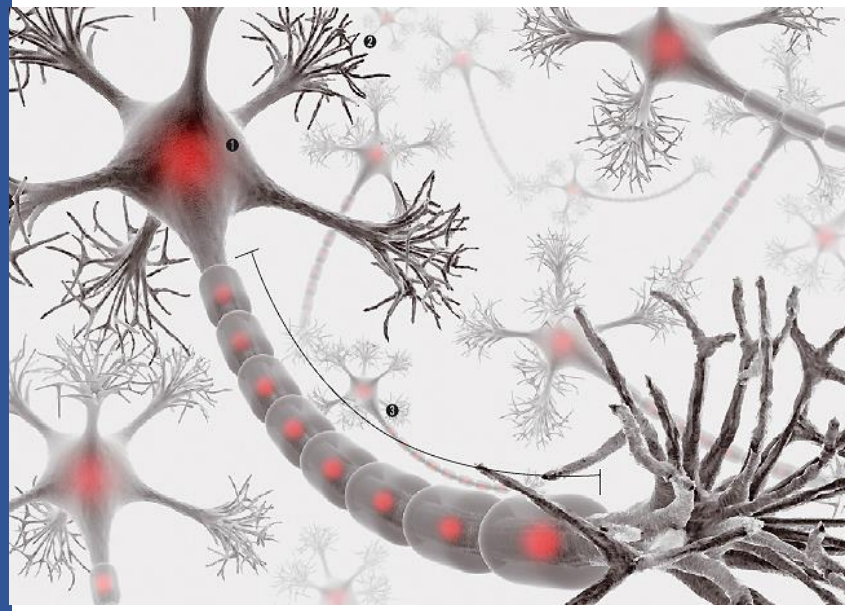
(DNN) Sine Regression

학습 목표

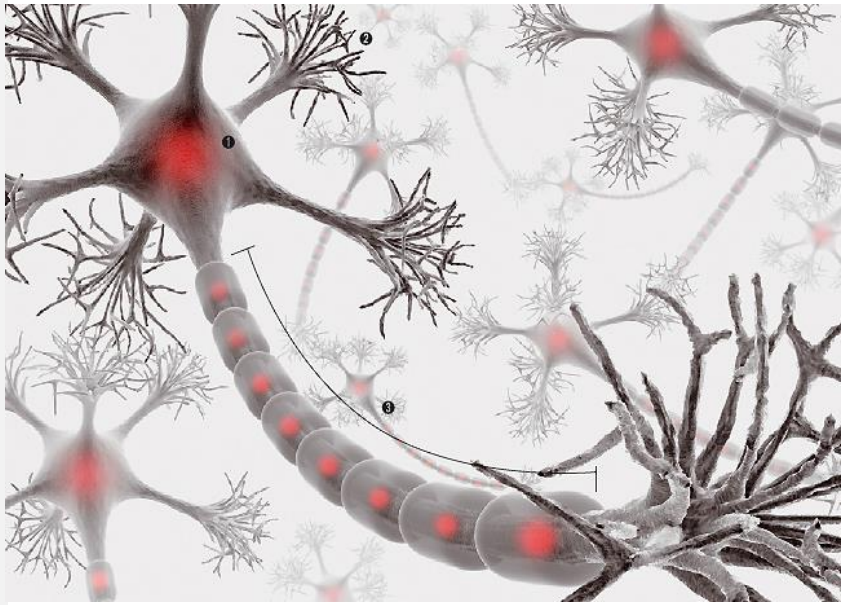
- Sine 함수에서 샘플링된 데이터를 Regression하는 신경망 모델을 Keras로 만들어 본다.

