1　乱数　214 頁

1.1　学習のポイント

乱数を発生させるための標準関数について学びます。

乱数を発生させるための標準関数として以下のものがあります。

関数	機能	引数の型	関数の型
rand()	0 から 32767(int 型の正の最大) の整数乱数を発生	void	int
srand()	乱数列の開始点を seed でセット	unsigned	void

これらの関数は、stdlib.h でプロトタイプ宣言されています。

rand() が実行されると 0 から 32767 の範囲の整数乱数が 1 個返されます。
0 から 1.0 の実数乱数を作るには次のようにします。

```
(double)rand()/32767.0
```

srand(seed) は、seed で乱数列の開始点をセットします。seed は、0 から 65535 の任意の整数を与えます。

1.2　例題 47 reidai47.c

rand を用いて 1 から 6 の乱数 (サイコロの目) を作りなさい。

```
/*
  rand を用いて 1 から 6 の乱数（サイコロの目）を作りなさい。
  reidai47.c
*/

#include <stdio.h>
#include <stdlib.h>

int main()
{
  int i;
  int x;
```

```
  for(i = 0; i < 20; i++){
    x = (int)(6 * (rand()/ 32767.1) + 1);
    printf("%d ", x);
  }

  return 0;
}
```

1.3 練習問題 47 rensyu47.c

2 つのサイコロをふったときに出る目の和の分布をヒストグラムにする。サイコロは 5000 回ふるものとする。

```
/*
  練習問題 47
  2 つのサイコロをふったときに出る目の和の分布をヒストグラムにする。
  サイコロは 5000 回ふるものとする。
  rensyu47.c
*/

#include <stdio.h>
#include <stdlib.h>

int main()
{
  static int hist[13] = {0,0,0,0,0,0,0,0,0,0,0,0,0};
  int        j, k, x, y;

  for(j = 1; j <= 5000; j++){
    x = (int)(6 * (rand()/32767.1) + 1);
    y = (int)(6 * (rand()/32767.1) + 1);
    hist[x + y]++;
  }

  for(j = 2; j <= 12; j++){
    printf("%2d  : ", j);
    for(k = 1; k <= hist[j] / 20; k++){
      printf("*");
    }
```

```
      printf("\n");
    }

    return 0;
  }
```

2 AI による大規模データ処理入門

2.1 言語処理アルゴリズム

自然言語で記述された大規模データから、その特徴を抽出する方法を考えます。人工知能の歴史においては、自然言語処理の研究は、機械翻訳や自動要約などへの応用を念頭に置いて、さまざまな側面から取り組まれてきました。

2.2 n-gram

自然言語処理の基礎　　コロナ社　奥村学

文字を単位として、頻度を数えてみるとなにがわかるだろうか?
1 文字ではあまり有用な情報は、得られないかもしれない。では、2 文字の連続、3 文字の連続について、その頻度を数えてみてはどうだろうか?　　このように隣接する n 単位の共起 (n-gram) についてコーパスを基に統計モデルを考えると面白いことがわかる。

一般に、n が大きいほど、大量のテキストからでないと信頼性の高い確立は計算できないので n-gram の n は、2,3 とすることが多い。

通常、n=2,3 の場合をそれぞれ、バイグラム (bigram)、トライグラム (trigram) と呼ぶ。英語の 2 文字の連続としては、'th','he','in','an','er' などが高頻度であり、'q' の次は確実に'u' が続くことがわかる。このように文字を単位として構成されている「単語」というものが、文字のならびに対してある種の傾向を与える様子が見て取れる。

2.3 n-gram による特徴抽出

自然言語で表現された文書の表層的な特徴を抽出することで、文書の特徴を表現します。

記号列の特徴を調べるために、同じような部分記号列を n 個の記号の並びとして切り出し、同じ n 個の記号の並びが繰り返し出現しているかどうかを数え上げます。このとき、n 個の記号の並びを n-gram と呼びます。
例では、英文の入力に対して、n=5 として 5-gram を作成していきます。例のように、5 文字の記号の並びを先頭から作成していき、その頻度を調べます。

解析対象文

```
Alice was beginning to  (5 文字の記号の並び (5-gram) を
                         解析対象文の先頭から作成していく。)
Alice
lice
ice w
ce wa
e was
 was
```

2.4 n-gram を作成するプログラム ngram.c

```
/************************************/
/*     n-gram の作成                */
/*        ngram.c                   */
/************************************/

#include <stdio.h>
#define N 5 /*n-gram の長さ n*/

/*関数のプロトタイプの宣言 */
void initngram(char ngram[]) ;
            /*n-gram の初期化*/
void putngram(char chr,char ngram[]) ;
         /* n-gram の出力       */

/**********************/
/*     main() 関数     */
/**********************/
int main()
{
```

```
 char chr  ;/*入力文字*/
 char ngram[N+1];/*n-gram*/

 /*n-gram の初期化*/
 initngram(ngram) ;

 /*n-gram の出力*/
 while((chr=getchar())!=EOF){
  putngram(chr,ngram) ;/*出力*/
 }

 return 0 ;
}
/**********************/
/* putngram() 関数     */
/* n-gram の出力       */
/**********************/
void putngram(char chr,char ngram[])
{
 int i ;

 for(i=0;i<N-1;++i)
  ngram[i]=ngram[i+1] ;
 ngram[N-1]=chr ;

 printf("%s\n",ngram) ;
}

/**********************/
/* initngram() 関数    */
/* n-gram の初期化     */
/**********************/
void initngram(char ngram[])
{
 int i ;

 for(i=0;i<N;++i)
  ngram[i]=' ' ;

 ngram[N]='\0' ;/*文字列の終わり*/
```

```
}
```

2.5 ngram.exe の使い方

```
./ngram < alice.txt
```