

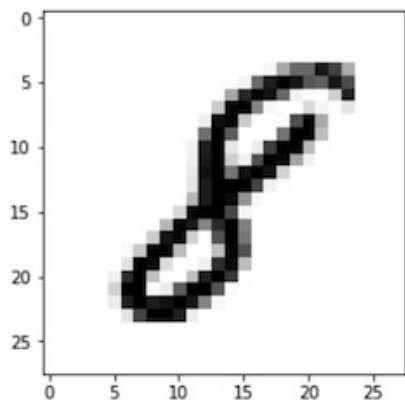
# 집값 예측



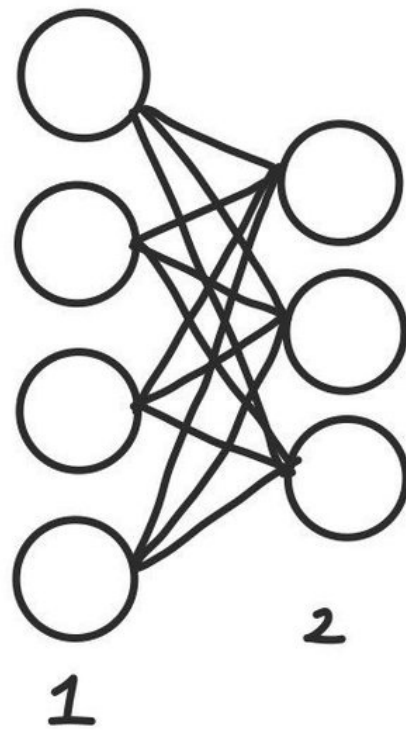
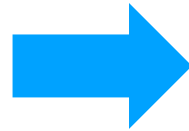
# 프로세스

- 데이터셋 (어떤 데이터셋인가?, 어떻게 구성되어 있나?)
- 데이터 전처리
- 모델 수립
- 학습/검증
- 시험/평가/가시화

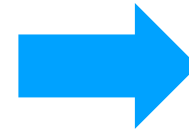
# 예) MNIST 데이터셋 학습



데이터셋  
+  
전처리



모델 수립



Value

이게 얼마가 될거 같니?

output을  
그냥 받는다.

MSE: Mean Squared Error

O/X

기냐? 아니냐?

output에  
sigmoid를 먹인다.

Binary CrossEntropy

Category

종류중에 요건 뭐냐?

output에  
softmax를 먹인다.

Categorical CrossEntropy

Loss 함수 결정

# 데이터 살펴보기

여기서 볼 수 있듯이 404개의 훈련 샘플과 102개의 테스트 샘플이 있고 모두 13개의 수치 특성을 가지고 있습니다.

1. Per capita crime rate.
2. Proportion of residential land zoned for lots over 25,000 square feet.
3. Proportion of non-retail business acres per town.
4. Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
5. Nitric oxides concentration (parts per 10 million).
6. Average number of rooms per dwelling.
7. Proportion of owner-occupied units built prior to 1940.
8. Weighted distances to five Boston employment centres.
9. Index of accessibility to radial highways.
10. Full-value property-tax rate per \$10,000.
11. Pupil-teacher ratio by town.
12.  $1000 * (B_k - 0.63) ** 2$  where  $B_k$  is the proportion of Black people by town.
13. % lower status of the population.