주요 내용

- 1. 문제 정의
- 2. 데이터 준비
- 3. 모델 정의 및 훈련, 검증



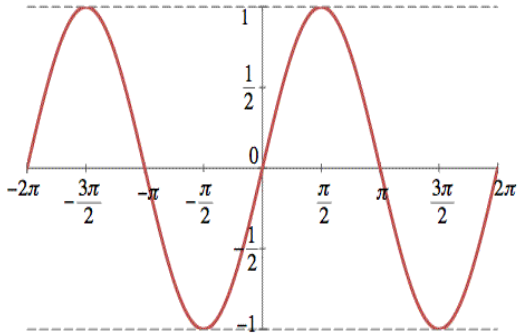
1 문제 정의



Regression 문제

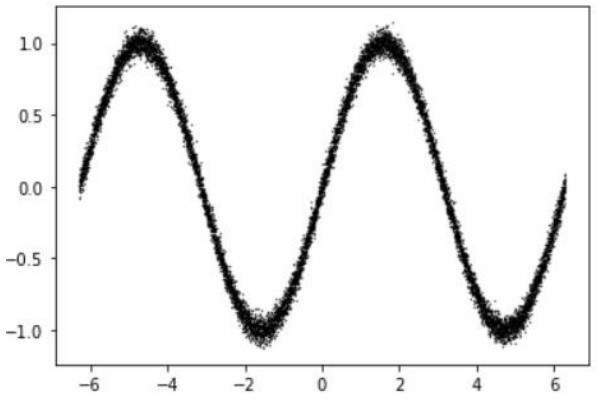
Sine 함수에서 샘플링한 데이터를 Regression해보자!

