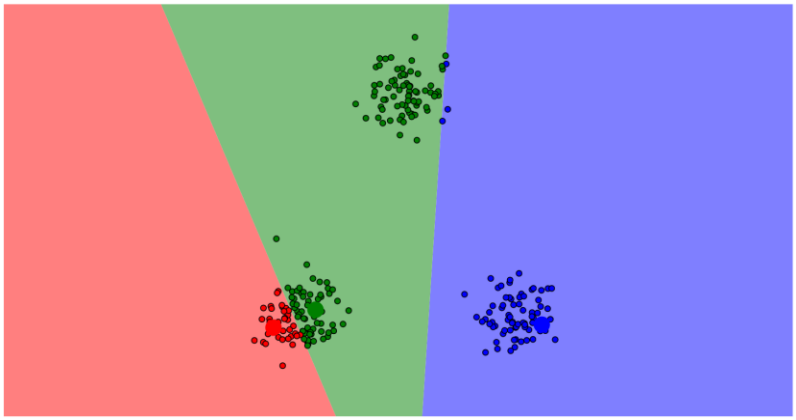
Repeat steps A and B until the centers converge to a stable solution.



8

데모: K-Means Cluster – Step 1A

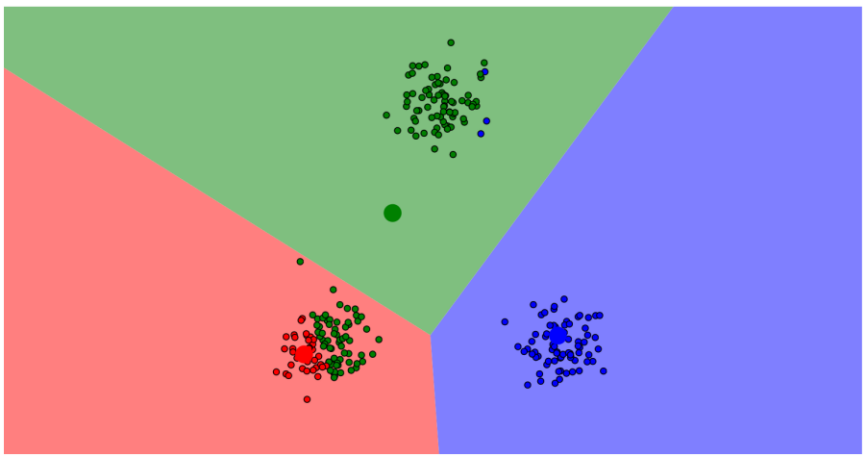
- https://www.naftaliharris.com/blog/visualizing-k-means-clustering/
- Random initial centeroids + Gaussian mixture + 3 centroids를 골랐을 때
  → 3개의 임의의 centers로부터 가까운대로 영역이 배정됨



9

데모: K-Means Cluster – Step 1B

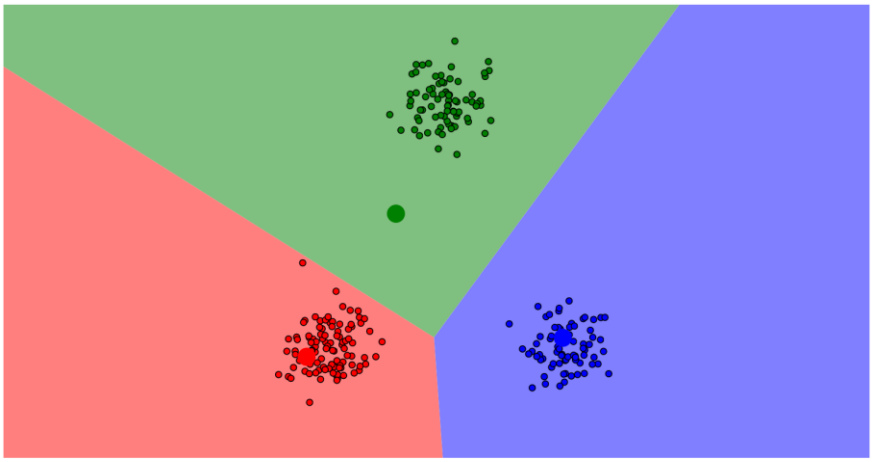
- 해당 영역의 중심에 해당하도록 3개 centeroids의 위치가 적절하게 update 됨