# 데이터 살펴보기

타겟은 주택의 중간 가격으로 천달러 단위입니다:

```
[ ] train_targets
```

```
array([[15.2, 42.3, 50. , 21.1, 17.7, 18.5, 11.3, 15.6, 15.6, 14.4, 12.1,
        17.9, 23.1, 19.9, 15.7,  8.8, 50. , 22.5, 24.1, 27.5, 10.9, 30.8,
        32.9, 24. , 18.5, 13.3, 22.9, 34.7, 16.6, 17.5, 22.3, 16.1, 14.9,
        23.1, 34.9, 25. , 13.9, 13.1, 20.4, 20. , 15.2, 24.7, 22.2, 16.7,
        12.7, 15.6, 18.4, 21. , 30.1, 15.1, 18.7,  9.6, 31.5, 24.8, 19.1,
        22. , 14.5, 11. , 32. , 29.4, 20.3, 24.4, 14.6, 19.5, 14.1, 14.3,
        15.6, 10.5,  6.3, 19.3, 19.3, 13.4, 36.4, 17.8, 13.5, 16.5,  8.3,
        14.3, 16. , 13.4, 28.6, 43.5, 20.2, 22. , 23. , 20.7, 12.5, 48.5,
        14.6, 13.4, 23.7, 50. , 21.7, 39.8, 38.7, 22.2, 34.9, 22.5, 31.1,
        28.7, 46. , 41.7, 21. , 26.6, 15. , 24.4, 13.3, 21.2, 11.7, 21.7,
        19.4, 50. , 22.8, 19.7, 24.7, 36.2, 14.2, 18.9, 18.3, 20.6, 24.6,
        18.2,  8.7, 44. , 10.4, 13.2, 21.2, 37. , 30.7, 22.9, 20. , 19.3,
        31.7, 32. , 23.1, 18.8, 10.9, 50. , 19.6,  5. , 14.4, 19.8, 13.8,
        19.6, 23.9, 24.5, 25. , 19.9, 17.2, 24.6, 13.5, 26.6, 21.4, 11.9,
        22.6, 19.6,  8.5, 23.7, 23.1, 22.4, 20.5, 23.6, 18.4, 35.2, 23.1,
        27.9, 20.6, 23.7, 28. , 13.6, 27.1, 23.6, 20.6, 18.2, 21.7, 17.1,
        8.4, 25.3, 13.8, 22.2, 18.4, 20.7, 31.6, 30.5, 20.3,  8.8, 19.2,
        19.4, 23.1, 23. , 14.8, 48.8, 22.6, 33.4, 21.1, 13.6, 32.2, 13.1,
        23.4, 18.9, 23.9, 11.8, 23.3, 22.8, 19.6, 16.7, 13.4, 22.2, 20.4,
        21.8, 26.4, 14.9, 24.1, 23.8, 12.3, 29.1, 21. , 19.5, 23.3, 23.8,
```

# 데이터 셋

```
from keras.datasets import boston_housing  
  
(train_data, train_targets), (test_data, test_targets) = boston_housing.load_data()
```

- Keras에서 기본 데이터 셋으로 제공
- 404개의 훈련 샘플 (102개의 테스트 샘플)
- 13개의 수치 특성 (범죄율, 주차장 크기, 상업시설, 주변집들의 연식 등)
- 핵심: 입력(주택값에 영향을 미치는 특징 13개), 출력(주택 값)





# 전처리

가 .

- 숫자 데이터를 정규화하여 입력으로 사용

```
[ ] mean = train_data.mean(axis=0)
    train_data -= mean
    std = train_data.std(axis=0)
    train_data /= std

    test_data -= mean
    test_data /= std
```

