3.1 回帰の例：住宅価格の予測

17T4028N 菊田尚樹

回帰とは

回帰：離散的なラベルではなく、連続値を予測すること

(例) ・気象データに基づいて明日の天気を予想
・ソフトウェアの仕様に基づいてプロジェクトの完了にかかる時間を予想

(注) ロジスティック回帰は回帰ではなく分類アルゴリズム

データセットの読み込み

Boston_housingデータセット

1970年代中頃のボストン近郊の住宅価格中央値を予測する

↑犯罪発生率・平均部屋数などの要素で予測

```
from keras.datasets import boston_housing

(train_data, train_targets), (test_data, test_targets) = \
    boston_housing.load_data()
```

```
>>> train_data.shape
(404, 13)
>>> test_data.shape
(102, 13)
```

訓練データが404個

テストデータが102個

どちらも13種類の要素(特徴量)を持つ

(注) 特徴量はそれぞれ異なる尺度で表されている

0~1, 0~100, 1~12 など

データの準備

尺度の異なる特徴量を用いるのは困難

→正規化：平均を0,標準偏差を1

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

test_data -= mean
test_data /= std
```

平均を引いて標準偏差で割る

◇訓練データ、テストデータともに、

変数 mean,std (訓練データの平均,標準偏差)で正規化されている

テストデータを使って計算した値は使用してはいけない

ニューラルネットワークの構築

```
from keras import models
from keras import layers

def build_model():
    # 同じモデルを複数回インスタンス化する必要があるため、
    # モデルをインスタンス化するための関数を使用
    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))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model
```

中間層は2層
各64ユニット

データが少ない場合は過学習に陥りやすい
→小さなネットワークを利用することで防ぐ

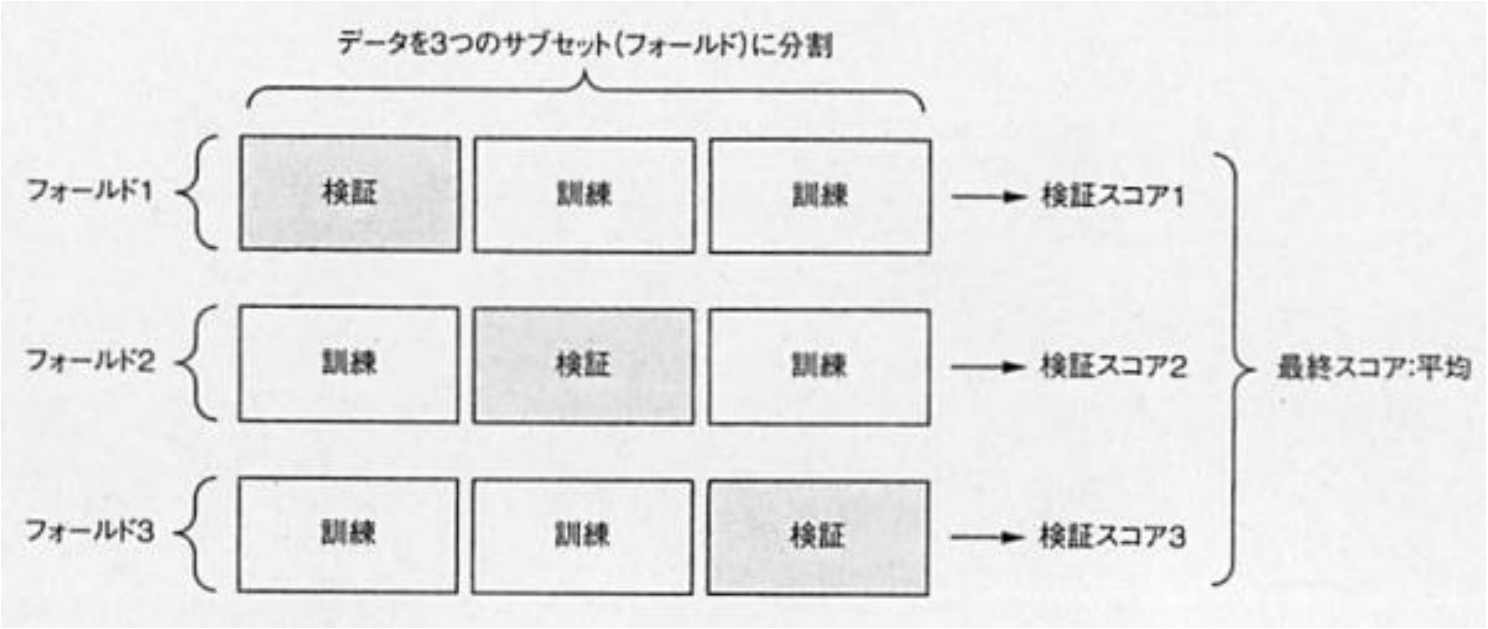
lose = ‘mse’ : 損失関数は平均二乗誤差
metrics = [‘mae’] ： 評価指標は平均絶対誤差

最後の層は活性化関数を使用していない
→範囲に制限を設けないため

K分割交差検証

少ないデータで学習する場合、訓練データとテストデータの分け方による影響が大きいため、通常の分け方では正しい評価はできない

→ K分割交差検証



K個に分割し、一つをテストデータ、残りを訓練データとして使いスコアを算出することテストデータを交換しながらK回行う

K分割交差検証

コード

```
import numpy as np

k = 4
num_val_samples = len(train_data) // k
num_epochs = 100
all_scores = []
    for i in range(k):
    print('processing fold #', i)

    # 検証データの準備：フォールドiのデータ
    val_data = \
        train_data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = \
        train_targets[i * num_val_samples: (i + 1) * num_val_samples]

    # 訓練データの準備：残りのフォールドのデータ
    partial_train_data = np.concatenate(
        [train_data[:i * num_val_samples],
         train_data[(i + 1) * num_val_samples:]],
        axis=0)
    partial_train_targets = np.concatenate(
        [train_targets[:i * num_val_samples],
         train_targets[(i + 1) * num_val_samples:]],
        axis=0)

    # Kerasモデルを構築（コンパイル済み）
    model = build_model()

    # モデルをサイレントモード（verbose=0）で適合
    model.fit(partial_train_data, partial_train_targets,
              epochs=num_epochs, batch_size=1, verbose=0)

    # モデルを検証データで評価
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_mae)
```

K分割交差検証

結果

```
>>> all_scores
[2.0750808349930412, 2.117215852926273, 2.9140411863232605,
2.4288365227161068]
>>> np.mean(all_scores)
2.3837935992396706
```

◇all_scoresはk個の平均絶対誤差の配列

K分割交差検証によって求めた平均値はどれか一つの検証スコアよりも信頼できる

平均値、約2.4は2,400ドルを示す

住宅価格が10,000~50,000なのでこれでも誤差はかなり大きい…

回帰のまとめ

・回帰で用いる損失関数は平均二乗誤差(mse)

・回帰で用いる評価指標は平均絶対誤差(mae)

・入力データとして使う特徴量がそれぞれ異なる尺度のときは
正規化を行い調整する

・利用するデータが少ない場合、k分割交差検証を行うほうがよい

・訓練データが少ない場合、小さなネットワーク(中間層が少ない)
にすることで過学習を防ぐ