Sine Function

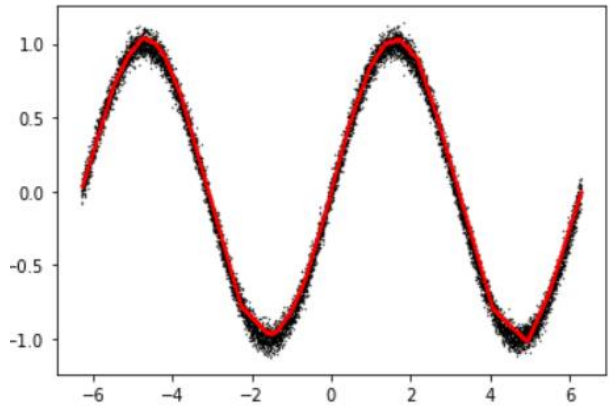


$$f(\theta) = \sin(\theta)$$

Data Sampling

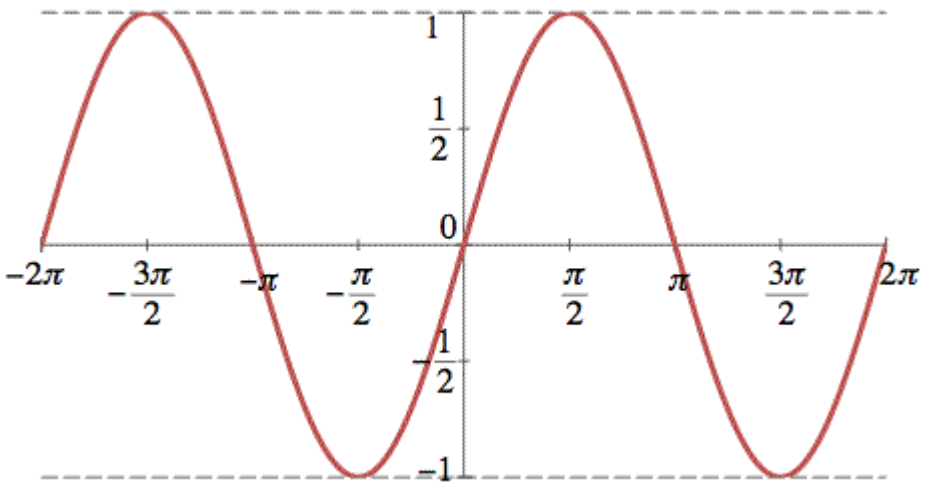


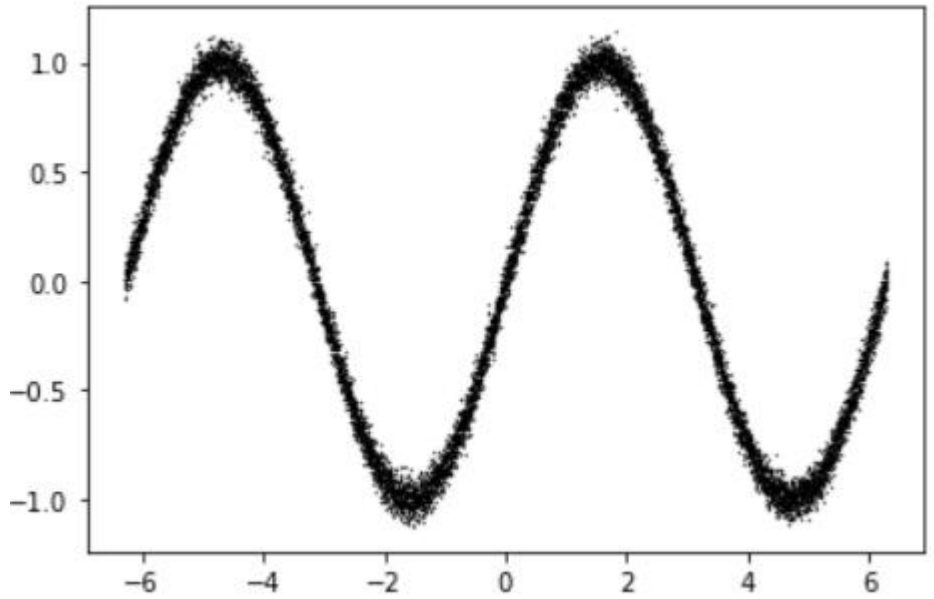
Regression



Hint : Dataset Sampling

$$f(\theta) = \sin(\theta)$$





Input 생성

θ : $[-2\pi, 2\pi]$에서 10000개의 θ를 등간격 샘플링

Label 생성

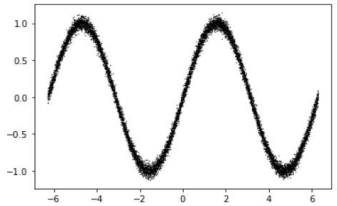
$$\sin(\theta) + 0.05 * \mathcal{N}(0,1)$$

