Naive Bayes を用いた手書き数字の Arduino 上での推論

,

2025 年 6 月 18 日
machaponn

,

1 はじめに

このプログラムは,作者が大学の学部時代に自由テーマ実験の一環で Arduino Uno R3 を用いて新規性のある物を作れというテーマが与えられた. 作者は手書き数字の認識の推論を Arduino Uno R3 で行うことを目標として実験を行った. このときのプログラムをリポジトリに示している.

ここでは,当時提出したレポートの一部を抜粋しどのような理論で実装したか,成果物の性能について述べる. 理論は Naive Bayes という教師データありの機械学習アルゴリズムを用いている. 結果として,認識性能は 60.3% となっている. 下記に詳しく述べているが学部時代の物であるため正しい理解の元にドキュメントを書いているとは限らないので鵜呑みにしないで欲しい.

使い方については Processing と Arduino の使い方等を調べて欲しい.ここには記載しない.

2 実装について

2.1 Naive Bayes

ところで,ニューラルネットワークでの学習をする場合,入力層,隠れ層,出力層が必要であり. 最低でも 3 層必要である. そのため,16×16 の二値化した重みを 8bit の 1 次配列で格納したとしても 65KB 超える. よって,Arduino Uno R3 のメモリは 32KB であるため,メモリ不足である. ここで,MNIST データセットを 2 値化し 16×16 リサイズし,ナイーブベイズ分類器で実装したところ 14KB であった. 今回の実験では Naive Bayes という機械学習アルゴリズムを採用した.

Naive Bayes は,ベイズの定理に基づいた確率的分類アルゴリズムである. このアルゴリズムは,特徴量の独立性を仮定しており各特徴量がクラスに対して独立しているとみなす. これは非常に簡単で効率的な分類方法であり,特にテキスト分類などの高次元データに適している. C_kを各クラスのラベルとし,xを特徴ベクトルとするときベイズの定理の数式を(1) 式に示す.

$$P(C_k|x) = \frac{P(x|C_k) \times P(C_k)}{P(x)} \tag{1}$$

(1) 式の$P(C_k|x)$とは,特徴ベクトルxが与えられたときのクラスC_kに属する確率（事後確率）である.

また，$P(x|C_k)$はクラスC_k与えられたときの特徴ベクトルxが観測される確率(尤度)である. 事前確率$P(C_k)$は,クラスが全データ中にどれだけ出現するかを示す確率である. これは,訓練データセットにおける各クラスの割合に基づいて計算される. 手書き数字認識の場合では,各数字(0 から 9 まで)の出現頻度に基づいて計算される. 尤度は,クラス C_kが与えられたとき,

特徴ベクトル$\boldsymbol{x}$が観測される確率を示す. よって,特徴量の独立性を仮定するため,尤度は次の (2) 式となる.

$$P(C_k|x) = \prod_{i=1}^{\pi} P(x_i|C_k) \tag{2}$$

ここで, x_iは特徴ベクトル$\boldsymbol{x}$の各成分を表す. 実際の計算では,対数尤度を使用して数値の桁落ちを防ぎ計算の安定性を向上させる対数を取ることで, (2) 式であらわされた積の計算が和の計算に変換され,次の (3) 式になる.

$$\log P(C_k|x) \propto \log P(C_k) + \sum_{i=1}^{n} \log P(x_i|C_k) \tag{3}$$

(3) 式より,与えられた特徴ベクトル$\boldsymbol{x}$が最も属すると考えられるクラスC_kが推定される.

2.2 システムについて

この実験では,学習を Python を用いて機械学習を行い学習結果(重み)を Arduino に保存し推論を行った. これらのシステムを以下に示す.

2.2.1 Python による機械学習

mnist_pre.py には前処理のプロブラム,nvbs_lern.py では学習のプログラム,npy_to_h.py は学習結果を C++で読み込める配列に変換プログラムである. mnist_pre.py では,MNIST データセットを 16×16 にリサイズし 2 値化処理を行い numpy 配列に変換し. npy ファイルで出力する. nvbs_lern.py ではライブラリを用いてナイーブベイズ分類器の計算を行い,事前確率の対数と特徴ベクトル対数確率を.npy ファイルに書き込む. npy_to_h.py では学習結果が格納された npy ファイルをヘッダファイルに変換するプログラムである.

