Chapter 9

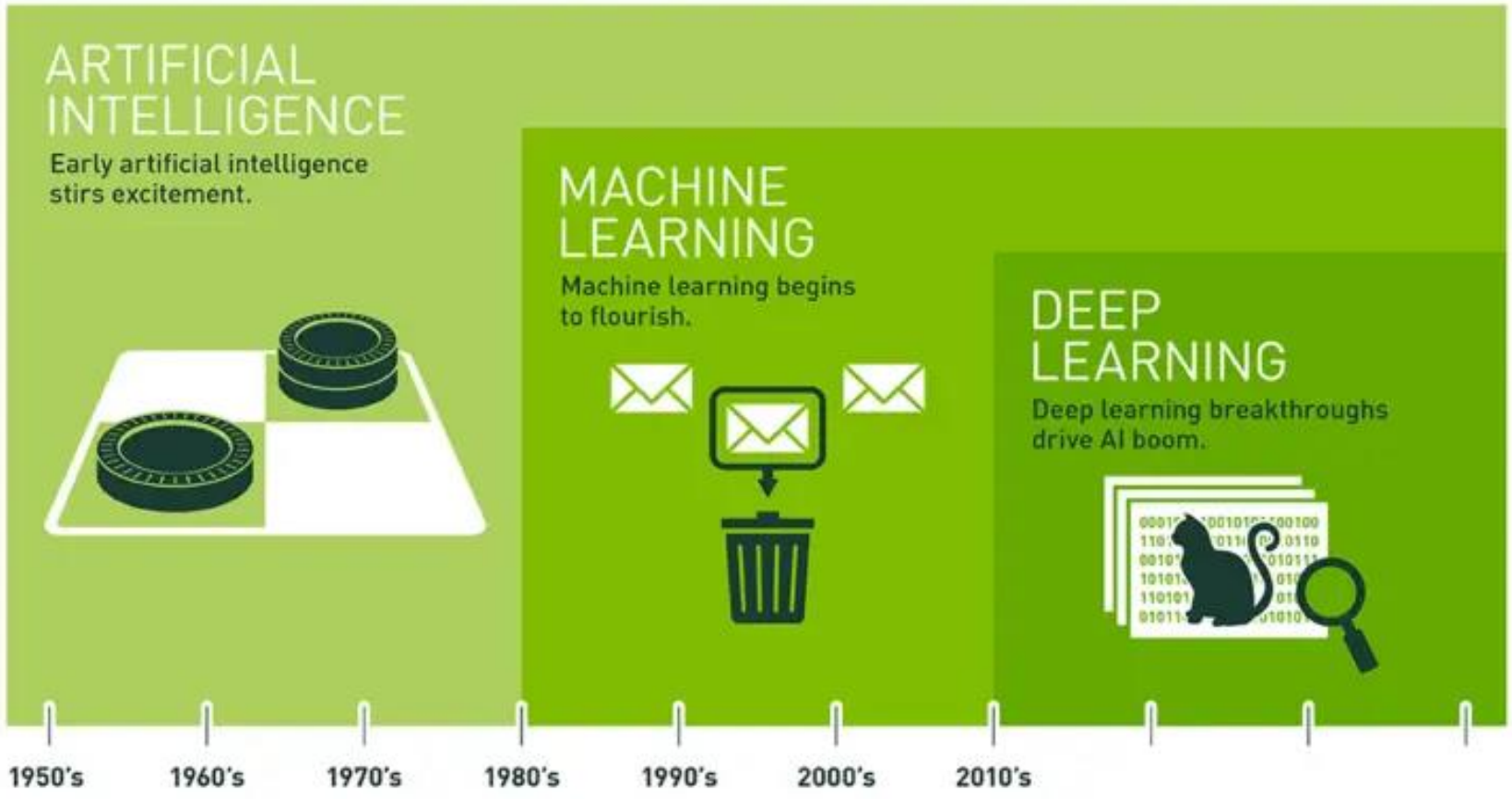
딥러닝을 이용한 주가 예측

인공지능이란

- 인공적으로 만들어진 지적 능력을 나타내는 말로 생각하는 기계를 말한다.