10

데모: K-Means Cluster – Step 2A

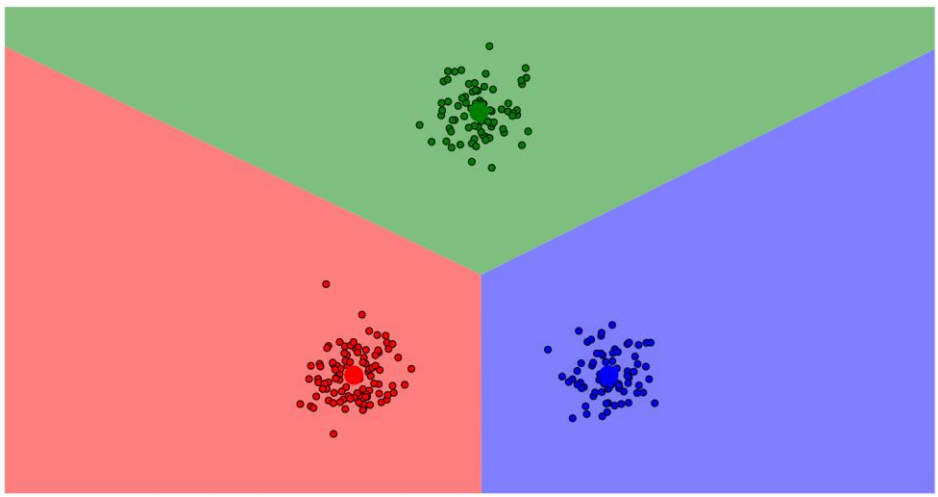
- Update된 3개 centers의 위치로부터 가까운대로 영역과 각 데이터 색깔이 재분류됨



11

데모: K-Means Cluster – Step 2B

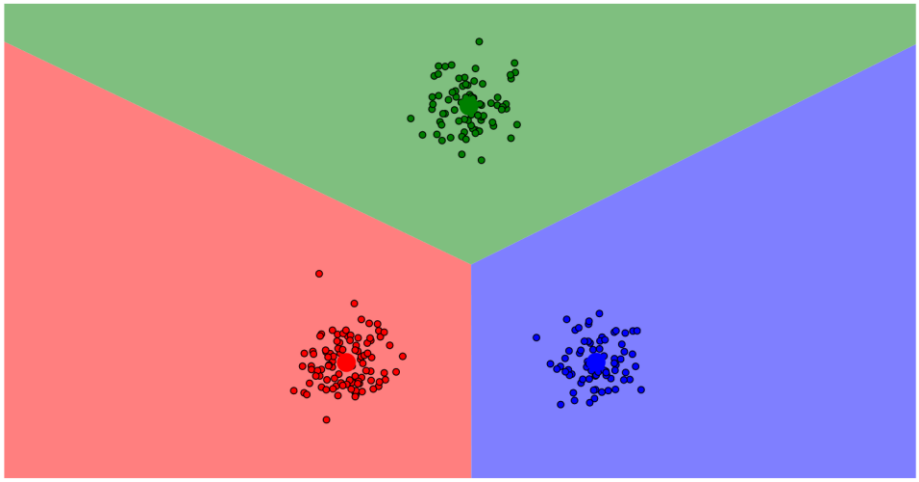
- Cluster center 가 다시 계산되어 centeroids가 update 됨



12

데모: K-Means Cluster – Converged

- 앞과 같이 몇번을 반복하여 더 이상 변화가 없는 상태에 이르게 됨 : clustering 완료

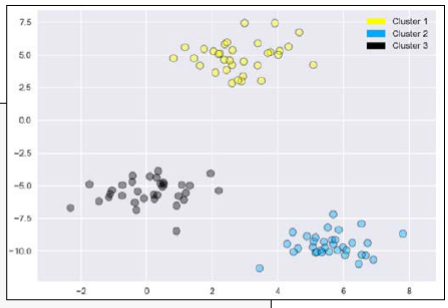


13

K-means Clustering Example

- Random_state 를 사용한 clustering 연습

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from adspy_shared_utilities import plot_labelled_scatter

