Machine Learning

2018.10 김광삼
kwangsam.kim@axa.co.kr

1. Regression
2. Classification
3. Deep Learning

Anaconda 설치

- 설치
  파이썬 공식 홈페이지의 다운로드 페이지(http://www.python.org/downloads)
  아나코나 공식 홈페이지 (https://www.anaconda.com/download/)

  에디터 : pycharm(https://www.jetbrains.com/pycharm/)
  ,이클립스
  , Jupyter notebook
  , Spyder
  , SublimeText3

- 기본문법
  점프 투 파이썬 (https://wikidocs.net/book/1)

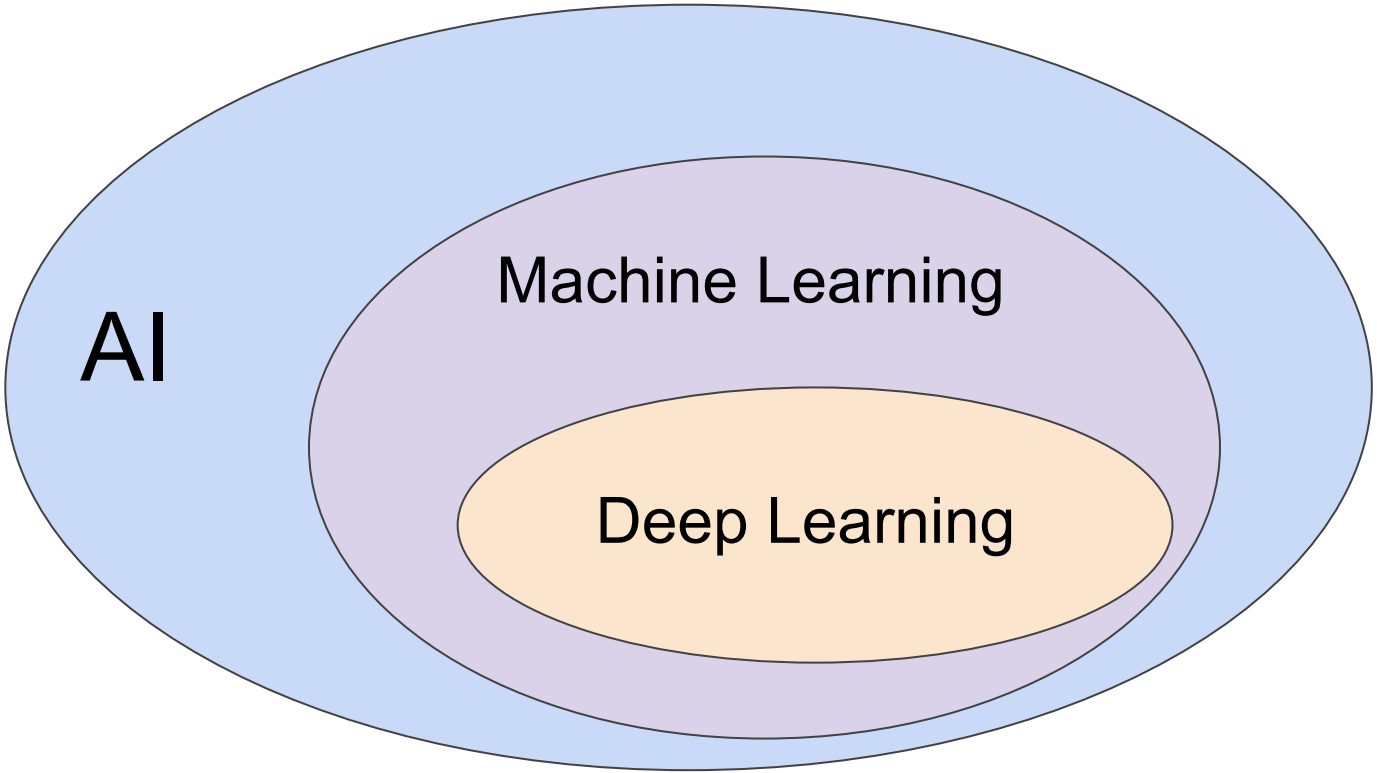
(참고) Jupyter notebook 자주사용하는 단축

command mode

- enter : edit mode로 바꿈. 선택된 cell로.
- a, b : 순서대로 위, 아래에 cell 추가
- y, m, r: 순서대로 cell 모드를 code, markdown, raw로 바꾸기
- j, k : 셀 이동. 아래, 위. vim 키랑 똑같다. shift 누른 후에 키를 누르면 여러 개 선택된다.
- x, c, shift+v, v : cut, copy, 위에 붙여넣기, 아래에 붙여넣기
- dd: 선택된 cell 삭제.
- z: 삭제 취소
- shift+M : 선택된 cell 합병. 하나만 선택되면 바로 아래꺼랑 합병
- shift+spacebar, spacebar : 스크롤 업, 다운

edit mode

- ESC: command 모드로 바꿈.
- shift+enter or ctrl+enter: 한 뭉치 실행
- ctrl + shift + - : 커서 위치 아래 부분을 split한다.



