筑波大学　情報学群　情報科学類・情報メディア創成学類

令和7年度　学群編入学試験

学力試験問題（専門科目）

[注意事項]

1. 試験開始の合図があるまで、問題の中を見てはいけません。
2. 解答用紙の定められた欄に、氏名、受験番号を記入すること。
3. この問題冊子は全部で９ページ（表紙、白紙を除く）です。
4. 問題１から問題４（数学、情報基礎）の計４問をすべて答えなさい。
5. 解答用紙は、４問に対して、各問１枚の合計４枚を用いること。
6. 解答用紙上部の［　　］欄に解答する問題番号を記入すること。
7. 解答用紙の裏面を使用する場合には、その旨を解答用紙の表面に示してください。

問題１　数学（１）

次式に示される領域 D の体積を重積分を用いて求める．

$$D=\left\{(x,y,z)\,\middle|\,x^2+y^2+\frac{1}{4}z^2\le 1,\ 0\le x,\ 0\le y,\ 0\le z\right\}$$

ここで x,y,z について，次のように変数変換して考える．

$x=r\sin\theta\cos\varphi, y=r\sin\theta\sin\varphi, z=2r\cos\theta$ （$r\ge 0, 0\le\theta\le\pi, 0\le\varphi\le 2\pi$）

(1) (r,θ,φ) から (x,y,z) への変数変換のヤコビ行列 J を求めなさい．

(2) (r,θ,φ) から (x,y,z) への変数変換のヤコビアン $|J|$ を求めなさい．

(3) (2)で求めた $|J|$ を用いて，領域 D の体積を求めなさい．途中の計算も記述すること．

問題2　数学（2）

正の実数 a を含む2次関数 $f(x,y,z)=x^2+y^2+z^2+2axy+2ayz$ は実対称行列 A と実ベクトル $\boldsymbol{x}=\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ を用いて，$f(x,y,z)=\boldsymbol{x}^T A\boldsymbol{x}$ と表される．また，$f(x,y,z)$ は直交行列 B と実ベクトル $\boldsymbol{u}=\begin{pmatrix} u \\ v \\ w \end{pmatrix}$ を用いた1次変換 $\boldsymbol{x}=B\boldsymbol{u}$ によって，$g(u,v,w)=\lambda_1 u^2+\lambda_2 v^2+\lambda_3 w^2$ の形に変換される (ただし，λ_1，λ_2，λ_3 は実数とする)．このとき，以下の設問に答えなさい．

(1) 行列 A を求めなさい．

(2) 行列 A の固有値をすべて求めなさい．

(3) $f(x,y,z)=1$ が楕円面となるための a の条件を求めなさい．

(4) $|\lambda_1|$，$|\lambda_2|$，$|\lambda_3|$ のうち最も値が大きいものが $\dfrac{3}{2}$ となるとき，a および行列 B を求めなさい．

問題３　情報基礎（１）

人工知能技術の基礎となる数理モデルに、単純パーセプトロンがある。単純パーセプトロンは、以下の説明のように、入力データに対して正解値を推定して出力を得る数理モデルであり、AND などの論理演算も表現できる。

単純パーセプトロンでは、N次元の入力ベクトルデータ$\boldsymbol{x}$に対して重みベクトル$\boldsymbol{w}$との内積$\boldsymbol{w}^T\boldsymbol{x}$を計算し、バイアス$b$を加算して$z$を得る。

$$z = \boldsymbol{w}^T\boldsymbol{x} + b$$

このzを以下のステップ関数σに入力し、出力値yを得る。

$$y = \sigma(z) = \begin{cases} 1 & z > 0, \\ 0 & z \leq 0. \end{cases}$$

入力データ$\boldsymbol{x}$とその正解値gのペアが与えられたとき、入力データに対して正解値を出力するように重みベクトル$\boldsymbol{w}$とバイアスbを更新することを、単純パーセプトロンの学習と呼ぶ。また、入力データと正解値のペアを学習データと呼ぶ。例えば、入力データとして$\boldsymbol{x}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$、それぞれの正解値として$g_0 = 0$, $g_1 = 1$が与えられたとき、単純パーセプトロンの学習では$\boldsymbol{x}_0$に対して0を出力し、$\boldsymbol{x}_1$に対して1を出力するように重みベクトル$\boldsymbol{w}$とバイアスbが更新される。

M個の入力データ$\boldsymbol{x}_k (k = 0, 1, \ldots, M-1)$と正解値$g_k (k = 0, 1, \ldots, M-1)$が与えられたとき、単純パーセプトロンの学習は以下の手順で行われる：

（手順1）　現在の重みベクトル$\boldsymbol{w}$とバイアスbを使い、入力データ$\boldsymbol{x}_k$に対する出力値y_kを計算する。

（手順2）　出力値y_kと正解値g_kの差を計算し、それを誤差eとする。

（手順3）　手順 2 で計算した誤差eを用い、以下の式を用いて重みベクトル$\boldsymbol{w}$とバイアスbを更新する。

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \gamma e \boldsymbol{x}_k$$
$$b \leftarrow b + \gamma e$$