X, y = make_blobs(random_state = 10)

kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)

plot_labelled_scatter(X, kmeans.labels_, ['Cluster 1', 'Cluster 2', 'Cluster 3'])
```

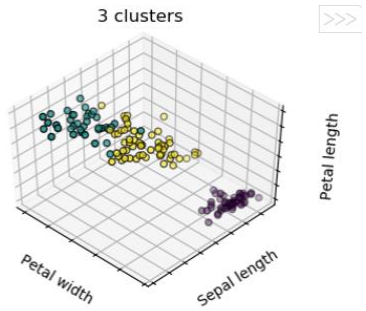


14

K-means Clustering Example : Iris Dataset

- Iris dataset 을 사용한 clustering 연습

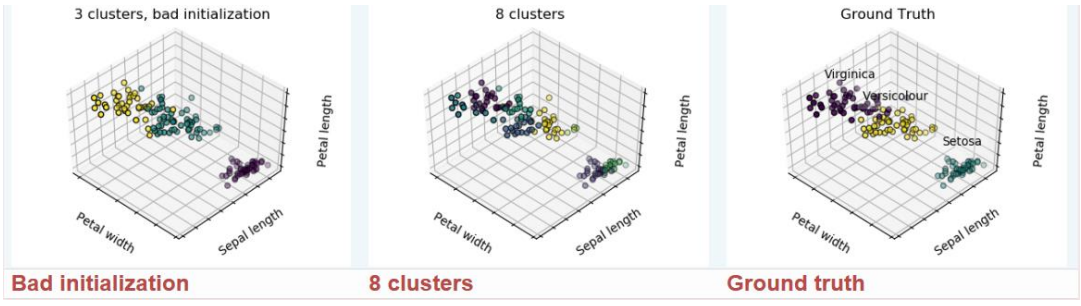
```
>>> from sklearn import cluster, datasets
>>> iris = datasets.load_iris()
>>> X_iris = iris.data
>>> y_iris = iris.target