머신러닝 및 신경망 학습의 목적은 손실 함수의 값을 가능한 한 낮추는 매개변수를 찾는 것

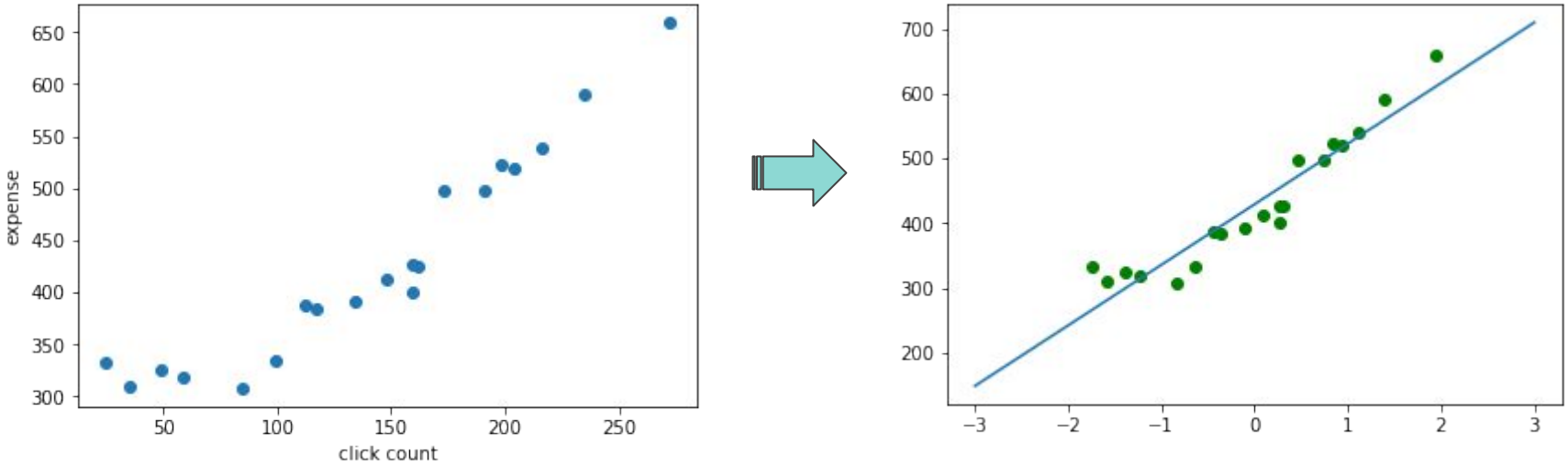
머신러닝 알고리즘

- 회귀(regression) : 연속적인 데이타 (몸무게 예측)
- 분류(Classification) : 입력값에 대한 label이 존재 (스펨메일 여부)
- 비지도학습 : 클러스트링(Clustering) , 강화 학습 ...

Regression

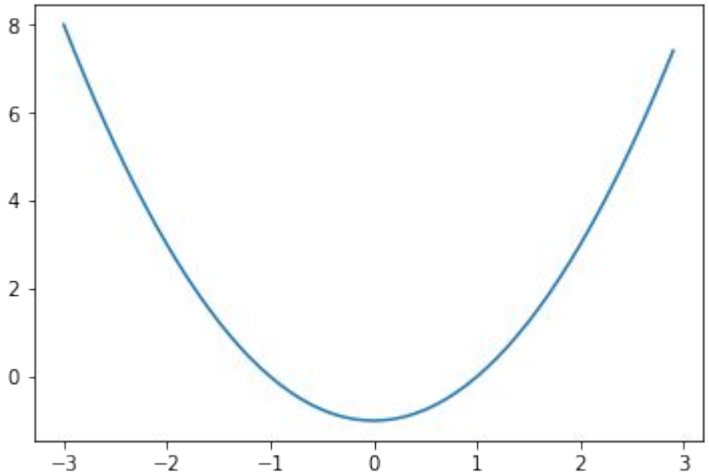
Regression(회귀)

➢ 회귀선의 사전적 의미 : 두 변수 x와 y와의 관계에 적합한 선. 회귀선이라고도 한다.



- 함수 h(x) = wx + b 에서 주어진 데이타 값 근접할수 있는 w , b 를 찾는 문제임

- 결국 주어진 데이타 y 값에 f(x) 의 결과값의 오차를 최소화 할 수 있는 매개변수를 찾는문제
- (y값 - 예측치)^2 의 값의 오차는 결국 2차 함수
- $\frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$: MSE(Mean Squared Error)



➢ 오차가 최소화 되는 지점의 기울기는 0에 가까워지므로 매개변수를 기울기가 작아 지는 지점으로 갱신하여 오차 범위를 줄여나간다.