- 컴퓨터 공학에서 인공 지능(AI)은 학습, 문제 해결, 패턴 인식 등과 같이 주로 인간 지능과 연결된 인지 문제를 해결을 이야기한다.

인공지능 기술의 분류

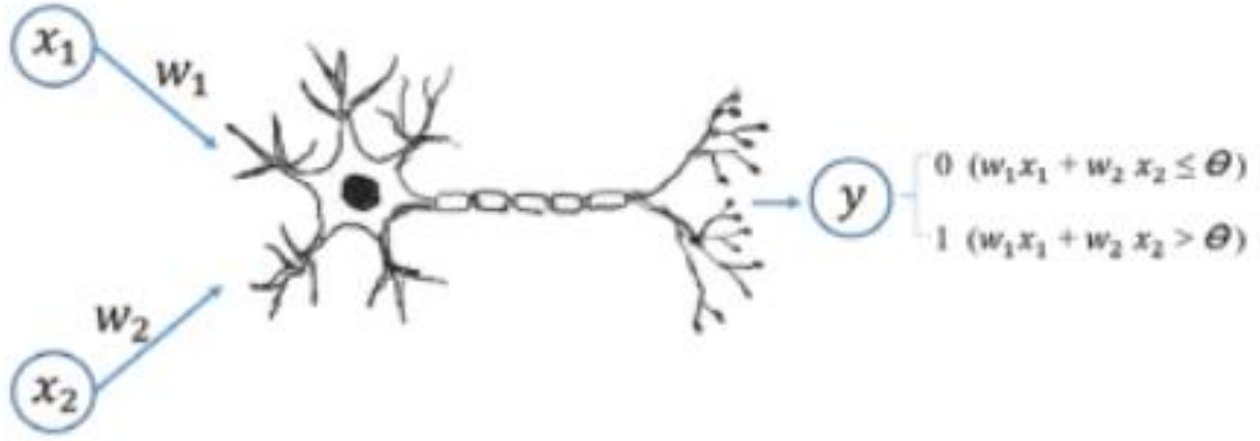


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

퍼셉트론

- 신경 세포를 인공적으로 모델링 한것 으로 받은 신호에 가중치를 더해 다음으로 전달해주는 과정을 이야기 한다



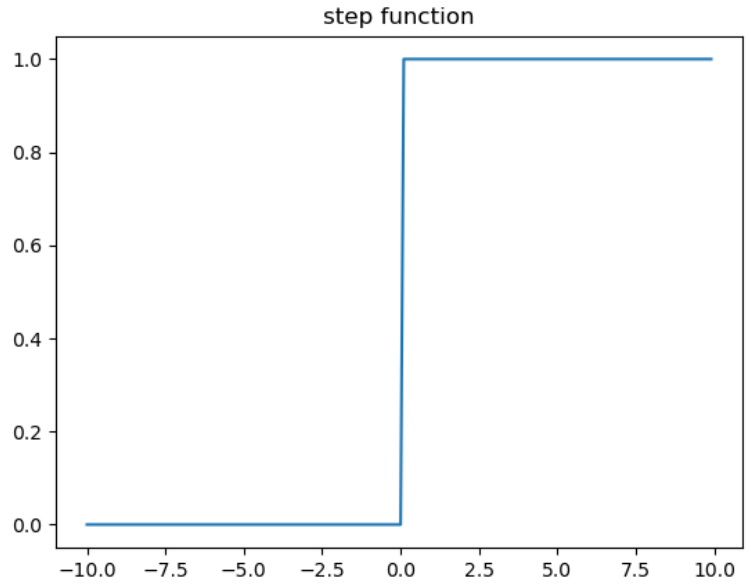

```python
def AND(x1,x2):
    w1 = 0.5
    w2 = 0.5
    theta = 0.7
    if w1 * x1 + w2 * x2 > theta:
        return 1
    else:
        return 0

def NAND(x1,x2):
    w1 = -0.5
    w2 = -0.5
    theta = -0.7
    if w1 * x1 + w2 * x2 > theta:
        return 1
    else:
        return 0

def OR(x1,x2):
    w1 = 0.5
    w2 = 0.5
    theta = 0.2
    if w1 * x1 + w2 * x2 > theta:
        return 1
    else:
        return 0
```

활성화 함수

입력을 변화 하는 함수로 선형 레이어를 아무리 쌓아도 한 층의 선형 레이어로 변환 되는 문제와 비선형으로 풀리는 문제를 해결해 준다.