>>> k_means = cluster.KMeans(n_clusters=3)
>>> k_means.fit(X_iris)
KMeans(algorithm='auto', copy_x=True, init='k-means++', ...
>>> print(k_means.labels_[::10])
[1 1 1 1 1 0 0 0 0 0 2 2 2 2 2]
>>> print(y_iris[::10])
[0 0 0 0 0 1 1 1 1 1 2 2 2 2 2]
```



** Ref) https://scikit-learn.org/stable/tutorial/statistical_inference/unsupervised_learning.html*

15

K-means Clustering Example : Iris Dataset



16

Decompositions

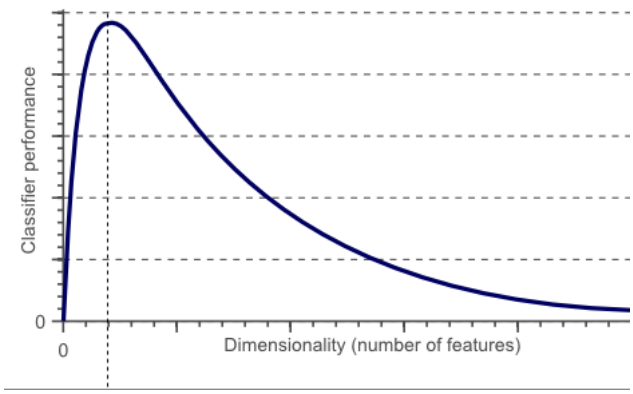
Dimensionality Reduction

17

차원을 줄여야 하는 이유

- 데이터의 features가 늘어남에 따라 분석의 차원이 높아져서, 모델이 점점 더 복잡해져서 시간이 오래 걸리고, overfitting 의 위험성이 커짐

차원의 저주 (The Curse of Dimensionality)



* Ref) https://towardsdatascience.com/dimensionality-reduction-for-machine-learning-80a46c2ebb7e

18

차원 복잡도를 줄이는 방법

- Feature Selection : 분석하려는 feature와 상관관계가 적은 feature 데이터 등은 제거하여 차원을 감소시킴

  - Variance Threshold, Univariate selection (Pearson Correlation, ANOVA, chi-square 등)

- Feature Engineering : 선형/비선형적인 변환(transformations) 기법을 적용

  - PCA (Principal Component Analysis), Factor Analysis, LDA (Linear Discriminant Analysis) 등

분산 한계치 기법 (Variance Threshold)

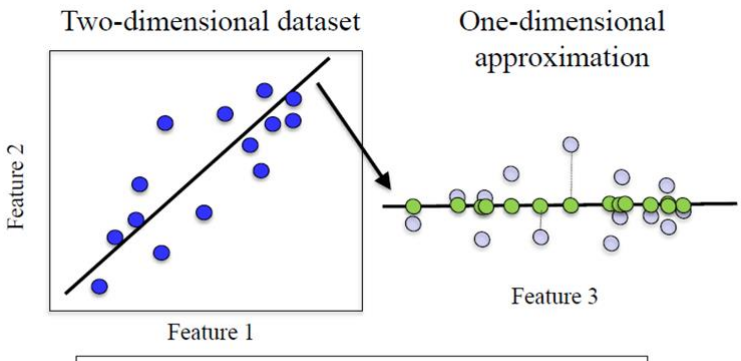
```
>>> X = [[0, 2, 0, 3], [0, 1, 4, 3], [0, 1, 1, 3]]
>>> selector = VarianceThreshold()
>>> selector.fit_transform(X)
array([[2, 0],
       [1, 4],
       [1, 1]])
```

* Ref) https://towardsdatascience.com/dimensionality-reduction-for-machine-learning-80a46c2ebb7e

19

차원 축소 (Dimensionality Reduction)

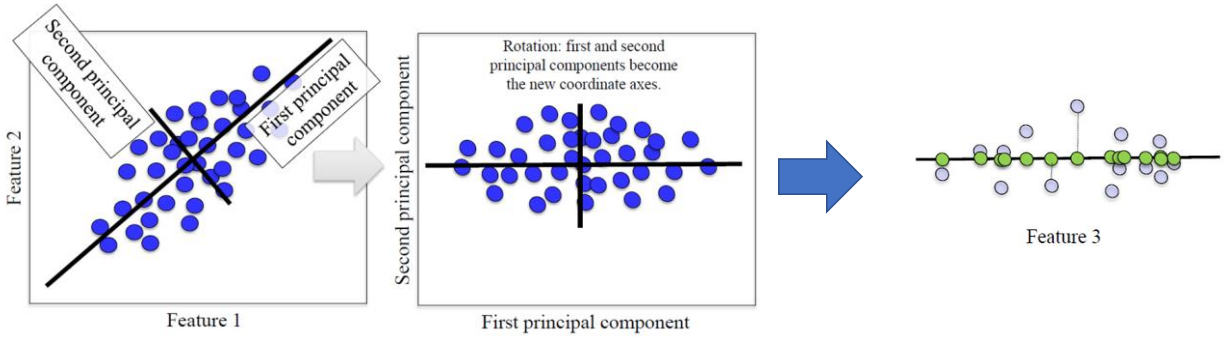
- 적은 수의 Features 이용하여 Dataset 약식 버전을 작성
- Dataset의 그룹핑이나 관계를 찾기 위해 주로 사용됨
- 보통 2차원 scatter plot으로 시각화 표현
- 압축 등에도 사용됨



The one-dimensional approximation is obtained by projecting the original points onto the diagonal line and using their position on that line as the new single feature.

20

간단한 PCA (Principal Component Analysis) 예



21

Decompositions: from a signal to components and loadings

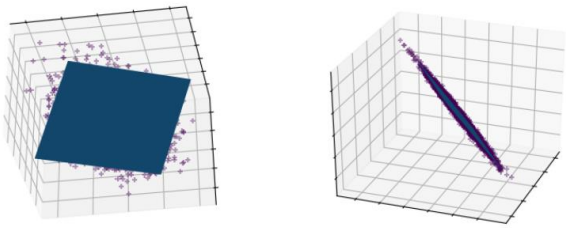
Components and loadings

If X is our multivariate data, then the problem that we are trying to solve is to rewrite it on a different observational basis: we want to learn loadings L and a set of components C such that $X = L\,C$. Different criteria exist to choose the components

Principal component analysis: PCA

Principal component analysis (PCA) selects the successive components that explain the maximum variance in the signal.

The point cloud spanned by the observations above is very flat in one direction: one of the three univariate features can almost be exactly computed using the other two. PCA finds the directions in which the data is not flat



** Ref) https://scikit-learn.org/stable/tutorial/statistical_inference/unsupervised_learning.html*

22

When used to transform data, PCA can reduce the dimensionality of the data by projecting on a principal subspace.

PCA (Principal Component Analysis)