즉 (y값 - 예측치)^2 미분값

- 각 변수에 대해 편미분 결과 매개변수 갱신식

$$b = 매개변수b - 학습율 * np.sum((h(x) - y))$$

$$w = 매개변수w - 학습율 * np.sum((h(x) - y)) * x$$

➢ **즉 머신러닝 및 신경망 학습의 목적은 손실 함수의 값을 가능한 한 낮추는 매개변수를 찾는 것.**

#Python 으로 구현 예제)

regression1.ipynb

➢ 입력 데이타가 여러개 존재할 경우

$$f(x) = w_1 x_1 + w_2 x_x + w_3 x_3 \cdots + b$$

➢ 위 식은 결국 아래의 표현식 처럼
가중치 와 입력값의 벡터 내적의 형태로 표현할 수 있다.

$$f(x) = W \bullet x$$

#Python 으로 구현 예제)

regression2.ipynb

Scikit-learn 라이브러리를 이용한 regression

모듈 : **sklearn.linear_model**

1. **LinearRegression**

2. **Lasso (L1 regularization)**
   - 영향이 적은 변수들을 **0**으로 보내서 없애고 영향력이 큰 변수들만 선택

3. **Ridge (L2 regularization)**
   - 변수 중에 서로 상관이 높아서 실제 사용가능한 변수가 적을 경우

4. **Elastic Net : L1,L2 의 혼합형**

#Python 으로 구현 예제) regression3.ipynb

예제) 보스턴 평균주택 가격 예측 문제 regression3.ipynb

data 는 13 개의 feature 로 이루어져 있고 label 은 주택가격의 중간값(MEDV)

데이터셋 구성

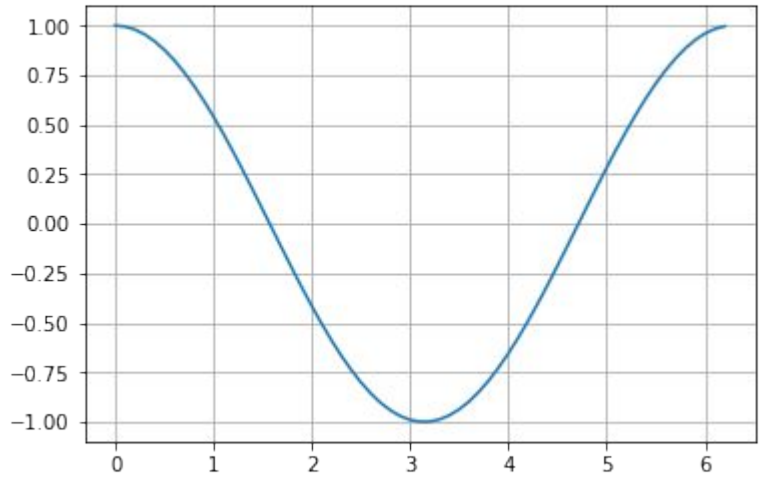
1. CRIM: 자치시(town) 별 1인당 범죄율
2. ZN: 25,000 평방피트를 초과하는 거주지역의 비율
3. INDUS:비소매상업지역이 점유하고 있는 토지의 비율
4. CHAS: 찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
5. NOX: 10ppm 당 농축 일산화질소
6. RM: 주택 1가구당 평균 방의 개수
7. AGE: 1940년 이전에 건축된 소유주택의 비율
8. DIS: 5개의 보스턴 직업센터까지의 접근성 지수
9. RAD: 방사형 도로까지의 접근성 지수
10. TAX: 10,000 달러 당 재산세율
11. PTRATIO: 자치시(town)별 학생/교사 비율
12. B: 1000(Bk-0.63)^2, 여기서 Bk는 자치시별 흑인의 비율을 말함.
13. LSTAT: 모집단의 하위계층의 비율(%)
14. MEDV: 본인 소유의 주택가격(중앙값) (단위: $1,000)

Classification

Classfication

벡터 내적의 정의

$$w \cdot x = |w| \cdot |x| \cdot \cos(\theta)$$



➢ $cos(\theta)$가 음수가 되는 구간은

90도 < θ < 270도 구간이므로

즉 weight 벡타의 반대의 영역이라고 판단할 수 있고 이를 이용하여 2진 분류가 가능하다.

