15. データサイエンス

プログラミング・データサイエンス I

2023/7/27

1　今日の目的

今日の目的

- データからの推計
- 機械学習

この講義は、「AI・プログラミング・データサイエンス」ついた一連の講義の端緒となる科目です。この講義の最後として、データサイエンスの香りをちょっとだけ吸ってみましょう。

データサイエンスとは、データの分析に基づき、様々な課題解決を行うことを目指すものです。その基本には、統計学などの数学とプログラミングやシミュレーション等の計算機科学があります。データサイエンスを活用する入り口として、具体的な例を見ることにします。今日のサンプルプログラムは、用意していますが、新しいパッケージが必要であり、動作しない人がいる懸念があります。また、大量の計算が必要であり、コンピュータによっては、負荷が大きくなりすぎます。話を聞くだけで結構です。

2　推計

2.1　標本

初めに、データからその統計的特徴量を推計することを考えましょう。データは、母集団から得るサンプルに過ぎないことに注意します。ここでは、簡単なシミュレーションを通じた例を示します。この例は以下にあります。

`https://github.com/first-programming-saga/dataScience`

平均 μ、標準偏差 σ の母集団を考えます。ここから n 個のサンプル $\{x_i\}$ を無作為抽出

しましょう。標本平均と標本標準偏差は

$$\overline{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i \tag{2.1}$$

$$s^2 = \frac{1}{n-1}\sum_{i=0}^{n-1} (x_i - \overline{x})^2 \tag{2.2}$$

となります。標本標準偏差を計算する際に、n ではなく $n-1$ で除していることに注意してください。

確率統計の重要な定理である中心極限定理によると、標本平均 $\overline{x}$ は、平均 μ、標準偏差 $\sigma/\sqrt{n}$ の正規分布に従います。このことは、母集団が平均と標準偏差を有していれば、その具体的な分布に依存しません。すごく強い内容を主張していますね。このような理由により、正規分布が偏在しているのです。

一様分布を例に考えます。一様分布とは、ある区間に標本が一様に分布するものです。$[0,1)$ の一様分布を図 1 に示します。この図は、疑似乱数を $100,000$ 個生成し、$1/100$ の大きさの区間のそれぞれに入る相対頻度を表しています。

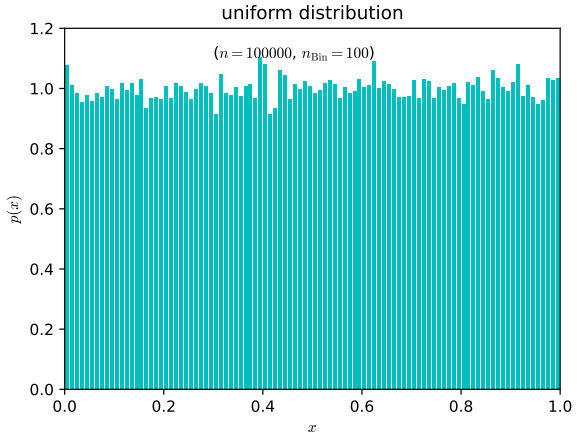

図 1　$[0,1)$ の一様分布。データ数 $10,000$。bin の数 100。

$[0,1)$ の一様分布に対して、大きさ 100 の標本を $10,000$ 個作って、標本平均の分布を求めてみましょう (`uniform.ipynb`)。

次に、標本平均 $\overline{x}$ の分布を調べましょう (`distributionOfSampleMean.ipynb`)。ソースコード 2.1 を見てください。こちらは、$[0,1)$ の一様分布に従う乱数から、標本平均を

求めるものです。母集団の平均は $\mu = 1/2$、分散は $\sigma^2 = 1/12$ です。標本平均 $\bar{x}$ は、図 2 に示すように、平均 μ、標準偏差 $\sigma/\sqrt{n}$ の正規分布となっています。

ソースコード 2.1　標本平均を求める

```python
def evalSampleMean(n: int) -> float:
    """
    大きさ n の標本に対して、平均を返す
    """
    x = 0
    for i in range(n):
        x += np.random.rand()#[0,1) の乱数
    return x / n

#標本平均を作る
sample_size = 100 #標本サイズ
num_samples = 10000 #標本数
sample_means = np.zeros(num_samples)
for i in range(num_samples):
    sample_means[i] = evalSampleMean(sample_size)
```