표준정규분포 Noise 추가

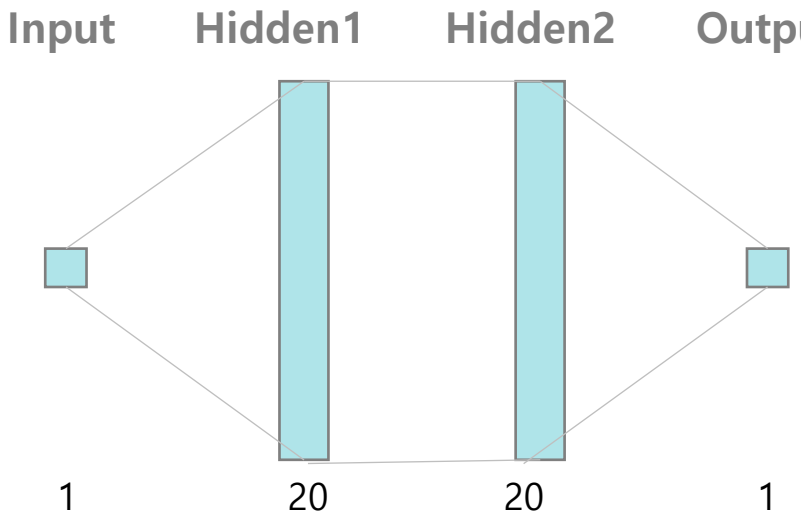
Hint : Network 구성

$x = (\theta)$

- 1차원 입력 벡터



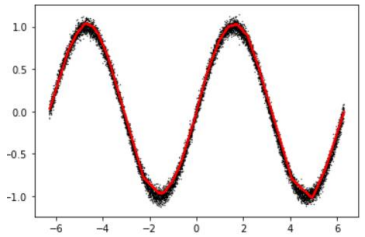
Fully Connected Network



- 2-hidden layers
- 20 neurons
- ReLu

$f(\theta) = \sin(\theta)$

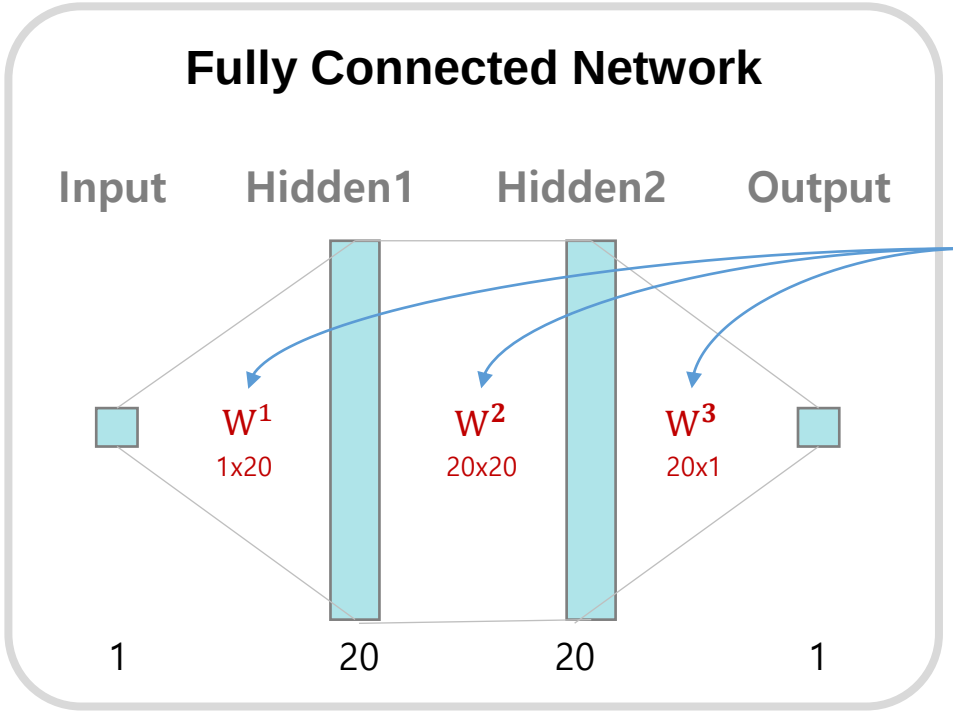
- Real-valued Scalar



Hint : Network 구성

$x = (\theta)$

- 1차원 입력 벡터

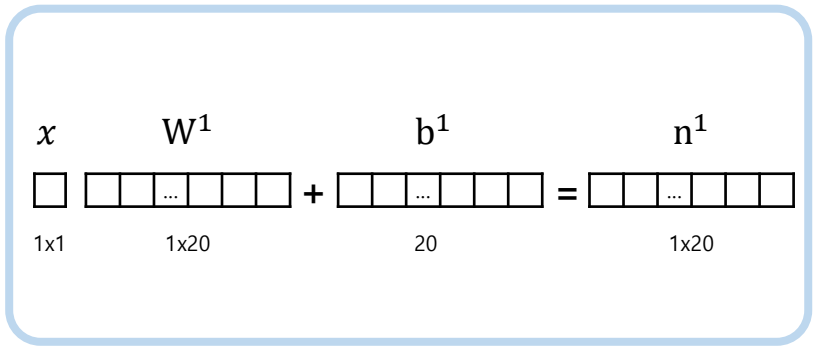


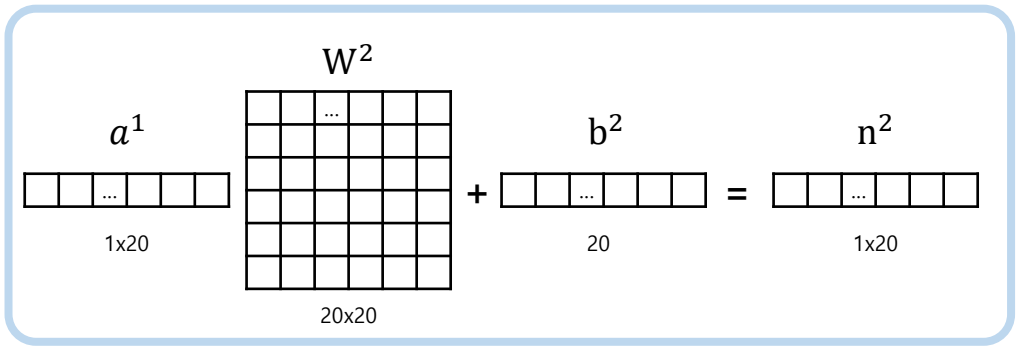
가중치 행렬의 크기는
(이전 Layer 뉴런 개수 x 현재 Layer 뉴런 개수)