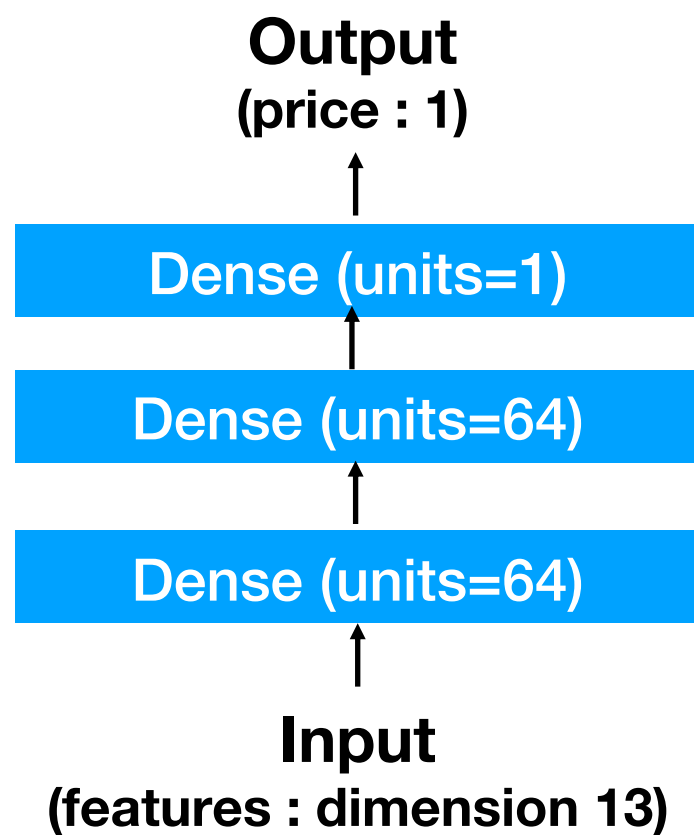
# 신경망 모델 만들기

- 모델을 코딩해 보자
- `m = models.Sequential()`
- `m.add(layers.Dense(...))`
- ...
- 입력 차원은 ~~10000 차원의 데이터 (One-hot)~~ 13차원
- 출력 차원은 1차원 (~~Positive일 확률~~) 집값

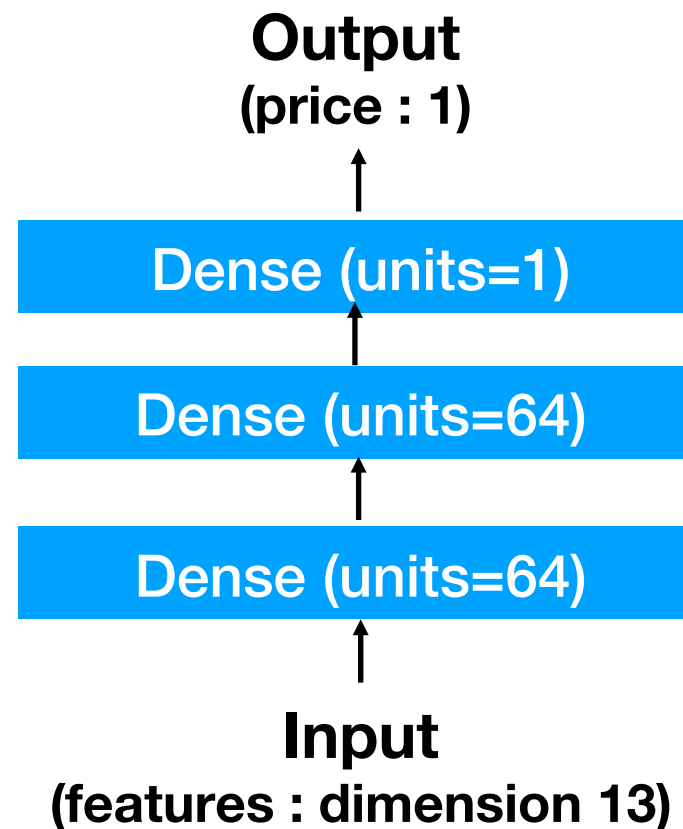
activation



# 신경망 모델 만들기



# 신경망 모델 만들기



```
model = models.Sequential()  
model.add(layers.Dense(64, activation='relu', input_shape=(train_data.shape[1],)))  
model.add(layers.Dense(64, activation='relu'))  
model.add(layers.Dense(1))
```

# 최적화

- Optimizer는?
- Loss 함수는?
- Metrics는?

# 최적화

- Optimizer는?
- Loss 함수는?
- Metrics는?

```
model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
```

이 모델은 `mse` 손실 함수를 사용하여 컴파일합니다. 이 함수는 평균 제곱 오차(mean squared error)의 약자로 예측과 타겟 사이 거리의 제곱입니다. 회귀 문제에서 널리 사용되는 손실 함수입니다.

훈련하는 동안 모니터링을 위해 새로운 지표인 평균 절대 오차를 측정합니다. 이는 예측과 타겟 사이 거리의 절댓값입니다. 예를 들어 이 예제에서 MAE가 0.5이면 예측이 평균적으로 \$500 정도 차이가 난다는 뜻입니다.

# 학습

```
# 전체 데이터로 훈련시킵니다
model.fit(train_data, train_targets,
          epochs=80, batch_size=16, verbose=0)
test_mse_score, test_mae_score = model.evaluate(test_data, test_targets)
```