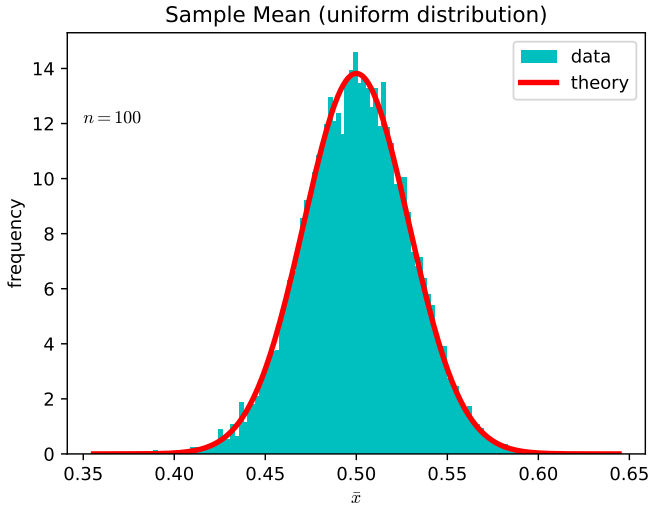

図 2　$[0, 1)$ の一様分布に対する標本平均の分布。標本サイズは 100

2.2 平均の推計

本来、母集団の平均と標準偏差を直接に知ることはできません。しかし、前述のように標本平均 $\bar{x}$ が正規分布に従うことを使うと、母集団の平均を推計することができます。標本平均 $\bar{x}$ が、平均 μ、標準偏差 $\sigma/\sqrt{n}$ の正規分布に従うことを出発点にしましょう。ただし、母集団の標準偏差 σ も知ることはできませんから、標本標準偏差 s で代用します。正規分布は、平均を中心としてある幅に入る確率を数値的に知ることができます。

標本平均 $\bar{x}$ と母集団平均 μ との差が、標本標準偏差のある範囲に入っているとしましょう。$f > 0$ はその範囲を定めるパラメタです。

$$-f\frac{s}{\sqrt{n}} < \bar{x} - \mu < f\frac{s}{\sqrt{n}} \tag{2.3}$$

95% の確率 (これを信頼係数という) でこの範囲に入るとすると、$f \simeq 1.96$ が該当します。これを変形することで、標本平均から信頼係数に依存して、母集団平均が入っている区間を推定することができます。

$$\bar{x} - \frac{fs}{\sqrt{n}} < \mu < \bar{x} + \frac{fs}{\sqrt{n}} \tag{2.4}$$

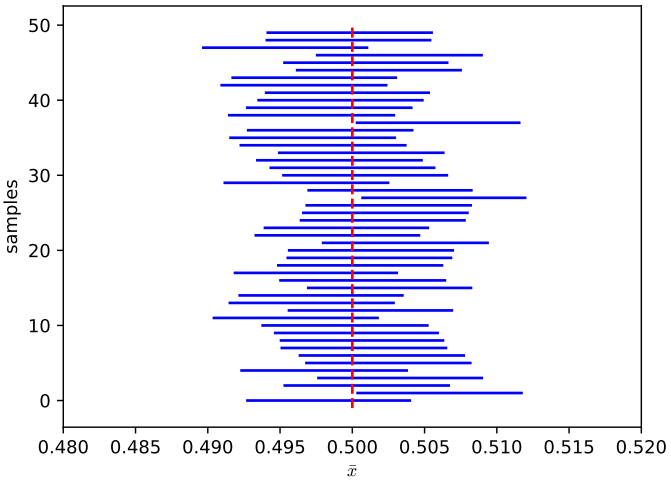

図 3　$[0, 1)$ の一様分布に対する標本平均から、母集団の平均の範囲を 95% で推計

ソースコード 2.2　標本平均から母集団平均を推定

```
def estimateRange(mean: float, s: float, n: int, f: float) ->
↪  dict[str, float]:
    """
    区間を推定
    """
    minX = mean - f * s / np.sqrt(n)
    maxX = mean + f * s / np.sqrt(n)
    return {'lower' : minX, 'higher' : maxX}

def evalSample(n : int) -> tuple[float, float]:
    """
    大きさ n の標本に対して、平均と分散を返す
    """
    d = np.zeros(n)
    for i in range(n):
        d[i] = np.random.rand()#[0,1) の乱数
    x = 0
    for i in range(n):
        x += d[i]
    mean = x / n
    s = 0
    for i in range(n):
        s += (d[i] - mean) * (d[i] - mean)
    s = s / (n - 1)
    return mean, np.sqrt(s)

alpha = 0.05 #信頼度
f = norm.isf(alpha / 2) #信頼度に対応した標準正規分布の変位
sampleSize = 10000 #標本サイズ
num_trial = 50 #標本数
result = list()
for i in range(num_trial):
    m, s = evalSample(sampleSize)
    r = estimateRange(m, s, sampleSize, f)
    result.append(r)
```

実際に、$[0, 1)$ の一様分布から 1000 個のサンプルを取り出し、95% の信頼度で、母集団平均を推計するイメージを示します。プログラムをソースコード 2.2 に示します (estimatingPopulationMean.ipynb)。図 3 の横棒は、各サンプルで推定した母集団平均の範囲を表しています。縦の破線は、正しい母集団平均です。