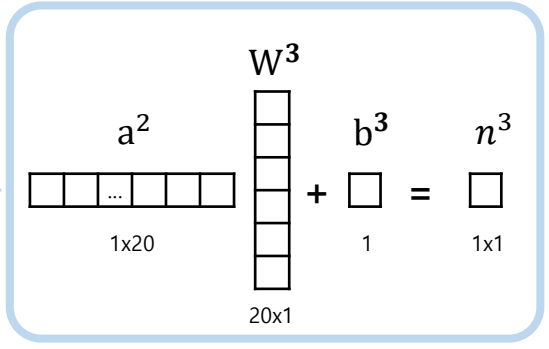
$f(\theta) = \sin(\theta)$

- Real-valued Scalar

입력 샘플이 1개 때

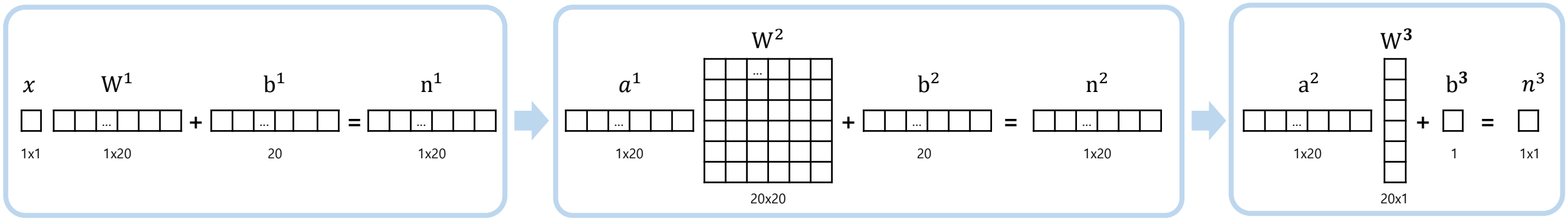






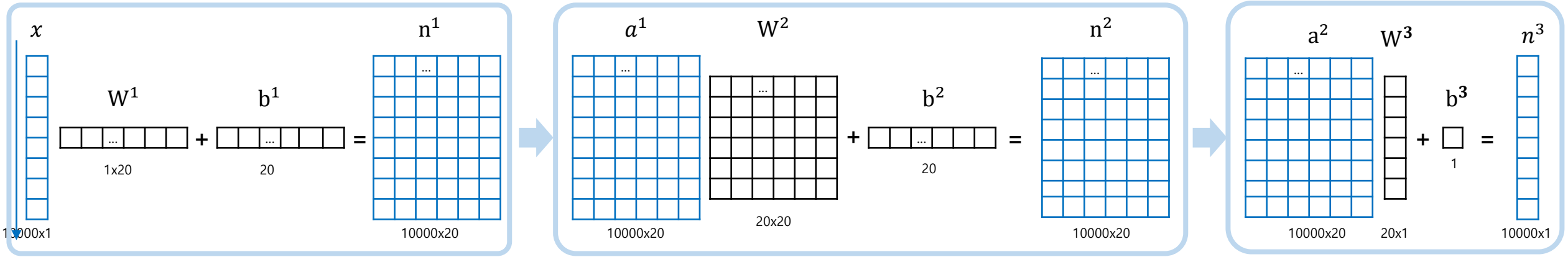
Hint : Network 구성

입력 샘플이 1개 때



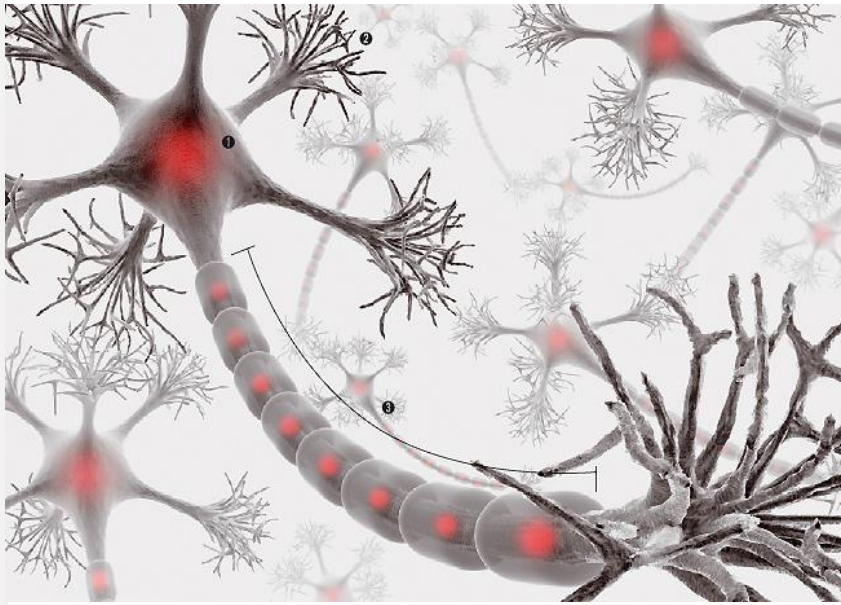
입력 배치가 10000일 때

Batch size



- 1차원이 배치 크기 : 뉴런의 입력과 출력 행렬의 1차원이 배치 크기에 따라 조정됨

2 데이터 준비



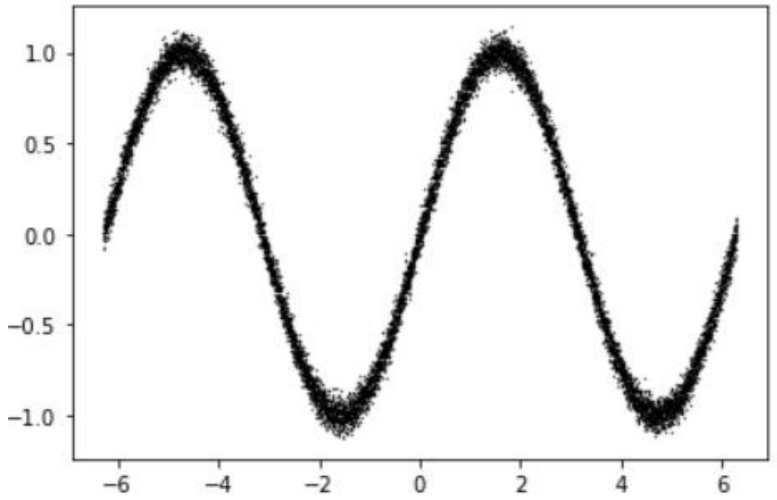
패키지 임포트

```