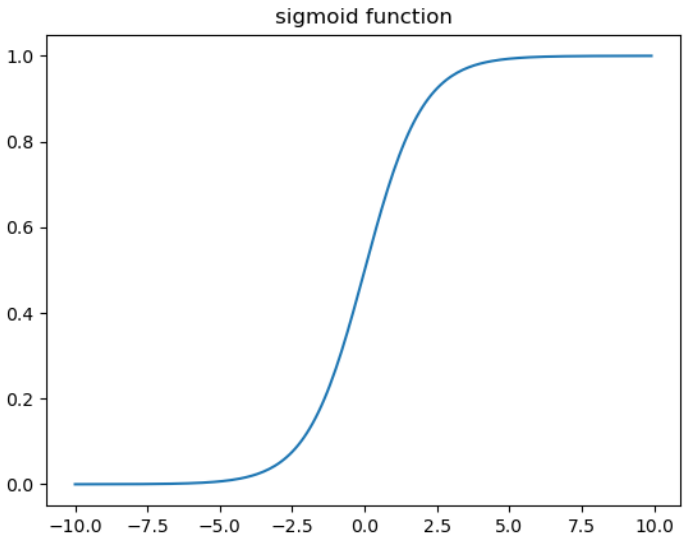
계단함수

```python
import numpy as np
import matplotlib.pyplot as plt

def stepfunc(x):
    return np.where(x <= 0, 0, 1)

x = np.arange(-10, 10, 0.1)
y = stepfunc(x)

plt.plot(x, y)
plt.title('step function')
plt.show()
```



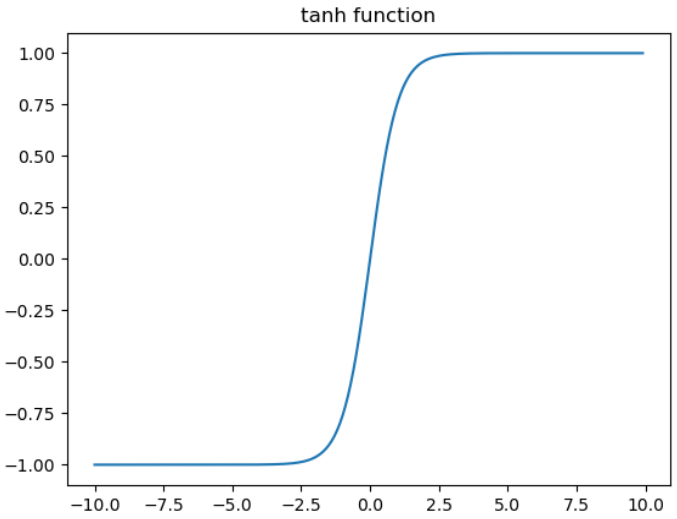
시그모이드 함수

```python
import numpy as np
import matplotlib.pyplot as plt

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

x = np.arange(-10, 10, 0.1)
y = sigmoid(x)

plt.plot(x, y)
plt.title('sigmoid function')
plt.show()
```