```
[5]  # Create a signal with only 2 useful dimensions
     x1 = np.random.normal(size=100)
     x2 = np.random.normal(size=100)
     x3 = x1 + x2
     X = np.c_[x1, x2, x3]
```

```
[6]  X.shape
     X[:5,:]
```

```
array([[ 0.50705492,  0.81203756,  1.31909248],
       [ 1.7209502 ,  0.90742505,  2.62837525],
       [-0.63530527, -1.58960194, -2.22490721],
       [-1.79359549,  1.00322039, -0.7903751 ],
       [ 1.33961366,  0.29727378,  1.63688744]])
```

```
[7]  from sklearn import decomposition
     pca = decomposition.PCA()
     pca.fit(X)
```

```
PCA(copy=True, iterated_power='auto', n_components=None, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

```
[8]  print(pca.explained_variance_)
```

```
[3.05322409e+00 1.06306789e+00 1.55741785e-32]
```

```
[9]  # As we can see, only the 2 first components are useful
     pca.n_components = 2
     X_reduced = pca.fit_transform(X)
     X_reduced.shape
     X_reduced[:5,:]
```

```
array([[-1.66026689,  0.04946392],
       [-3.14055635,  1.05226249],
       [ 2.7612284 ,  0.33786   ],
       [ 0.65801188, -2.04934056],
       [-1.91551906,  1.04644725]])
```

** Ref) https://scikit-learn.org/stable/tutorial/statistical_inference/unsupervised_learning.html*

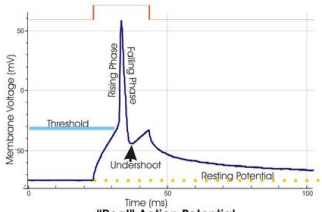
23

Optional :

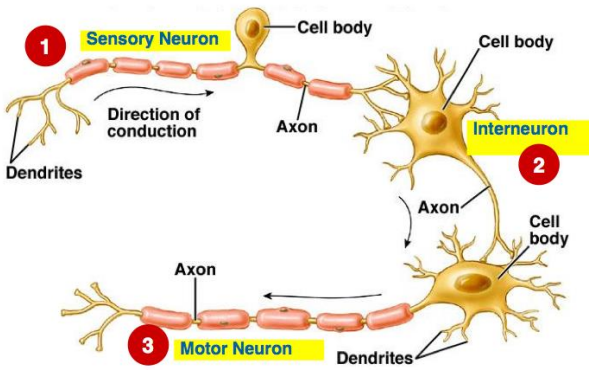
Introduction to Deep Learning & Artifical Neural Network

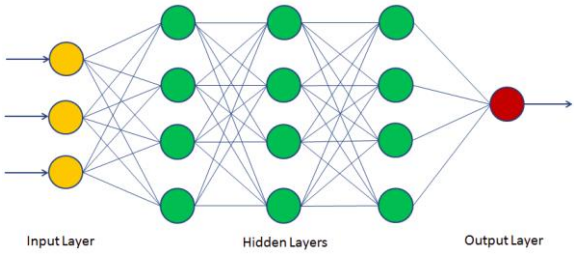
24

인공신경망(Artificial Neural Network)