# tensorflow를 임포트합니다
import tensorflow as tf

# 헬퍼(helper) 라이브러리를 임포트합니다
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

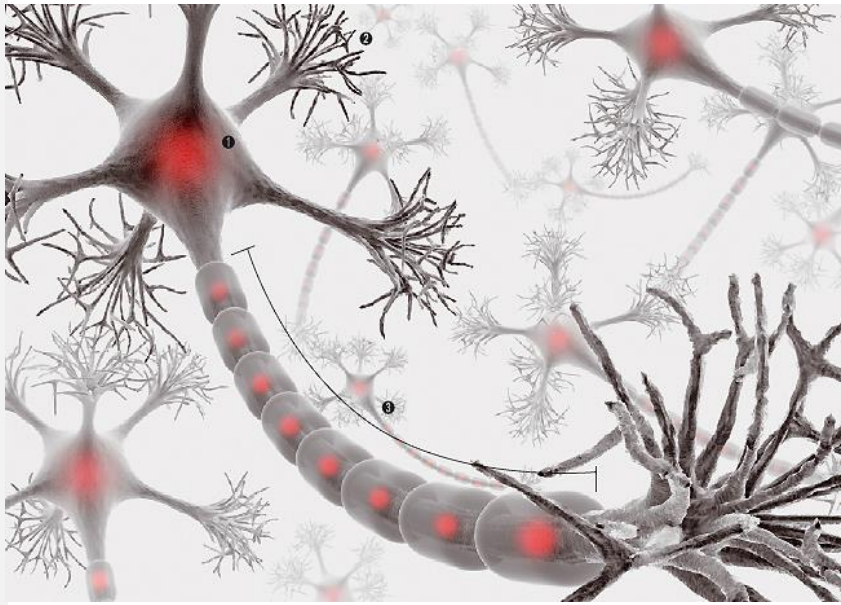
```
2.0.0-dev20190524
```

데이터셋 생성

```
# generate the data
inputs = np.linspace(-2*np.pi, 2*np.pi, 10000)[:, None]                    # inputs   (10000, 1)
outputs = np.sin(inputs) + 0.05 * np.random.normal(size=[len(inputs),1])  # outputs (10000, 1)