#Python 으로 구현 예제)
classfication1.ipynb

Perceptron

AND gate
0 0 | 0
0 1 | 0
1 0 | 0
1 1 | 1

OR gate
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 1

XOR gate
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 0

```
def test(logic):
    for x1, x2 in [(0, 0), (0, 1), (1, 0), (1, 1)]:
        y = logic(x1, x2)
        print(x1, x2, '|', y)

def make_neuron(w, b):
    def neuron(*x):
        x = np.array(x)
        z = np.dot(x, w) + b
        y = 1 if z > 0 else 0
        return y
    return neuron

AND = make_neuron(w=np.array([0.5, 0.5]), b=-0.7)
NAND = make_neuron(w=np.array([-0.5, -0.5]), b=0.7)
OR = make_neuron(w=np.array([0.5, 0.5]), b=-0.2)
```

그럼 **XOR** 문제는?

다층퍼셉트론(MLP)

AND
0 0 | 0
0 1 | 0
1 0 | 0
1 1 | 1

OR
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 1

NAND
0 0 | 1
0 1 | 1
1 0 | 1
1 1 | 0

XOR
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 0

- **그럼 XOR 문제는?**

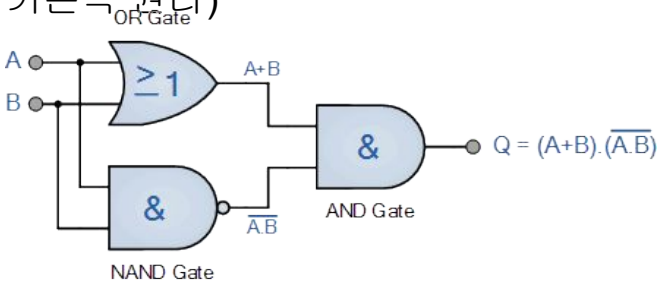
NAND 게이트결과와 OR케이트 결과를 AND케이트 입력으로 전달
(신경망(neural nework) 의 기본적 원리)

```
def XOR(x1, x2):
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    y = AND(s1, s2)
    return y
```