ここで、γは学習率と呼ばれる定数である。

上記の手順 1~3 をすべての学習データに対して行い、さらにそれをあらかじめ決めた回数E（エポック数と呼ぶ）だけ繰り返すことで、単純パーセプトロンの学習が完了する。

次ページへ続く

プログラム1は、単純パーセプトロンの学習をC言語で実装したものである。このプログラムについて、以下の問いに答えなさい。

(1) ステップ関数 step_function の空欄（ア）と（イ）をそれぞれ埋めなさい。

(2) 与えられたデータに対して単純パーセプトロンの出力を得る関数 predict の空欄（ウ）と（エ）をそれぞれ埋めなさい。

(3) main 関数の中の単純パーセプトロンの学習について、重みベクトルの更新式およびバイアスの更新式を空欄（オ）と（カ）にそれぞれ1文で書きなさい。

(4) このプログラムの単純パーセプトロンは、どのような論理演算を学習しているか説明しなさい。

(5) 単純パーセプトロンの学習に対する時間計算量（オーダー）について、入力データの次元数N、学習データ数M、エポック数Eを用いて説明しなさい。

（プログラム 1）

```
#include <stdio.h>

#define LEARNING_RATE 0.1  //学習率
#define NUM_EPOCHS 10      //エポック数
#define NUM_SAMPLES 4      //学習データ数
#define NUM_DIMS 2         //入力データの次元数

// 入力データ
int X_in[NUM_SAMPLES][NUM_DIMS] = {
    {0, 0}, {0, 1}, {1, 0}, {1, 1}
};
int G[NUM_SAMPLES] = {0, 1, 1, 1}; // 入力データに対する正解値
```

次ページへ続く

```
// ステップ関数
int step_function(double z) {
    if (          (ア)          ) {
        (イ)
    } else {
        return 0;
    }
}

// 単純パーセプトロン
int predict(double w[NUM_DIMS], int x[NUM_DIMS], double bias) {
    double z = 0.0;
    for (int i = 0; i < NUM_DIMS; i++) {
        (ウ)
    }
    return (エ)
}

int main() {
    double weights[NUM_DIMS] = {0.0, 0.0};
    double bias = 0.0;

    // 単純パーセプトロンの学習
    for (int epoch = 0; epoch < NUM_EPOCHS; epoch++) {
        for (int k = 0; k < NUM_SAMPLES; k++){
            int y_pred = predict( weights, X_in[k], bias );
            double error = G[k] - y_pred;
            for (int i = 0; i < NUM_DIMS; i++) {
                (オ)
            }
            (カ)
        }
    }

    return 0;
}
```

問題４　情報基礎（２）

値の大小関係の比較に基づかずに，整数のデータを整列させる 2 つのアルゴリズム（バケットソートと基数ソート）に関する文章を読んで問いに答えなさい．

バケットソートは整列対象となるデータの個数とデータの取りうる値の範囲があらかじめ定まっていることを前提とした整列アルゴリズムであり，値同士の大小比較は一切行われない．まずデータが取りうる値の種類の数と同数の空のバケット（バケツ，器）を用意し，それぞれのバケットにデータの値に対応する名前を付けておく．次に，整列対象となるデータを 1 つずつ順に，対応する名前のバケットに格納する．全てのデータを格納し終えたら，小さい値に対応するバケットから順番に中身を取り出せば，昇順に並べられた整数データを得ることができる．

以下の関数 bucket_sort は，上述のバケットソートのアルゴリズムを C 言語で書いたものである．関数 bucket_sort は，整数データの入った配列 a と配列 a の要素数 n を受け取り，a の要素をバケットソートによって昇順に並べ替える．そして，同じ値のデータが複数存在する場合でも正しく動作するように，バケットはキューで実装されている．このプログラムに関して以下の問いに答えなさい．配列 a の要素は 0 以上 MAX_VAL 以下の整数とする．

(1) プログラム中の空欄［　（ア）　］～［　（オ）　］を埋めて関数 bucket_sort を完成させなさい．

(2) 同じ値のデータが複数存在する場合に，整列前と整列後で，その並んでいる順番が保存されるようなアルゴリズムを安定な整列アルゴリズムと呼ぶ．例えば，3a，2，3b，1 の整数（3 を区別するため便宜上 a, b の添字を付けている）を安定な整列アルゴリズムで昇順に並び替えると，1，2，3a，3b となり，3 の並び順が整列前と整列後に変化しない．このバケットソートは安定な整列アルゴリズムであるか否かについて，理由とともに答えなさい．