3　機械学習

3.1　機械学習とは

次は機械学習を考えましょう。コンピュータに、大量の例を見せて、学習をさせる方法です。

最初に、ヒトが学習する過程を想像しましょう。小さなこどもが文字を覚える過程には、まだまだ分からないことがたくさんありますが、大まかには以下のような過程と考えることができるでしょう。子どもは、周囲にある書かれた文字を見て、あるいは大人が文字を書く様子をみて、その形や筆順をマネして書くことを繰り返します。ひらがなであれば、形と音を組み合わせて覚えていきます。時には、周囲の大人が、形や音を修正をします。このように、文字の形とそのラベル (ひらがなの場合には音) を、あるいはどれとどれが同じか異なるかを試行錯誤しながら覚えます。

成長してから漢字、特に画数の多い漢字を覚えることを想像しましょう。小さなこどもが、ひらがなを覚えるのとは少し違う方法をとっていると思われます。自分が知っている漢字の部品と同じものは覚えやすいのですが、知らない部品で出来ていると覚えるのが大変ですね。このときの学習過程は、すでに、文字という記号、つまり抽象化された概念を道具として使っているのです。

人工知能の研究が始まったころには、後者のように、既知の知識を集め、それに基づいて、コンピュータが何かを判断することが目標とされていました。狭くて明確な知識領域への応用では一定の成果を挙げましたが、知識領域が広い場合や、曖昧さを含む情報に基づくものへの応用は困難でした。

現在の人工知能の中核的手法である機械学習が目指すのは前者の方法です。データそのものから、アルゴリズムを使って特徴を学習し、それに基づき判断をすることを目指しています。そのため、アルゴリズムが学習した特徴を、ヒトが分かるように表現することは、困難になります。

機械学習は、データ分析の一つの手法であり、データサイエンスの重要な手法の一つです。今回は、手書きの数字を判別する機械学習の例を見ていきましょう。

本日のプログラムは以下にあります。

`https://github.com/first-programming-saga/MNIST`

3.2 手書き数字

アメリカ国立標準技術研究所 (National Institute of Standards and Technology, NIST) は、計量や標準化を担う連邦機関です。NIST が保有していたデータから、機械学習向けに変更したデータが MNIST データです。この中の手書き数字を例にしましょう。

Python には、データ分析のパッケージ scikit-learn があります。インストールには scikit-learn という名前を使うのですが、python コードに import する際には sklearn という名前を使います。

ソースコード 3.1　MNIST データ読み込み

```
1 from sklearn.datasets import fetch_openml
2
3 #データの取得
4 mnist = fetch_openml('mnist_784', version=1)
```

scikit-learn には、MNIST のデータが含まれています。ソースコード 3.1 が、データの読み込み部分です。この中には、$28 \times 28 = 784$ 個のビットからなる、手書き文字画像が 70,000 個入っています。

これから行うのは、教師付き学習 (supervised learning) という機械学習です。つまり、各イメージを表す数字、つまり正答が対応付けられています。この正答をラベルと言います。

サンプルを 100 個、図 4 に示します。ヒトが見ても、判断に迷うようなものも含まれていますね。

3.3 二値分類

機械学習では、データを訓練データと試験データに分割します。訓練データを使ってシステムを学習させます。その後に、システムが試験データに対して正答を答えることができれば成功です。70,000 個のうち 60,000 を訓練データとして、残りを試験データとすることにします (ソースコード 3.2)。

初めに、手書き文字を"5"とそれ以外に分ける学習を考えましょう。データを二つに分類する課題であるため、二値分類 (binary classification) と言います。

図 4　イメージサンプル

ソースコード 3.2　データを訓練データと試験データに分割

```
x_train, x_test =
↪  mnist['data'].iloc[:60000],mnist['data'].iloc[60000:]
y_train, y_test =
↪  mnist['target'].iloc[:60000],mnist['target'].iloc[60000:]
y_train = y_train.astype(np.int8)
y_test = y_test.astype(np.int8)
```

元の画像データは、$28 \times 8 = 784$ 個の 0 と 1 の列です。これを 784 次元空間の点と考えることにします。"5"を表現する点のグループと、それ以外を区別する超平面、今の場合は $784 - 1$ 次元の平面を求めるのが、簡単な手法です。訓練データから、試行錯誤をしてこの面を求めます。ここでは、`SGDClassifier` というモジュールを使います (ソースコード 3.3)。