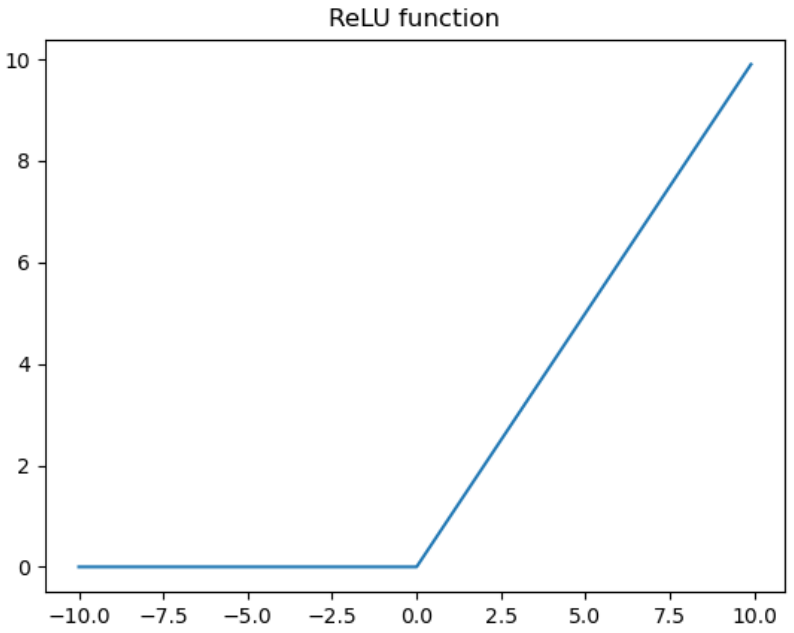
Tanh 함수

```python
import numpy as np
import matplotlib.pyplot as plt

def tanh(x):
    return (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))

x = np.arange(-10, 10, 0.1)
y = tanh(x)

plt.plot(x, y)
plt.title('tanh function')
plt.show()
```



ReLU 함수

```python
import numpy as np
import matplotlib.pyplot as plt

def relu(x):
    return np.maximum(0, x)

x = np.arange(-10, 10, 0.1)
y = relu(x)

plt.plot(x, y)
plt.title('ReLU function')
plt.show()
```



소프트맥스 함수

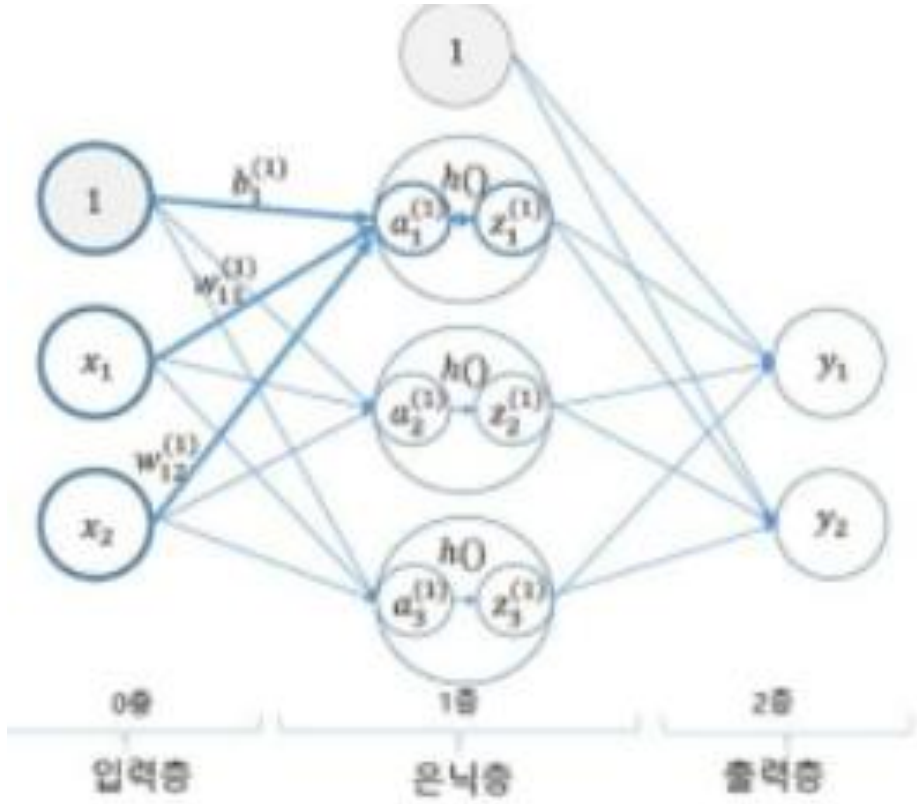
```python
import numpy as np
import matplotlib.pyplot as plt

def softmax(x):
    return np.exp(x) / np.sum(np.exp(x))

print(softmax([2,3,5]))
```

```
C:\Users\김종하\PycharmProjects\StockA
[0.04201007 0.1141952  0.84379473]

Process finished with exit code 0
```

다층 퍼셉트론

퍼셉트론을 여러 층으로 구성한 것을 말한다.
비 선형 문제를 해결할 수 있다.