"Real" Action Potential

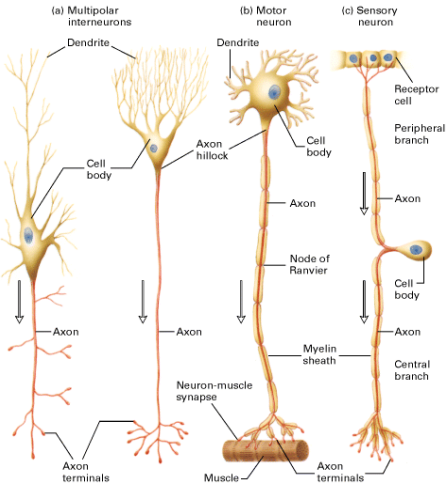


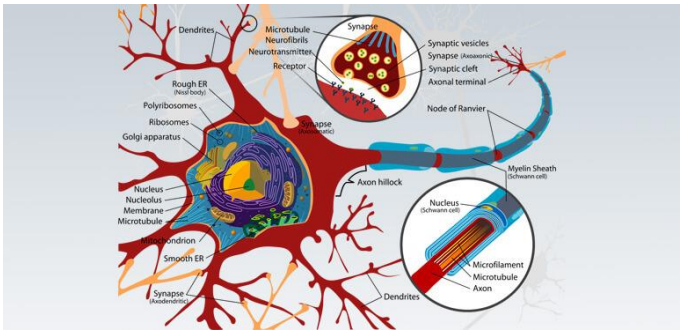


* Ref) https://www.humanbrainfacts.org/neurons-in-the-brain.php ;
https://id.m.wikipedia.org/wiki/Berkas:Action_potential_vert.png

25

참고: 신경의 종류와 뇌 신경



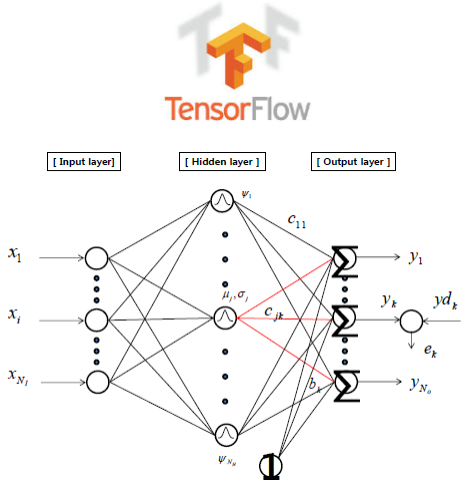


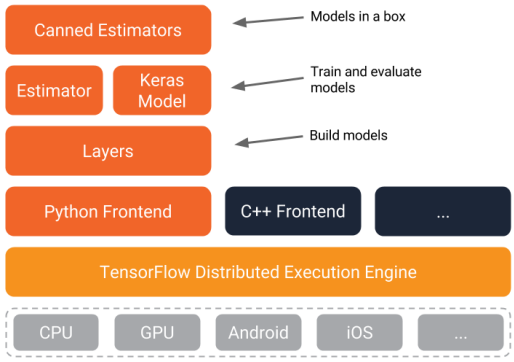
* Ref) https://wiki.tum.de/display/btt/Group+4%3A+Brain+Histology ; https://www.humanbrainfacts.org/neurons-in-the-brain.php

26

텐서플로우 (Tensor Flow)

- 구글이 만든 인공신경망 오픈소스 라이브러리 (무료공개)
- 수학 계산과 데이터의 흐름을 노드(Node)와 엣지(Edge)를 사용한 방향 그래프로 표현
- 텐서는 학습된 데이터가 저장되는 다차원 배열