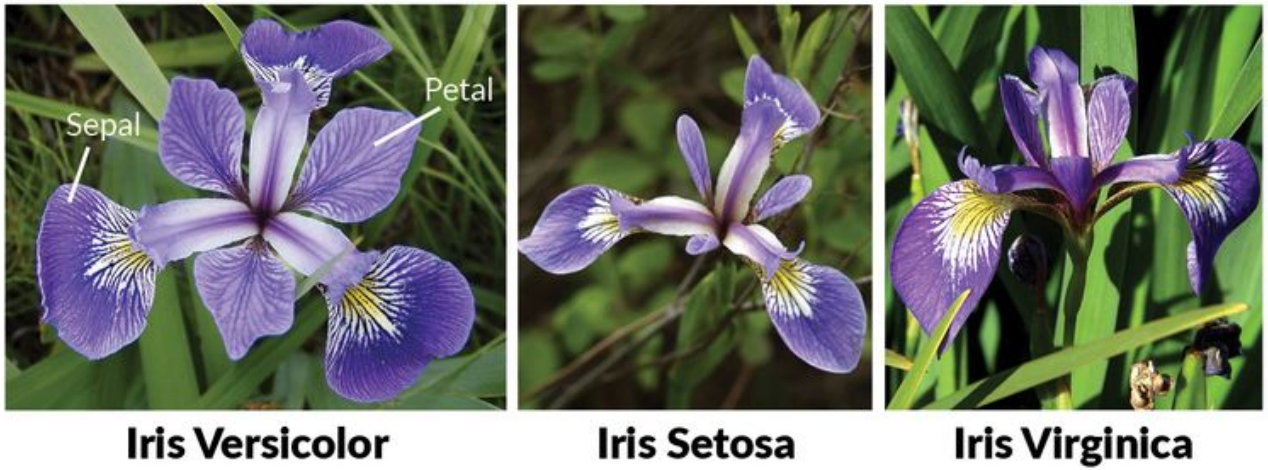
---

test(XOR) 결과는

0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 0

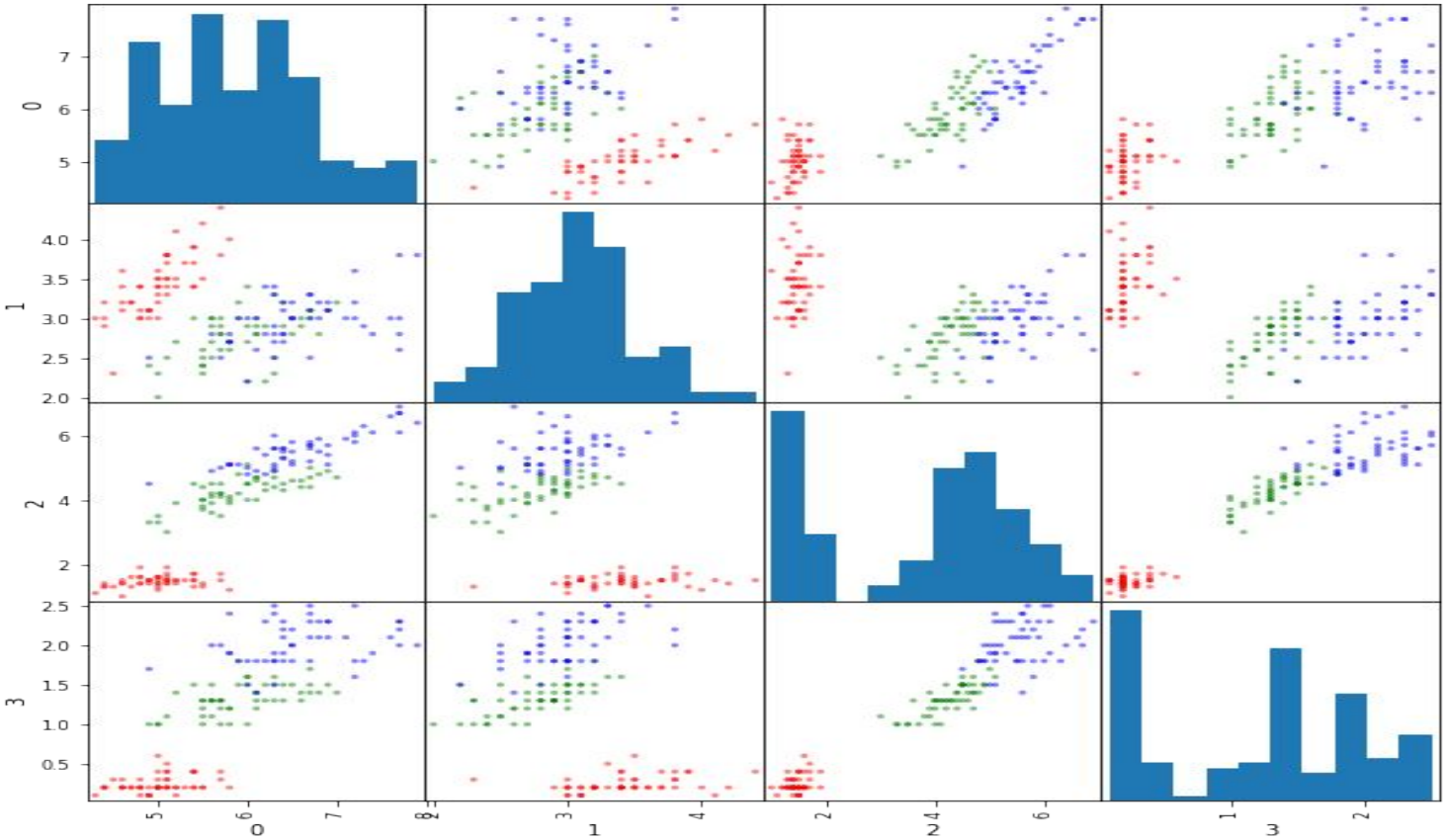


➢ http://archive.ics.uci.edu/ml/datasets.html (Iris(붓꽃) 데이타 사용)



- 데이타 형태
  - 총 150개의 꽃받침(sepal)의 길이,폭 과 꽃잎(petal)의 길이,폭에 대한 종 데이터
  - 품종 : Setosa, versicolor, virginica

➢ Iris(붓꽃) 데이타 분포



➢ Perceptron 를 이용한 iris 데이타 분류 (classfication1.ipynb 참고)

```
import numpy as np

class Perceptron:
    def __init__(self, 학습횟수=10, 학습률=0.1):
        self.epochs = 학습횟수
        self.learning_rate = 학습률

    def activation(self, z):
        return np.where(z > 0, 1, -1)

    def predict(self, x):
        z = np.dot(x, self.w) + self.b
        return self.activation(z)
```

```
    def fit(self, X, y):
        self.w = np.zeros(X.shape[1])
        self.b = 0.

        error_history = []
        for i in range(self.epochs):
            # 각 샘플에 대해
            sum_square_error = 0
            for xi, yi in zip(X, y):
                y_pred = self.predict(xi)
                error = yi - y_pred
                sum_square_error += error**2
                # 가중치 갱신
                update = self.learning_rate * error
                self.w += update * xi
                self.b += update
```