Numpy를 이용해 입력층에서 출력층까지 신호가 전달되는 과정을 볼 수 있다.




```python
import numpy as np
X = np.array([10, 20])  # ①
W1 = np.array([[0.1, 0.3, 0.5], [0.2, 0.4, 0.6]])
B1 = np.array([1, 2, 3])  # ③

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

A1 = np.dot(X, W1) + B1
Z1 = sigmoid(A1)

print('A1 :', A1)
print('Z1 :', Z1)
```

```
C:\Users\김종하\PycharmProjects\StockAnaly
A1 : [ 6. 13. 20.]
Z1 : [0.99752738 0.99999774 1.        ]

Process finished with exit code 0
```

텐서플로

구글 브레인팀에서 심층 신경망 연구를 위해 개발한 머신러닝 라이브러리이다.

pip install tensorflow 를 통해 설치할 수 있다.

Tensor 는 수학적인 개념으로 데이터의 배열이라 한다.
N차원의 배열이나 리스트로 표현할 수 있다.
차원, 형태, 자료형을 지닌다.

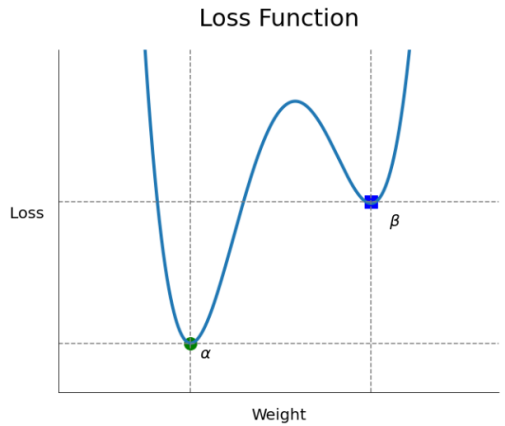
```
1    import tensorflow as tf
2    hello = tf.constant("hi, tensorflow")
3    print(hello)
```

```
C:\Users\김종하\PycharmProjects\StockAnalysisInPython_IES
2022-08-18 02:32:40.483711: W tensorflow/stream_executor
2022-08-18 02:32:40.484701: I tensorflow/stream_executor
2022-08-18 02:32:58.977023: W tensorflow/stream_executor
2022-08-18 02:32:58.977292: W tensorflow/stream_executor
2022-08-18 02:32:59.006769: I tensorflow/stream_executor
2022-08-18 02:32:59.007467: I tensorflow/stream_executor
2022-08-18 02:32:59.011258: I tensorflow/core/platform/c
To enable them in other operations, rebuild TensorFlow w
tf.Tensor(b'hi, tensorflow', shape=(), dtype=string)

Process finished with exit code 0
```

