

# データサイエンス概論I & II データサイエンス総論I & II

まえがき

九州大学 数理・データサイエンス教育研究センター

ようこそ！  
データサイエンス実践特別講座へ！

データ解析技術を学びたい人が  
学問分野の枠を越えて  
自由に学べる場を提供します！

# いまはもう、こんな時代： 「普通の言葉」で思い通りに計算機を操れる

## ChatGPT 4o

“MNISTという手書き  
数字画像データセット  
がどのようなものなのか、  
それぞれのデータと全体の  
状況を可視化する  
pythonプログラムをつ  
つってください。”

MNISTという手書き数字画像データセットがどのようなものなのか、それぞれのデータと全体の状況を可視化するpythonプログラムをつくってください。

MNISTデータセット全体の状況と、それぞれの手書き数字画像を可視化するpythonプログラムを作成します。このプログラムでは、以下の2つを可視化します。

1. 各数字のサンプル画像: 各数字 (0から9) の例を表示する。
2. 全体のラベル分布: データセット全体における各数字の出現回数を棒グラフで示す。

以下に、データの可視化を行うプログラムを示します。

プログラムコード

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

# MNISTデータセットを読み込む
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# 各数字のサンプル画像を表示する
def plot_sample_images(x, y):
    plt.figure(figsize=(10, 5))
    for i in range(10):
        # 各数字 (0-9) の最初のサンプルを取得
        sample_image = x[np.where(y == i)][0]
        plt.subplot(2, 5, i+1)
        plt.imshow(sample_image, cmap='gray')
        plt.axis('off')
        plt.title(f"Label: {i}")
    plt.show()

# 各ラベルの分布を可視化する
def plot_label_distribution(y):
    labels, counts = np.unique(y, return_counts=True)
    plt.figure(figsize=(8, 5))
    plt.bar(labels, counts, tick_label=labels)
    plt.xlabel('Digit')
    plt.ylabel('Count')
    plt.title("Distribution of digits in MNIST dataset")
    plt.show()

# 各数字のサンプル画像を表示
print("Sample images of each digit:")
plot_sample_images(x_train, y_train)

# トレーニングデータのラベル分布を可視化
print("Distribution of digits in the training set:")
plot_label_distribution(y_train)
```

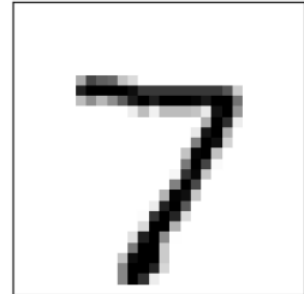
①自動で生成

②コピー

## Google Colaboratory



# もちろんもっとすごいこともできる



7 (7)

## ChatGPT 4o

“MNISTの手書き数字画像データセットを使って、他の数字画像を認識できるようなAIを作ってください。”

MNISTデータセットを使用して、手書き数字を認識するAIモデルを構築するについて説明します。ここでは、TensorFlowとKerasを使用して、\*\*畳み込みニューラルネットワーク（CNN）\*\*を構築し、手書き数字を認識できるAIを構築します。

### ステップ概要

1. データのロードと前処理
2. モデルの構築
3. モデルのトレーニング
4. モデルの評価
5. 新しい数字の予測

以下に具体的なコードを示します。

#### 1. データのロードと前処理

```
python
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np

# MNISTデータセットを読み込む
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# データの前処理（画像を0-1の範囲に正規化）
x_train = x_train / 255.0
x_test = x_test / 255.0

# 画像の形状を4次元に変換（サンプル数、高さ、幅、チャンネル）
x_train = x_train.reshape((x_train.shape[0], 28, 28, 1))
x_test = x_test.reshape((x_test.shape[0], 28, 28, 1))
```

#### 2. モデルの構築

次に、畳み込みニューラルネットワーク（CNN）モデルを構築します。CNNは、画像処理に適したモデル構造です。

```
python
# CNNモデルを構築
model = models.Sequential()
```

①自動で生成

②コピー

## Google Colaboratory

③ボタン押せば実行結果が

The screenshot shows the Google Colaboratory interface with the following code and output:

```
# 1. データのロードと前処理
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train / 255.0
x_test = x_test / 255.0
x_train = x_train.reshape((x_train.shape[0], 28, 28, 1))
x_test = x_test.reshape((x_test.shape[0], 28, 28, 1))

# 2. モデルの構築
model = models.Sequential()
model.add(layers.Conv2D(10, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(10, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(10, activation='softmax'))

# 3. モデルのトレーニング
history = model.fit(x_train, y_train, epochs=10, batch_size=64, validation_data=(x_test, y_test))

# 4. モデルの評価
predictions = model.predict(x_test)
accuracy = np.mean(np.argmax(predictions, axis=-1) == np.argmax(y_test, axis=-1))
print('Accuracy: %.2f' % accuracy)
```

The output shows the model's performance on the test set, with an accuracy of approximately 97.7%.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 10, 10, 10)	100
max_pooling2d (MaxPooling2D)	(None, 5, 5, 10)	0
conv2d_1 (Conv2D)	(None, 5, 5, 10)	100
flatten (Flatten)	(None, 250)	0
dense (Dense)	(None, 10)	2600

Total params: 2700, Trainable params: 2700, Non-trainable params: 0

Epoch 1/10: 6.00s/step - accuracy: 0.9770 - loss: 0.0425 - val\_accuracy: 0.9770 - val\_loss: 0.0425

Epoch 2/10: 6.00s/step - accuracy: 0.9770 - loss: 0.0425 - val\_accuracy: 0.9770 - val\_loss: 0.0425

Epoch 3/10: 6.00s/step - accuracy: 0.9770 - loss: 0.0425 - val\_accuracy: 0.9770 - val\_loss: 0.0425

Epoch 4/10: 6.00s/step - accuracy: 0.9770 - loss: 0.0425 - val\_accuracy: 0.9770 - val\_loss: 0.0425

Epoch 5/10: 6.00s/step - accuracy: 0.9770 - loss: 0.0425 - val\_accuracy: 0.9770 - val\_loss: 0.0425