2.2.2 Arduino の実装

sketch_naive_bayes.ino に Arduino のプログラムである. また, 図 1 にフローチャートを示す. 図 1 よりシリアルポートからの通信を受けるとデータを 配列に格納する. さらに,”\n” を受信したらデータの格納をやめ推論を始める. 推論に関しては (3) 式を実装しコード上にコメントアウトしている. また, (3) 式(の計算が-INIFINITY を超えた ときに結果が出力される.

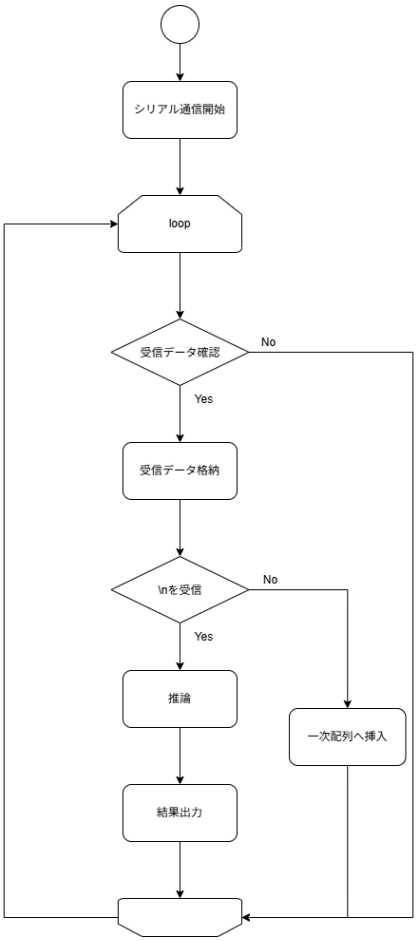

図 1: Arduino のフローチャート

2.2.3 Processing(Java)の実装

image_input.pde に手書きの数字を送信するプログラムを示した. また, Processing でに,手書き数字を送信するシステムのステートチャート図を示す. 図 2 より,手書き数字を書きスペースを押すと二値化処理が行われ bit データとして,Arduino とポート通信を行う. “\n” をエンドデータとして送信する. 図 2 を image_input.pde に実装した.

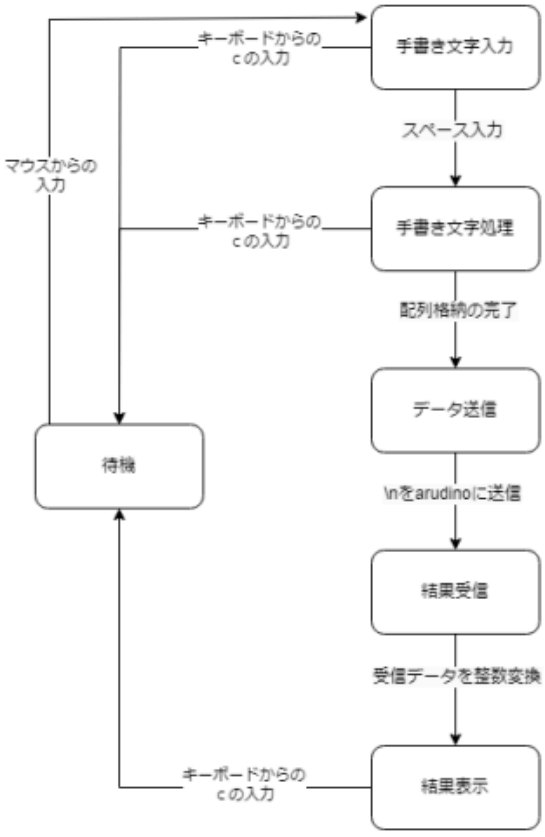

図 2: MNIST データ送信システムのステートチャート図

2.2.4 システム全体につて

ここで, 図 3 にはシステム全体像をしめした. 図 3 より(Processing)から手書きの数字データを Arduino unoR3 に送信したのち Arduino unoR3 で推論した結果を PC に送信する構造となっている. また,システム全体でどのようになっているかユーザーシーケンス図を 図 4 に示す. 図 4 には,ユーザが Processing で書いた文字がスペースを押すことで, Arduino に送信され結果が変えてくるのをしした.