선형 모델

주어진 데이터를 가장 잘 나타내는 y = ax + b 꼴을 찾는 모

경사 하강 알고리즘

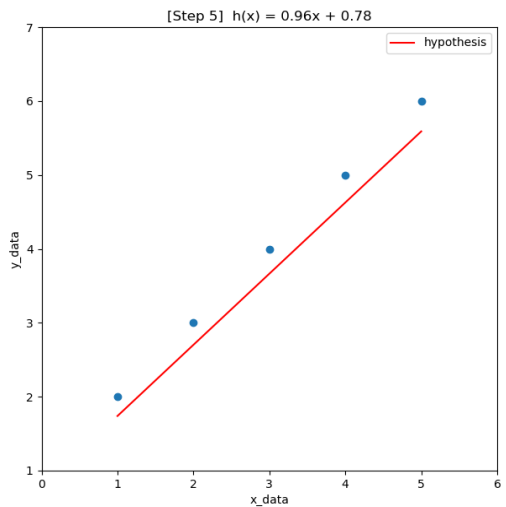


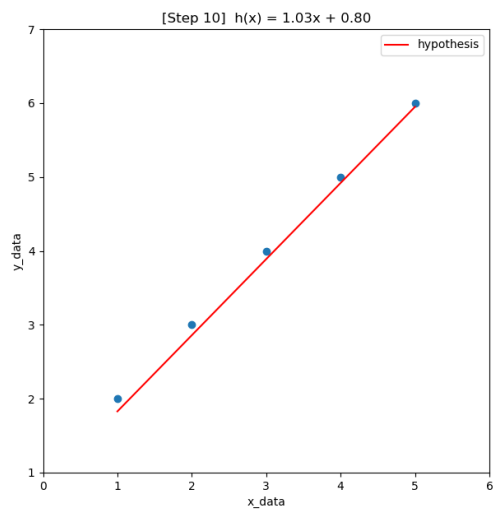
예측값과 실제 값의 차이를 제곱해서 평균낸 오차 제곱 평균을 가지고 비용이 적게 나오는 쪽으로 경사를 타고 내려가는 알고리즘이다.

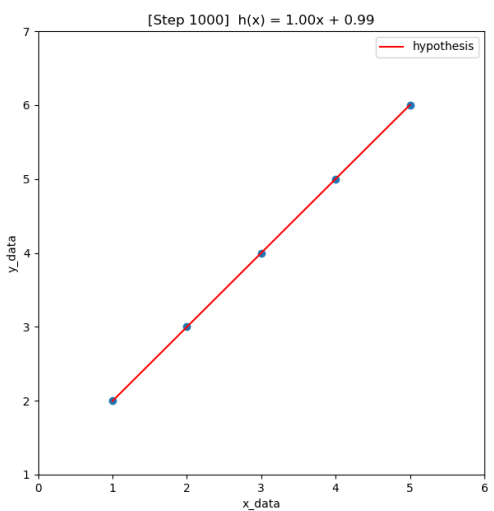
단점으로는 지역적인 최저점이 전체의 최대가 아닐 수 있다는 것과 계산 시간이 길다는 점이다.

```python
import matplotlib.pylab as plt
import tensorflow as tf

x_data = [1, 2, 3, 4, 5]
y_data = [2, 3, 4, 5, 6]

w = tf.Variable(0.7)
b = tf.Variable(0.7)
learn_rate = 0.01

print(f'step|    w|    b| cost')
print(f'----|-----|-----|-----')
```

```python
for i in range(1, 1101):
    with tf.GradientTape() as tape:
        hypothesis = w * x_data + b
        cost = tf.reduce_mean((hypothesis - y_data)**2)  # tf.losses.mean_squared_error(y, y_hat)
    dw, db = tape.gradient(cost, [w, b])
    w.assign_sub(learn_rate * dw)  # a = a - b
    b.assign_sub(learn_rate * db)

    if i in [1, 3, 5, 10, 1000, 1100]:
        print(f"{i:4d}| {w.numpy():.2f}| {b.numpy():.2f}| {cost:.2f}")
        plt.figure(figsize=(7, 7))
        plt.title(f'[Step {i:d}]  h(x) = { w.numpy():.2f}x + {b.numpy():.2f}')
        plt.plot(x_data, y_data, 'o')  # ⑥
        plt.plot(x_data, w * x_data + b, 'r', label='hypothesis')  # ⑦
        plt.xlabel('x_data')
        plt.ylabel('y_data')
        plt.xlim(0, 6)
        plt.ylim(1, 7)
        plt.legend(loc='best')
        plt.show()
```