plt.scatter(inputs[:, 0], outputs[:, 0], s=0.1, color='k', marker='o')
```



3 모델 정의 및 훈련, 검증

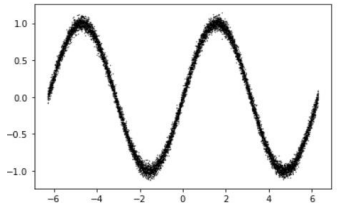


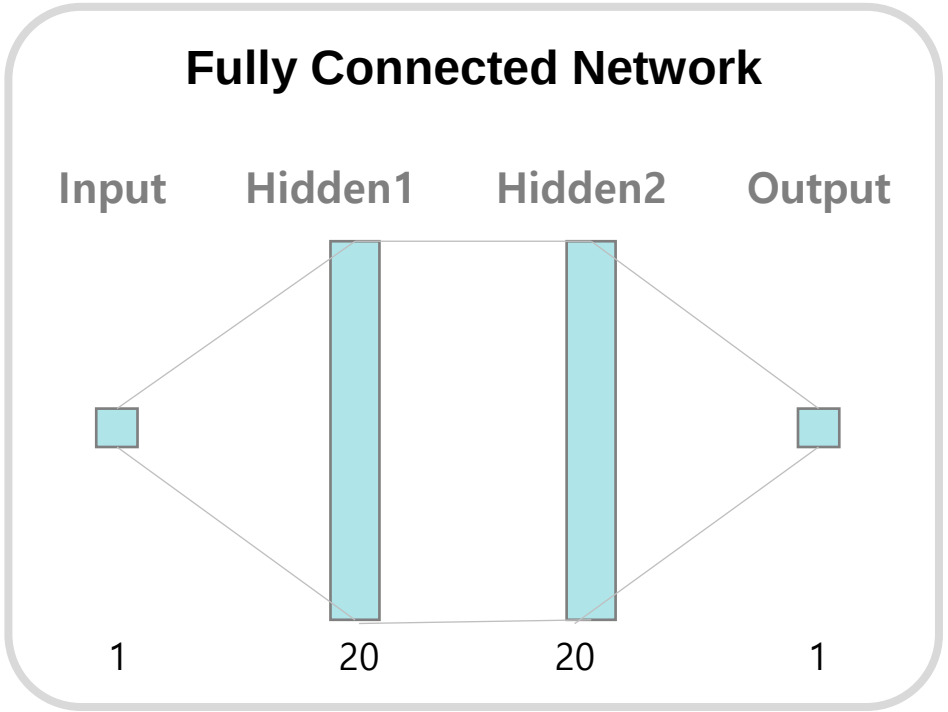
문제

Sine Function Regression을 Keras로 구현해보자!

$x = (\theta)$

- 1차원 입력 벡터

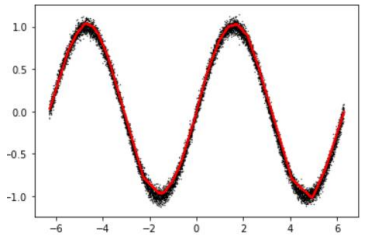




- 2-hidden layers
- 20 neurons
- ReLu

$f(\theta) = \sin(\theta)$

- Real-valued Scalar



모델 정의 (문제)

```
# Dense를 이용해서 3계층 신경망을 구축하시오.
# hidden 20, hidden 20, output 1
# 첫번째 Dense 계층에 input_shape을 지정하시오
model = keras.Sequential(#your code)
```

참고 tf.keras.layers.Dense

```
tf.keras.layers.Dense(
    units, activation=None, use_bias=True, kernel_initializer='glorot_uniform',
    bias_initializer='zeros', kernel_regularizer=None, bias_regularizer=None,
    activity_regularizer=None, kernel_constraint=None, bias_constraint=None,
    **kwargs
)
```

- **units**: 뉴런 개수, Positive integer, dimensionality of the output space.
- **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
- **use_bias**: Boolean, whether the layer uses a bias vector.
- **kernel_initializer**: Initializer for the kernel weights matrix.
- **bias_initializer**: Initializer for the bias vector.
- **kernel_regularizer**: Regularizer function applied to the kernel weights matrix.
- **bias_regularizer**: Regularizer function applied to the bias vector.
- **activity_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
- **kernel_constraint**: Constraint function applied to the kernel weights matrix.
- **bias_constraint**: Constraint function applied to the bias vector.

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

모델 정의 (정답)


```
model = keras.Sequential([
  keras.layers.Dense(20, activation='relu', input_shape=(1,)),
  keras.layers.Dense(20, activation='relu'),
  keras.layers.Dense(1)
])
```

모델 훈련 (문제)