学習の後に、試験用の $10,000$ 個のデータを"5"とそれ以外に分類します。"5"をそれ以外と判断したものが 158 個、"5"でないものを"5"と判断したものが 182 個ありました。正答率にすると 96 パーセントです。簡単な手法なのに、そこそこ良い結果です。間違えのあった文字の例を示します (図 5)。

ソースコード 3.3　5 とそれ以外に分割して、学習を実行

```
y_train5 = (y_train == 5)
sgd_clf = SGDClassifier(random_state=48)
sgd_clf.fit(x_train,y_train5)
```

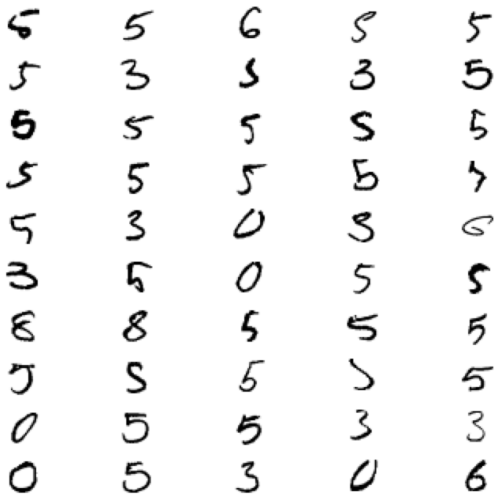

図 5　判断を誤った入力例

3.4　多値分類

一つの文字とそれ以外を区別することを、それぞれの文字を判別することに拡張することは、それほど困難ではありません。ある手書き文字をどの文字と判定するかの「確らしさ」を評価する方法があれば可能です。詳細は、省略します。

正答率を表 1 に示します。概ね良い結果を示していますが、"2"や"5"は誤りが非常に多いことがわかります。結果を詳細に見ると、いずれの文字も"3"と誤って判定する場合が多いことがわかります。

結果を図にしたものを図 6 に示します。横軸は、試験データのラベル、つまり正答です。縦軸は、予想結果です。色が明るいほど件数が多いことを表しています。対角成分が

ラベル	データ数	正答率
0	980	0.98776
1	1135	0.95683
2	1032	0.63953
3	1010	0.96436
4	982	0.87271
5	892	0.55717
6	958	0.91127
7	1028	0.92023
8	974	0.81520
9	1009	0.85927

表 1　各試験データの正答率 (小数 6 桁切り捨て)

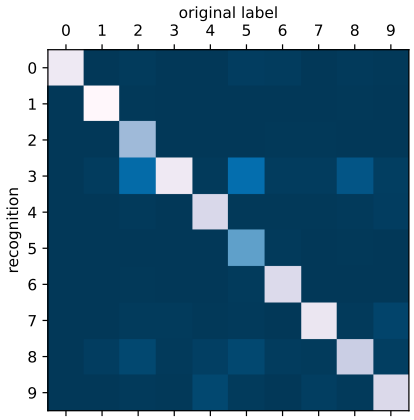

図 6　試験データの分析結果

特に明るくなっていることは、多くの手書き文字を正しく判定していることを表しています。しかし、"2"や"5"は、判定が難しいことを表しています。縦軸の値が"3"であるところが対角成分以外で明るい色のところがあります。これは、"3"と誤って判断する場合が、他の場合よりも多いことを示しています。

3.5　多値分類: その 2

学習手法を少し改善しましょう。784 次元の空間を曲面で区切ることを許しましょう。また、訓練データからその局面をできるだけ離すような方法を使います。

ラベル	データ数	正答率
0	980	0.99286
1	1135	0.99207
2	1032	0.97481
3	1010	0.98515
4	982	0.97862
5	892	0.97646
6	958	0.98539
7	1028	0.96887
8	974	0.97536
9	1009	0.96135

表 2　各試験データの正答率 (小数 6 桁切り捨て)

正答率を表 2 に示します。各文字の正答率が大きく向上していることがわかります。誤り率は 2% 程度になっています。

結果を図示しましょう (図 7)。先ほどの例 (図 6) と比べると、対角成分がさらに明るくなっています。また、対角でない部分の明るい色の部分がなくなりました。予想精度が高くなっていることがわかります。

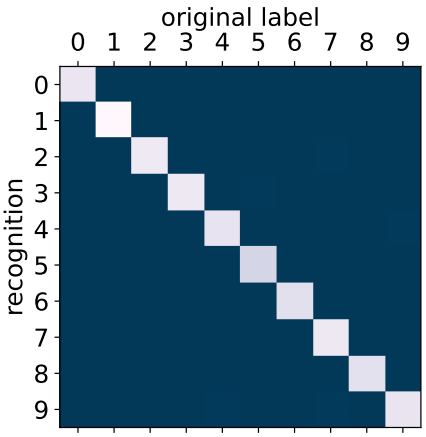

図 7　試験データの分析結果