Scikit-learn 라이브러리를 이용한 classfication

예제): classfication2.ipynb

1. LogisticRegression
   - Sigmoid function 를 사용한 가중치 update

2. SVM (Support Vector Machine)
   - Margin을 최대가 되도록 하는 알고리즘 (선형 kernel='linear' , 비선형 : kernel='rbf')

3. SGDClassifier
   - SGD 알고리즘을 적용한 분류(데이타가 대용량이면 효율적일 수도 있음)

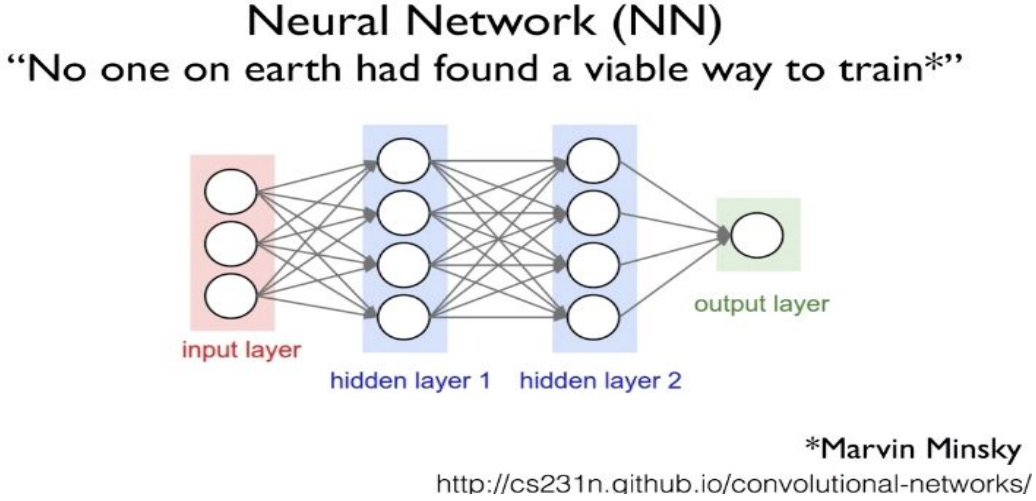
4. 의사 결정 트리(DecisionTreeClassifier)

5. KNN(k-Nearest Neighbors) : 최근접 인근 알고리즘

Deep Learning

Neural Network (신경망)

1. ANN(Artificial neural network) : 입력계층, 은닉계층, 출력 계층으로 구성되어 있고, 초기에는 기술적인 문제로 총3개의 계층으로 구성
2. DNN(Deep neural network) 은닉 계층을 여러 개 쌓아서 만든 인공신경망
3. CNN(Convolutional neural network) : 영상처리에 많이 활용되는 신경망 기술
4. RNN. AE. GAN. UNET .....

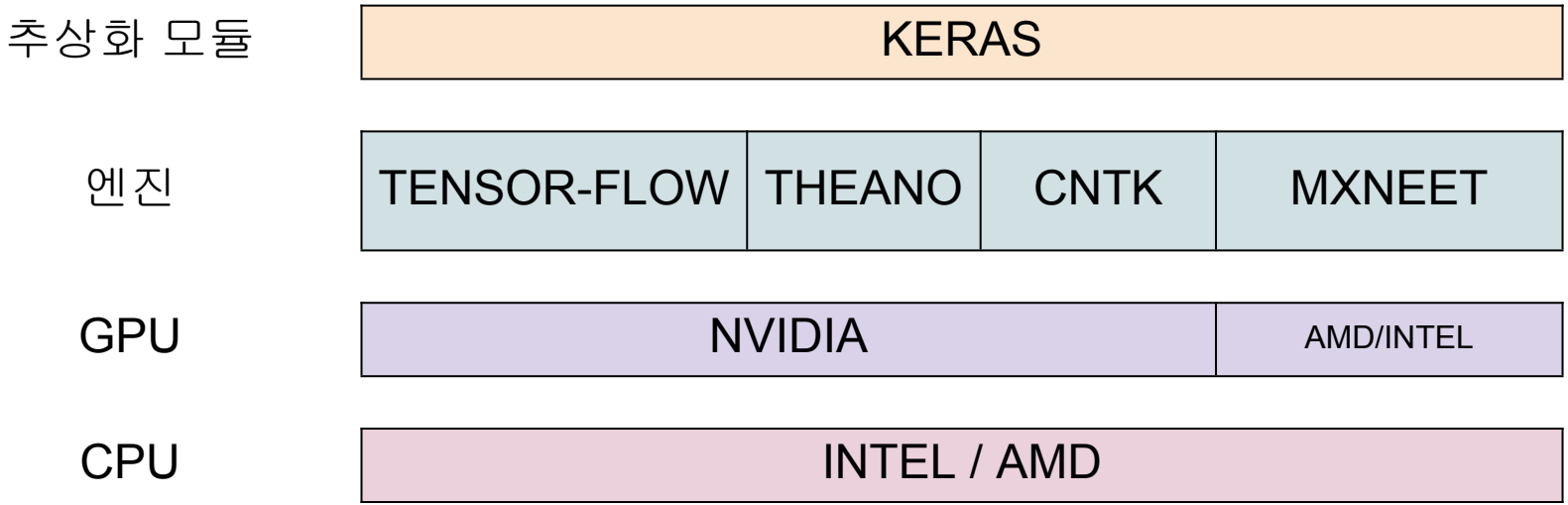


```
class Layer:
    def __init__(self, input_size, output_size, activation):
        self.weights = np.random.randn(input_size, output_size)
        self.bias = np.random.randn(output_size)
        self.activation = activation

    def output(self, x):
        z = np.dot(x, self.weights) + self.bias
        return self.activation(z)

class FeedForwardNet:
    def __init__(self):
        self.layers = []

    def add_layer(self, layer):
        self.layers.append(layer)

    def predict(self, X):
        layer_input = X
        for layer in self.layers:
            layer_input = layer.output(layer_input)
        return layer_input
```

```
neuralnet = FeedForwardNet()

neuralnet.add_layer(Layer(13, 10,
activation=sigmoid))

neuralnet.add_layer(Layer(10, 20,
activation=sigmoid))

neuralnet.add_layer(Layer(20, 3,
activation=softmax))
```