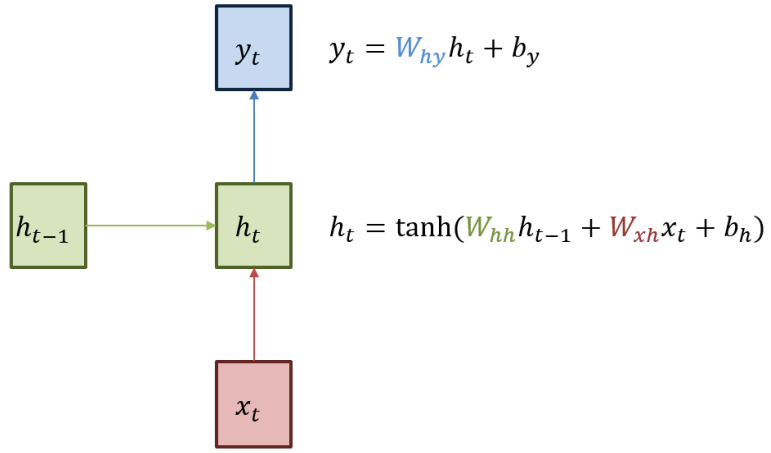
주가 예측 모델

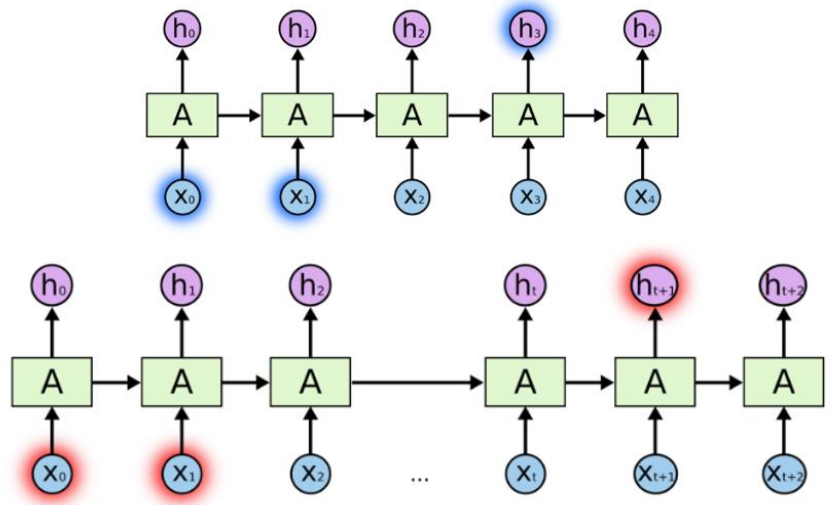
딥러닝에는 다양한 학습 방법이 있다.

그 중 음성, 문자 같은 순차적인 데이터 처리에 적합한 RNN

RNN에서 관련 정보와의 거리문제를 해결한 LSTM을 살펴보았다.

이는 이전 주가에 영향을 받는 주식에서도 적용 가능하다.