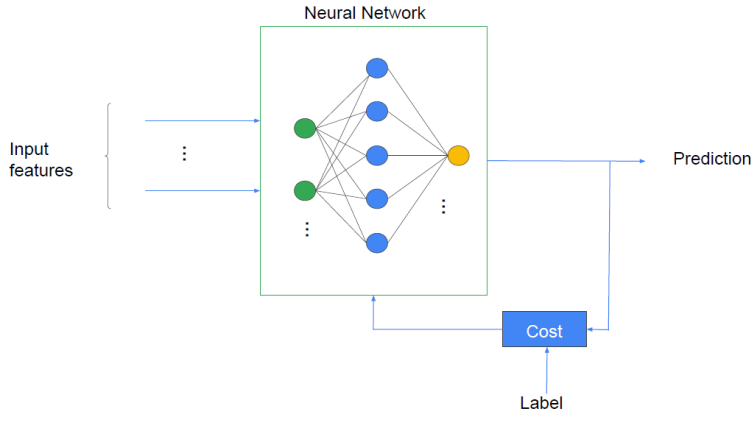


27

텐서플로우를 이용한 기계학습

- 텐서플로우 역시 알고리즘의 일종이며, 일반적 기계학습과 같은 학습 과정이 필요

Supervised machine learning requires features and labels

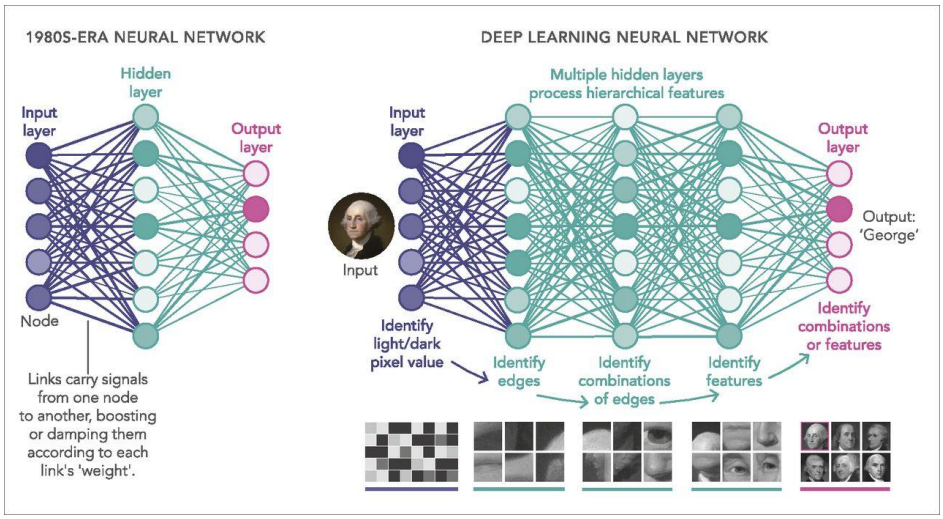


* Ref) https://commons.wikimedia.org/wiki/File%3ANeural_network.svg ; Google Coursera materials

28

딥러닝 (Deep Learning)

- Neural Network 의 Hidden Layers가 무수히 많은 것

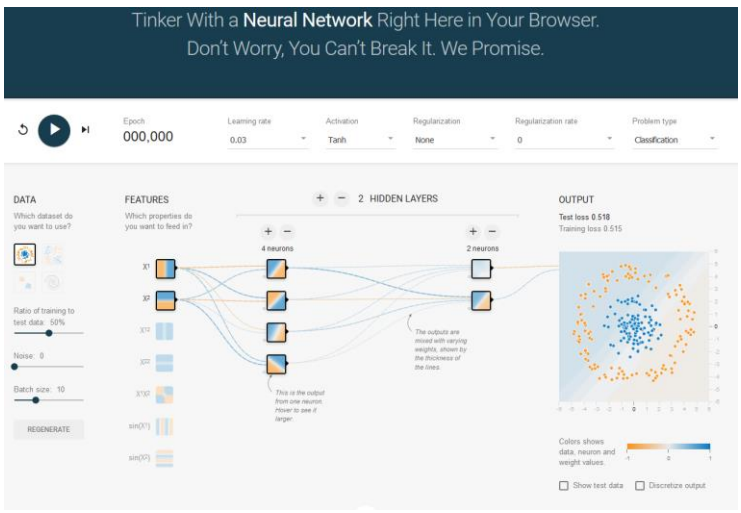


* Ref) https://www.pnas.org/content/116/4/1074

29

텐서플로우 실습

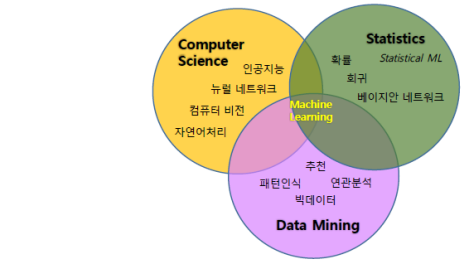
- https://playground.tensorflow.org/

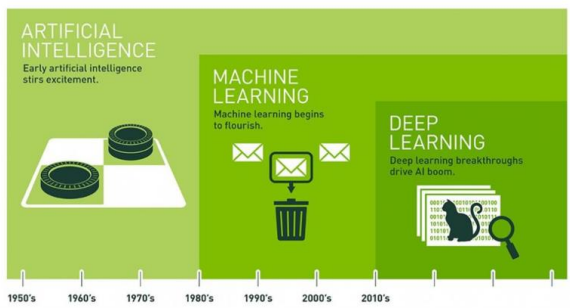


30

Wrap-Up

기계학습 (Machine Learning)