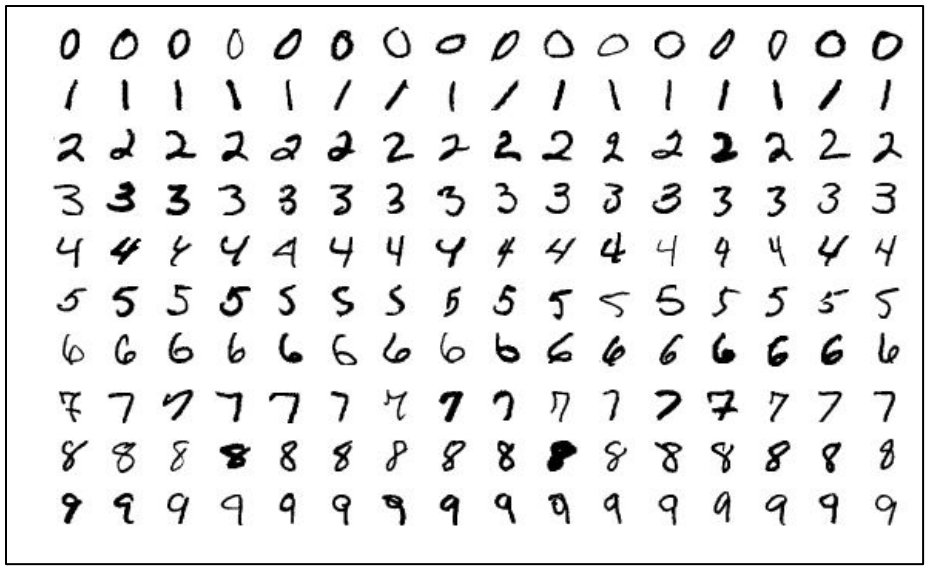
➢ 출력층의 데이타가 다음 layer의 입력층으로 들어간다.

Deep Learning library



학습데이타 설명

- MNIST 데이타
  - 손으로 쓴 숫자들로 이루어진 대형 데이타이며 28x28 픽셀의 흑백이미지로서 6만장의 트레이닝 이미지와 만장의 테스트 이미지를 포함하고 있다.



학습데이타 설명

- Cifar10
  - 10가지 사물이 담긴 컬러 이미지로 32 x 32 픽셀로 총 6만장이며 5 만장은 학습용이고 1만장은 평가용이다.

예제) DNN_iris.ipynb (iris데이타), DNN_mnist.ipynb, DNN_CIFAR.ipynb

Keras 구현

- 설치 : conda install keras (또는 keras-gpu)
- Keras.layers
  - 각 계층을 만드는 모듈

- Keras.models
  - 각 layer 들을 연결하여 신경망 모델을 생성 후, 컴파일하고, 학습시키는 역할
  - models.Model 객체에는 compile, fit, predict, evaluate 함수를 제공

간단 예제)

```
model = keras.models.Sequential()
model.add(keras.layers.Dense(1, input_shape=(1,)))
model.compile('SGD', 'mse')
model.fit(x[:2], y[:2], epochs=1000, verbose=0)
```

ex)ANN_regression.ipynb, ANN_Classification.ipynb, poly_Dnn_test.ipynb

DNN (Deep Neural network)

- DNN은 은닉계층이 여러개인 신경망으로 DNN은 수십에서 수백의 hidden layer 으로 구성되기도 한다.
- DNN은 복잡한 데이타를 학습하기 위해 hidden layer 를 늘린 방식이다. (과적합에 대해 고민하여야함)
- DNN을 구성하는 가중치의 학습은 예측값의 목표값에 대한 오차를 역방향으로 되돌리면서 계산하는데 이를 Back propagation(오차역전파) 라고 한다.
- DNN에서는 vanishing gradient problem(경사도 소실 문제) 때문에 activation function으로 sigmoid함수 보다는 relu 함수를 사용함.