<관련 정보와 그 정보를 사용하는 지점 사이 거리가 멀 경우 RNN 학습능력 저하>

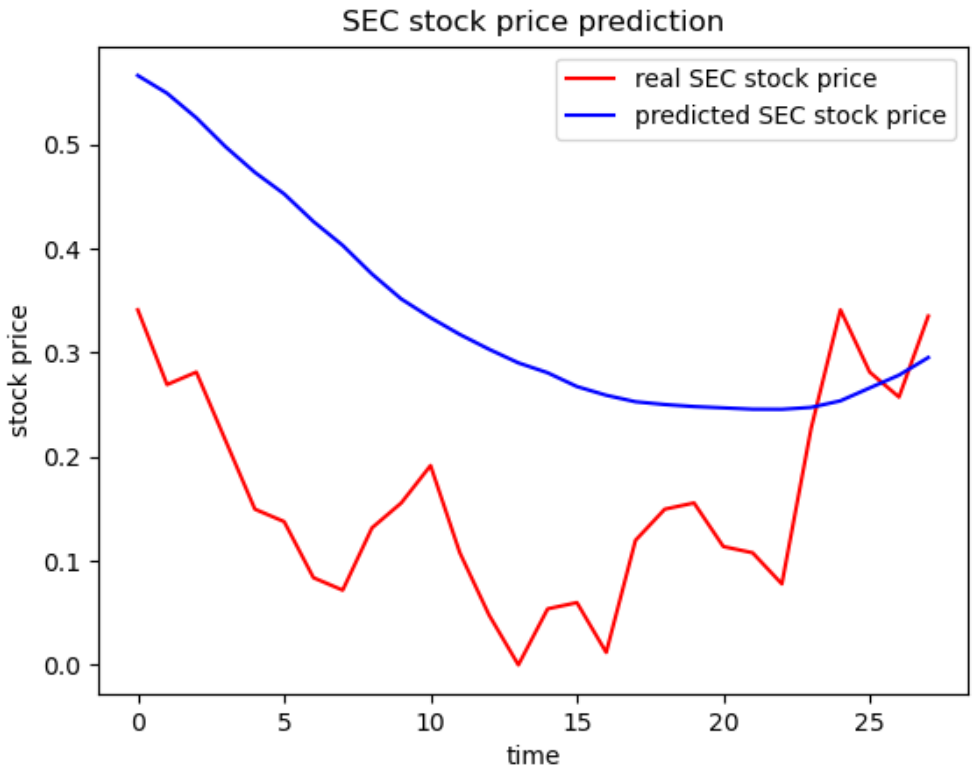
코드

```python
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
import numpy as np
import matplotlib.pyplot as plt
from Investar import Analyzer

mk = Analyzer.MarketDB()
raw_df = mk.get_daily_price('005930', '2018-05-10')

window_size = 10
data_size = 5

def MinMaxScaler(data):
    """최솟값과 최댓값을 이용하여 0 ~ 1 값으로 변환"""
    numerator = data - np.min(data, 0)
    denominator = np.max(data, 0) - np.min(data, 0)
    # 0으로 나누기 에러가 발생하지 않도록 매우 작은 값(1e-7)을 더해서 나눔
    return numerator / (denominator + 1e-7)

dfx = raw_df[['open','high','low','volume', 'close']]
dfx = MinMaxScaler(dfx)
dfy = dfx[['close']]

x = dfx.values.tolist()
y = dfy.values.tolist()

data_x = []
data_y = []
```

```python
for i in range(len(y) - window_size):
    _x = x[i : i + window_size]  # 다음 날 종가(i+windows_size)는 포함되지 않음
    _y = y[i + window_size]      # 다음 날 종가
    data_x.append(_x)
    data_y.append(_y)
    print(_x, "->", _y)

train_size = int(len(data_y) * 0.7)
train_x = np.array(data_x[0 : train_size])
train_y = np.array(data_y[0 : train_size])

test_size = len(data_y) - train_size
test_x = np.array(data_x[train_size : len(data_x)])
test_y = np.array(data_y[train_size : len(data_y)])

# 모델 생성
model = Sequential()
model.add(LSTM(units=10, activation='relu', return_sequences=True, input_shape=(window_size, data_size)))
model.add(Dropout(0.1))
model.add(LSTM(units=10, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(units=1))
model.summary()

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(train_x, train_y, epochs=60, batch_size=30)
pred_y = model.predict(test_x)
```

```python
# Visualising the results
plt.figure()
plt.plot(test_y, color='red', label='real SEC stock price')
plt.plot(pred_y, color='blue', label='predicted SEC stock price')
plt.title('SEC stock price prediction')
plt.xlabel('time')
plt.ylabel('stock price')
plt.legend()
plt.show()

# raw_df.close[-1] : dfy.close[-1] = x : pred_y[-1]
print("Tomorrow's SEC price :", raw_df.close[-1] * pred_y[-1] / dfy.close[-1], 'KRW')
```

주가 예측 코드 실행




```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10, 10)            640

 dropout (Dropout)           (None, 10, 10)            0

 lstm_1 (LSTM)               (None, 10)                840

 dropout_1 (Dropout)         (None, 10)                0

 dense (Dense)               (None, 1)                 11

=================================================================
Total params: 1,491
Trainable params: 1,491
Non-trainable params: 0
_________________________________________________________________
Epoch 1/60
3/3 [==============================] - 2s 7ms/step - loss: 0.2404
Epoch 2/60
3/3 [==============================] - 0s 6ms/step - loss: 0.2161
Epoch 3/60
3/3 [==============================] - 0s 5ms/step - loss: 0.1853
```

```
3/3 [==============================] - 0s 7ms/step - loss: 0.0184
Epoch 56/60
3/3 [==============================] - 0s 8ms/step - loss: 0.0201
Epoch 57/60
3/3 [==============================] - 0s 8ms/step - loss: 0.0224
Epoch 58/60
3/3 [==============================] - 0s 8ms/step - loss: 0.0187
Epoch 59/60
3/3 [==============================] - 0s 7ms/step - loss: 0.0196
Epoch 60/60
3/3 [==============================] - 0s 7ms/step - loss: 0.0139
1/1 [==============================] - 0s 327ms/step
Tomorrow's SEC price : [54450.46] KRW

Process finished with exit code 0
```