次ページに続く

```
#define MAX_VAL 100
typedef struct queue {
  int *buffer;
  int length;
  int front;
  int rear;
} Queue;
void q_init(Queue *this_queue, int len) {
  this_queue->buffer = (int*)malloc(sizeof(int) * len);
  this_queue->length = len;
  this_queue->front = 0;
  this_queue->rear = 0;
}
void q_enq(Queue *this_queue, int d) {
  this_queue->buffer[      (ア)      ] = d;
  this_queue->rear = (this_queue->rear + 1) % this_queue->length;
}
int q_deq(Queue *this_queue) {
  int x = this_queue->buffer[      (イ)      ];
  this_queue->front = (this_queue->front + 1) % this_queue->length;
  return x;
}
int q_is_empty(Queue *this_queue) {
  if (                    (ウ)                    ) {
    return 1;
  } else {
    return 0;
  }
}
void q_remove(Queue *this_queue) {
  free(this_queue->buffer);
}
void bucket_sort(int a[], int n) {
  Queue *b = (Queue *)malloc(sizeof(Queue) * (MAX_VAL + 1));
  int i;
  for (i = 0; i <= MAX_VAL; i++) {
    q_init(&b[i], n + 1);
  }
  for (i = 0; i < n; i++) {
      (エ)      ;
  }
  int j = 0;
  for (i = 0; i <= MAX_VAL; i++) {
    while (q_is_empty(&b[i]) != 1) {
      (オ)      ;
      j++;
    }
  }
  // Queue array deleted
  for (i = 0; i <= MAX_VAL; i++) {
    q_remove(&b[i]);
  }
  free(b);
}
```

次ページに続く

基数ソートは下の桁から順番にバケットソートを適用することによって，最終的に整列されたデータを得る整列アルゴリズムである．基数ソートもバケットソートと同様に整列対象となるデータの個数とデータの取りうる値の範囲があらかじめ定まっていることを前提とした，値同士の大小比較は一切行わない整列アルゴリズムであるが，バケットソートよりもバケットの数を少数に抑えることができる点に特徴がある．k 桁の 10 進数，すなわち，0 以上 10^k-1 以下の整数を対象とした場合，バケットソートでは 10^k 個のバケットが必要になるが，基数ソートでは，10 個のバケットのみを用い，下の桁から順番に桁ごとにバケットソートを行う操作を k 回繰り返すことによって，全データの整列が完了する．例えば 21，33，12 の整数を基数ソートによって昇順に並べ替える場合，まず一の位でバケットソートをするので並び順は 21，12，33 となり，その後に十の位でバケットソートをすることによって整列済みの整数データ 12，21，33 が得られる．

以下の関数 radix_sort は，上述の基数ソートのアルゴリズムを C 言語で書いたものである．関数 radix_sort は 10 進数の整数データの入った配列 a，配列 a の要素数 n，整数データの最大桁数 k を受け取り，0 以上 10^k-1 以下の整数である a の要素を基数ソートによって昇順に並べ替える．このプログラムに関して以下の問いに答えなさい．

(3) プログラム中の空欄［　（カ）　］～［　（ケ）　］を埋めて関数 radix_sort を完成させなさい．

(4) 配列 a の要素が 143，322，246，755，123，563，514，522 である場合，関数 radix_sort からの標準出力を書きなさい．

(5) 配列 a の要素数 n が大きく，かつ n 個のデータが全て異なる整数であるとき $k \fallingdotseq \log_{10} n$ と表せる．このときのプログラムの時間計算量がどのようになるかを考えてみる．

基数ソートはバケットソートを k 桁分実行するアルゴリズムであるので，バケットソートの時間計算量をまず考える．バケットソートは，バケットの初期化のための O(バケットの数)の処理，データをバケットに入れるための O(n)の処理，そしてデータを全てのバケットから取り出し配列に戻すための O(バケットの数＋n)の処理から成る．したがって，バケットソートの時間計算量は全体で O(バケットの数＋n)となる．

次ページに続く

ここで,バケットの数は10進数の整数データを対象としているために10個であり，配列aの要素数nを大きくすれば 10 << n となるので，バケットソートの計算量はO(n)と見なせる.

これらを踏まえ，nを用いて，基数ソートの時間計算量を理由とともに答えなさい.

```
void radix_sort(int a[], int n, int k) {
  // 10進数なので，10個のバケットを用意
  Queue *b = (Queue *)malloc(sizeof(Queue) * 10);
  int i, j, m;
  for (i = 0; i < 10; i++) {
    q_init(&b[i], n + 1);
  }
  // 下の桁から順番に桁ごとにバケットソートを行うという操作をk回繰り返す
  for (i = 1, m = 1; i <= k; i++, m *= 10) {
    int idx_d;  // 配列aの要素のインデックス
    int idx_q;  // バケットのインデックス
    for (idx_d = 0; idx_d < n; idx_d++) {
      int radix = (      (カ)      ) % 10;
      (キ)                          ;
    }
    for (idx_d = 0, idx_q = 0; idx_q < 10; idx_q++) {
      while (                (ク)                ) {
        (ケ)                          ;
        idx_d++;
      }
    }
    // 配列aの中身を標準出力
    for (j = 0; j < n; j++) {
      printf("%d ", a[j]);
    }
    printf("¥n");
  }
  // Queue array deleted
  for (i = 0; i < 10; i++) {
    q_remove(&b[i]);
  }
  free(b);
}
```