```
# loss, optimizer, metric 설정
# optimizer는 'adam'으로 metric은 'mae', 'mse'로 설정
model.compile(optimizer= #your code,
                        loss= #your code,
                        metrics= #your code)

# epoch는 20 이상
model.fit(#your code)
```

```
Epoch 1/20
313/313 [==============================] - 0s 1ms/step - loss: 0.2912 - mae: 0.4602 - mse: 0.2912
Epoch 2/20
313/313 [==============================] - 0s 1ms/step - loss: 0.1331 - mae: 0.2897 - mse: 0.1331
Epoch 3/20
313/313 [==============================] - 0s 1ms/step - loss: 0.1186 - mae: 0.2587 - mse: 0.1186
…
Epoch 20/20
313/313 [==============================] - 0s 1ms/step - loss: 0.0050 - mae: 0.0551 - mse: 0.0050
```

모델 훈련 (정답)


```
# loss, optimizer, metric 설정
# optimizer는 'adam'으로 metric은 'mae', 'mse'로 설정
model.compile(optimizer='adam',
              loss='mse',
              metrics=['mae', 'mse'])

# epoch는 20 이상
model.fit(inputs, outputs, epochs=20)
```

```
Epoch 1/20
313/313 [==============================] - 0s 1ms/step - loss: 0.2912 - mae: 0.4602 - mse: 0.2912
Epoch 2/20
313/313 [==============================] - 0s 1ms/step - loss: 0.1331 - mae: 0.2897 - mse: 0.1331
Epoch 3/20
313/313 [==============================] - 0s 1ms/step - loss: 0.1186 - mae: 0.2587 - mse: 0.1186
…
Epoch 20/20
313/313 [==============================] - 0s 1ms/step - loss: 0.0050 - mae: 0.0551 - mse: 0.0050
```

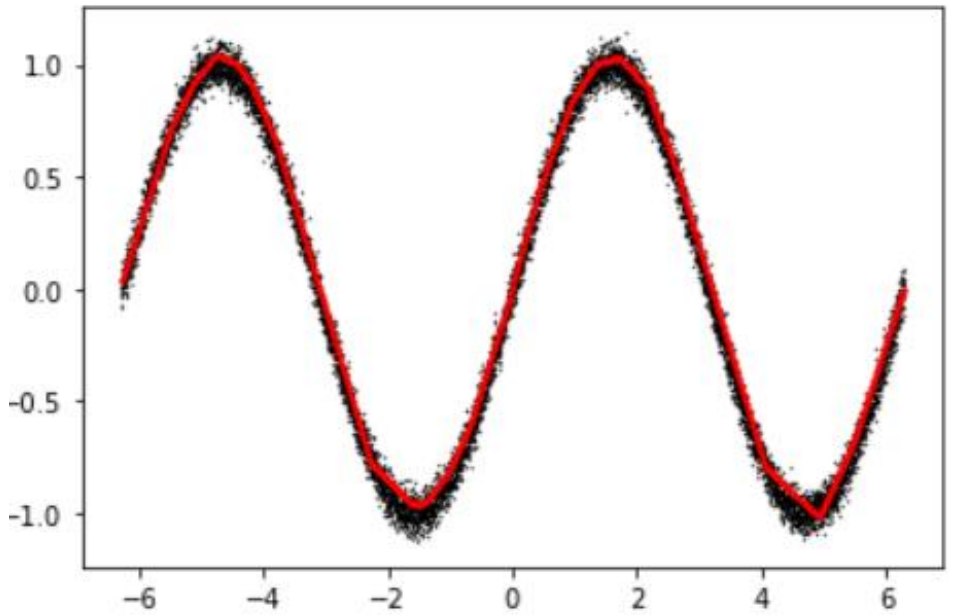
모델 평가

```
test_loss, test_mse, test_mae = model.evaluate(inputs, outputs, verbose=0)
print('\n테스트 MSE:', test_mse)
```

```
테스트 MSE: 0.046272732
```

모델 예측

```
test_output_pred = model.predict(inputs)

plt.scatter(inputs[:, 0], test_output[:, 0], c='k', marker='o', s=0.1)
plt.scatter(inputs[:, 0], test_output_pred[:, 0], c='r', marker='o', s=0.1)
```



문제

모델의 크기를 늘리고 Epoch 수를 늘려보세요!

성능이 어떻게 달라졌는지 살펴보도록 합니다.

Thank you!