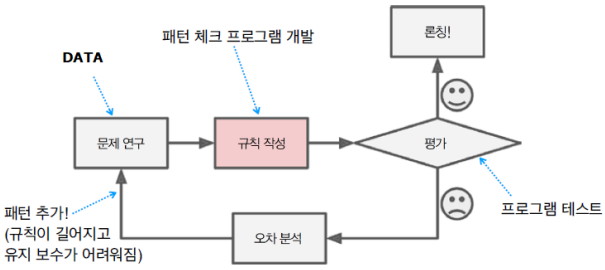


- **Machine Learning Application**

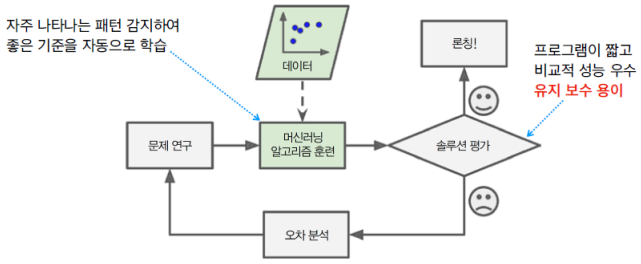
- 많은 수동 조정과 규칙이 필요한 문제
- 해결하기 너무 어렵거나 알려진 해가 없는 문제
  - 음성인식('one' vs 'two')
  - 얼굴인식(눈, 코, 입의 위치)
- 변화하는 환경에 적응해야 하는 문제
- 복잡한 문제와 대량의 데이터에서 통찰 얻기(데이터 마이닝)

전통적 개발 vs. 기계학습 기반 개발

- **Conventional**

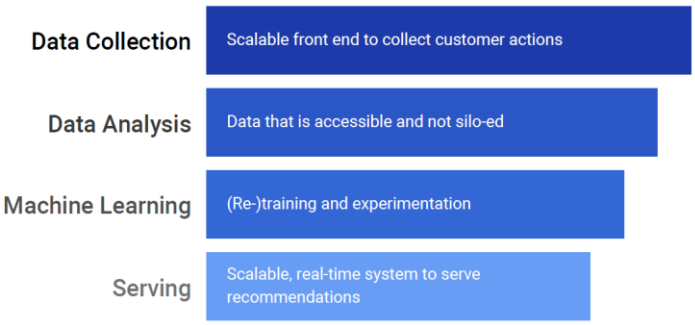


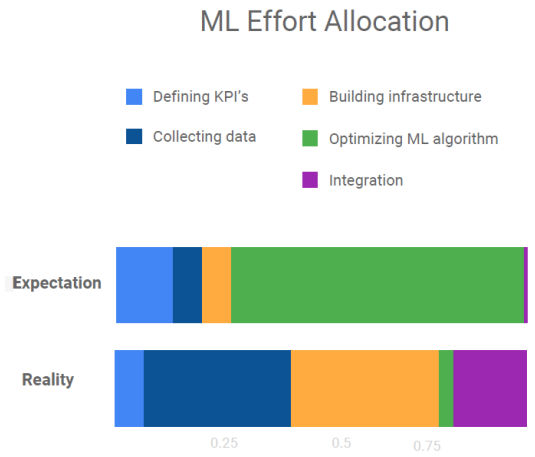
- **Machine Learning Based**



33

데이터 수집과 기계학습





34

기계학습의 분류

Learning Algorithm

- **Supervised Learning**
  - **Classification**
  - **Regression**
  - **Self-supervised Learning**
- **Unsupervised Learning**
  - **Clustering**
  - **Dimensionality Reduction**
- **Reinforcement Learning**

35

Let's Practice

36