Epoch 6/10: 6.00s/step - accuracy: 0.9770 - loss: 0.0425 - val\_accuracy: 0.9770 - val\_loss: 0.0425

Epoch 7/10: 6.00s/step - accuracy: 0.9770 - loss: 0.0425 - val\_accuracy: 0.9770 - val\_loss: 0.0425

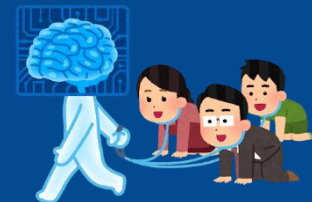
Epoch 8/10: 6.00s/step - accuracy: 0.9770 - loss: 0.0425 - val\_accuracy: 0.9770 - val\_loss: 0.0425

Epoch 9/10: 6.00s/step - accuracy: 0.9770 - loss: 0.0425 - val\_accuracy: 0.9770 - val\_loss: 0.0425

Epoch 10/10: 6.00s/step - accuracy: 0.9770 - loss: 0.0425 - val\_accuracy: 0.9770 - val\_loss: 0.0425

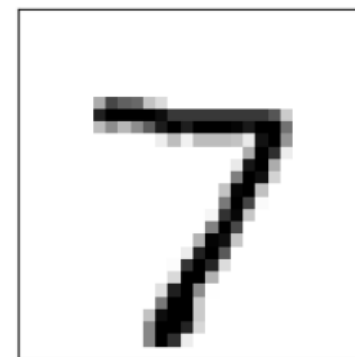
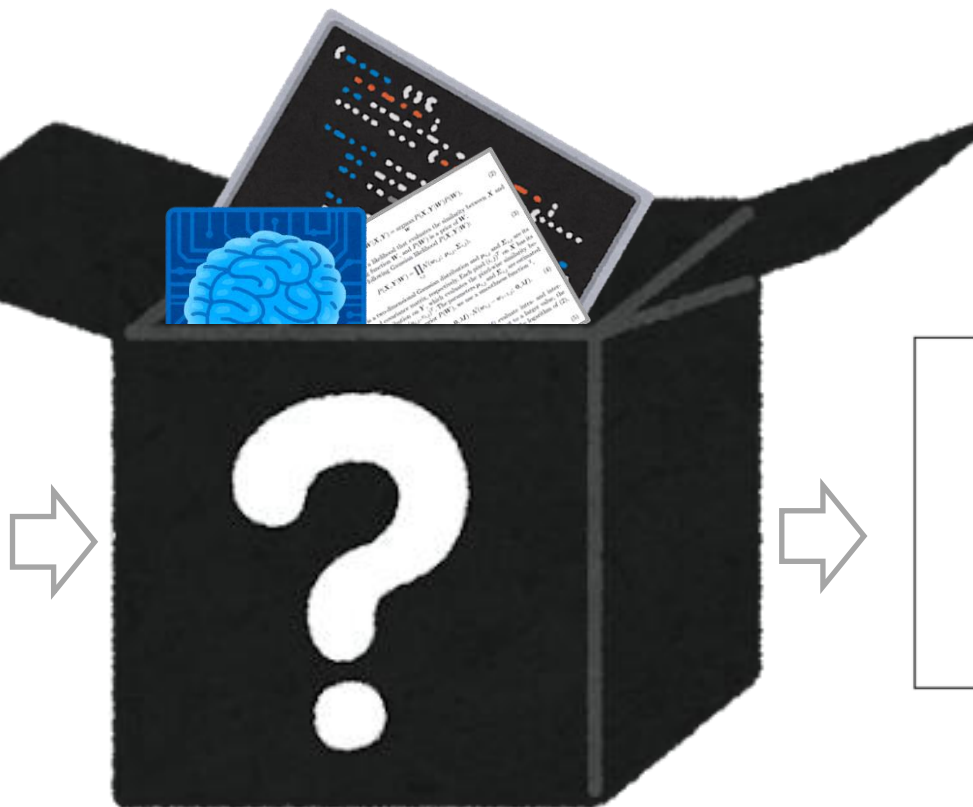
97.70%

# なんだか分からないが「できちゃう」



“MNISTの手書き数字画像データセットを使って、他の数字画像を認識できるようなAIを作ってください。”

したいこと



7 (7)

できた!

よくわからない  
「数式」「プログラム」  
「AIによる支援」

こんな時代だからこそ、  
自分の身は自分で守らないと！



なにがどのように処理されて  
いまの結果が求まっているか？

それは本当に自分が  
望んだ処理になっているのか？

間違っているとすれば  
どのように修正すればいいのか？  
わかる人材にならないと！

# 3つの科目

- データサイエンス概論I&II
  - 講義. データ解析の(基本中の)基本技術
- データサイエンス実践I～IV
  - 概論の一部復習 + 統計処理・画像処理実践準備
  - そして講義内容に沿ったプログラム演習
- データサイエンス発展I&II
  - 各学生の研究課題のための個別サポート！

# 九州大学

## 「データサイエンス概論」について

- 全分野の学生さんに，データ解析の基本的な考え方をわかってもらうための講座
  - 「卒論・修論・D論で急に先生に『データ解析しとけよ』と言われた」
  - 「ベクトル？ 内積？ なにそれおいしいもの？」
  - 「主成分分析，意味わかんない(でも使ってる)」
- 数学的に厳密な議論よりも，直観で分かってもらうことを最優先！
  - 「データ解析って，そんなに恐ろしくない」
    - 「各解析手法も，その目的・意味はそんなに難しくないかも」
    - 「ならば，自分のデータのためには，どんなデータ解析をやるべきだろうか？」
  - 興味を持てれば，そこから厳密なところを深く学んでほしい！
- 余談とか参考とか書いているスライドは，飛ばして構いません



# データサイエンス概論の内容

- データとデータ分析
- データのベクトル表現と集合
- 平均・分散・相関
- データ間の距離と類似度
- クラスタリングと異常検出
- 線形代数に基づくデータ解析の基礎
- 主成分分析
- 予測と回帰分析
- 可視化
- 確率と確率分布
- 信頼区間と統計的検定
- データ収集とバイアス
- 非構造化データ解析
- パターン認識とニューラルネットワーク
- 人工知能入門