Cross Entropy Error (CEE) $-\sum_{i=1}^{n} y_i log(y_i')$

첫번째 넷이 주는 계산 결과:

계산결과	라벨(A/B/C)	correct?
0.3 0.3 0.4	0 0 1 (A)	yes
0.3 0.4 0.3	0 1 0 (B)	yes
0.1 0.2 0.7	1 0 0 (C)	no

두번째 넷이 주는 계산 결과:

계산결과	라벨(A/B/C)	correct?
0.1 0.2 0.7	0 0 1 (A)	yes
0.1 0.7 0.2	0 1 0 (B)	yes
0.3 0.4 0.3	1 0 0 (C)	no

첫번째, 두번째 계산 결과는 정답률이 66% 이다. 그러나
2번째 결과가 첫번째 보다 좀 더 정확하다는것을 알 수 있다.

이와 비교하여 CrossEntropy로 오차를 계산하면
-((ln(0.3)*0) + (ln(0.3)*0) + (ln(0.4)*1)) = -ln(0.4)

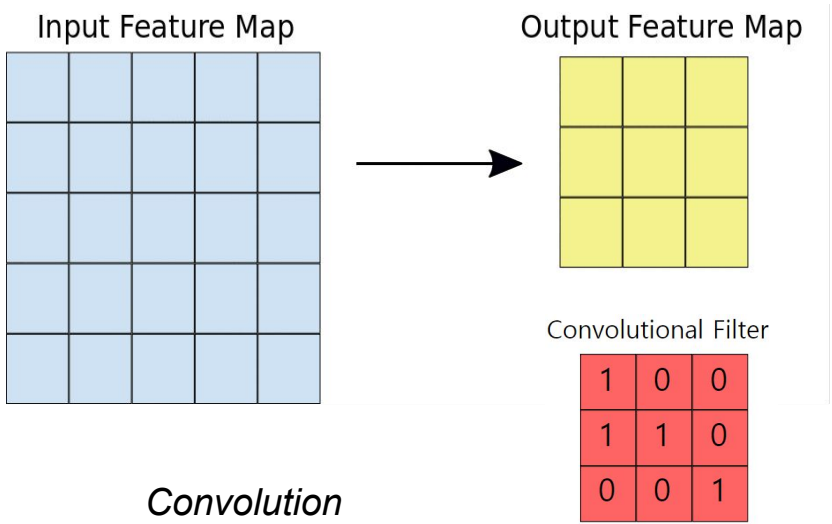
나머지 두 샘플 모두에 대해 계산하고 평균하면
-(ln(0.4) + ln(0.4) + ln(0.1)) / 3 = 1.38

두번째 넷에 대해 평균 cross entropy를 계산하면
-(ln(0.7) + ln(0.7) + ln(0.3)) / 3 = 0.64

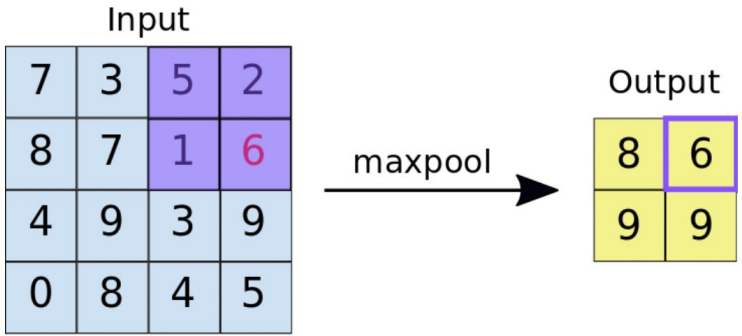
결과를 비교해 보면 두번째 넷이 오차가 더 작음을 알 수 있다

CNN(Convolution Neural network)

CNN은 convolution filter(합성곱 필터)를 이용하여 동작한다. DNN은 전계층을 1차원 방식으로 신호를 처리하기 때문에 2차원 특성을 처리하기에는 한계가 있다. 반면 CNN은 2차원 합성곱으로 각 노드를 처리하기 때문에 이미 처리에 적합하다.



Convolution



Pooling

CNN(Convolution Neural network)

추가적으로 사용되는 객체

- Conv2D : 2차원 합성곱을 계산하는 클래스
- MaxPooling2D : 2차원 max pooling 을 계산하는 클래스
- Flattern : 다차원 입력을 1차원으로 변환하는 클래스

참고) mnist 이미지는 흑백이미지로 컬러 RGB 에 대한 채널 정보가 존재하지 않는다.
그래서 흑백이미지는 입력데이타의 차원을 하나 더 추가하여야 한다.
(해당 입력데이타가 2차원 앞 또는 뒤 존재 여부는 설정파일에 존재)
Keras 설정 정보에 존재 : "image_data_format": "channels_last"

예제) CNN_mnist.ipynb, CNN_CIFAR10

이런 모습?