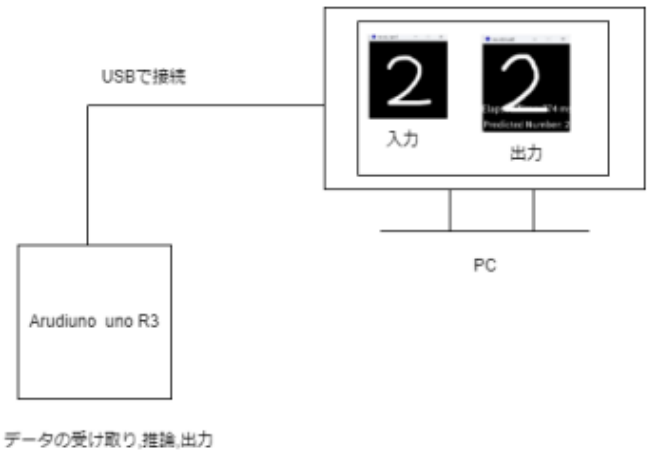

図 3: 全体像

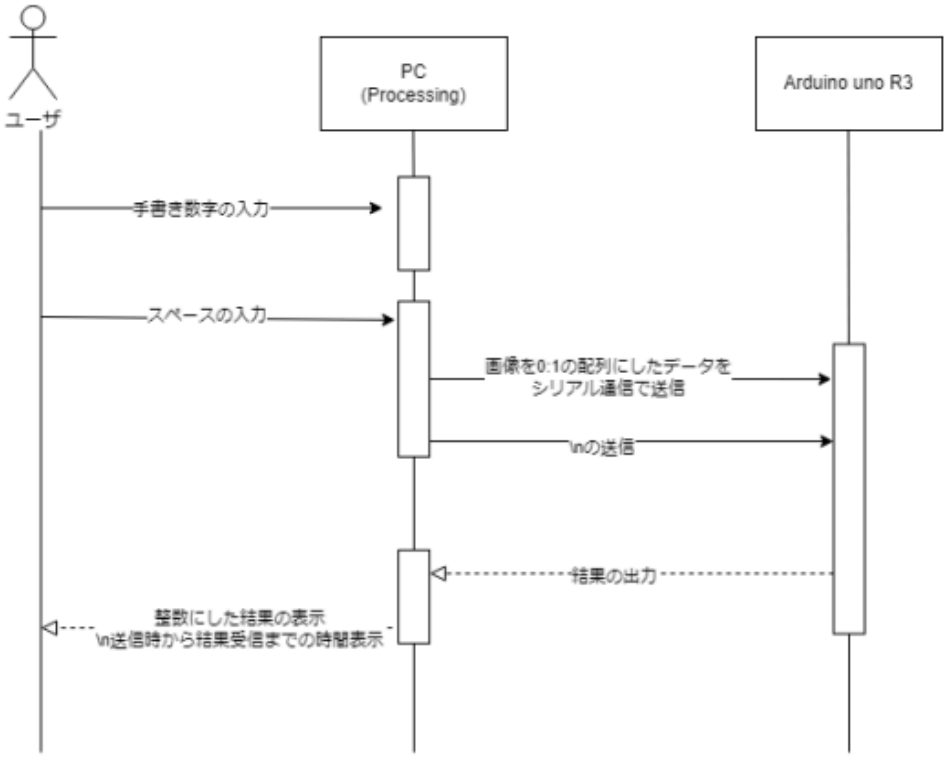

図 4: ユーザーシーケンス図

3 数字認識の性能

実装したものの精度をに表す. ここに,示したように全体の正答率は 60.3% となった.

表 1:

入力文字	正答率 %
0	56.7
1	97.7
2	90.0
3	93.3
4	93.3
5	73.3
6	76.7
7	51.3
8	20.0
9	23.3
全体	60.3