パターン認識入門

パターン認識

- 音や画像に中に隠れた
  パターンを認識する。
  - 音素・音節・単語・文・・・
  - 基本図形・文字・指紋・物体・人物・顔・・・
- 「パターン」は唯一のデータではなく、
  似通ったデータの集まりを表している。
  - 多様性
  - ノイズ
- 「等しい」から「似ている」へ
- 「～だ」から「～らしい」へ

「等しい」から「似ている」へ

- 完全に等しいかどうかではなく、
  「似ているか」どうかを判定する。
  - パターンを代表する模範的データと
    どのくらい似ているか。
- 例：二つの文字列の比較
  - 多少の文字の食い違いや、
    文字の欠落を許して、似ているか。
  - アラインメントという。
- 動的計画法を活用。

DNAやタンパクの比較

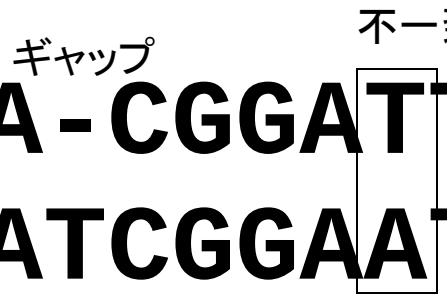
- DNA --- 塩基（四種類）の配列
  - セントラル・ドグマ

    DNA → RNA → タンパク
  - 三塩基（コドン）が一つのアミノ酸に
  - 似ている配列は似ているタンパクに
  - 似ている配列は同じ祖先から進化
- タンパク --- アミノ酸（２０種類）の配列
  - 似ている配列は似た構造に
  - 似ている配列は似た機能を

アラインメント

- アラインメント
  二つ(複数)の文字列の比較
  - 音声認識・文字認識
  - DNAやタンパクの比較

GACGGATTAG と GATCGGAATAG



アラインメント

- ぴったり同じでなくとも、
  似ているかどうかを判定。
- スコア
  - 一致
  - 不一致
  - ギャップ

  足し合わせる → アラインメントの類似度
- 類似度が最大の
  アラインメント（ギャップの入れ方）を求める。

最適化

例

スコア：　一致　+2
不一致　−1
ギャップ　−2

AAC と ATAG をアラインするには

AA と ATA のアラインメントに C と G を付加

2　**A-A**　→　**A-AC**　1　これを採用！
ATA　**ATAG**

AAC と ATA のアラインメントに - と G を付加

0　**AAC**　→　**AAC-**　−2
ATA　**ATAG**

AA と ATAG のアラインメントに C と - を付加

0　**A-A-**　→　**A-A-C**　−2
ATAG　**ATAG-**

再帰版のアラインメント

```
def align_sub(s,t,i,j)
  if i==0 || j==0
    i*g() + j*g()
  else
    v0 = align_sub(s,t,i,  j-1) + g()
    v1 = align_sub(s,t,i-1,j-1) + q(s[i-1],t[j-1])
    v2 = align_sub(s,t,i-1,j)   + g()
    max(v0,max(v1,v2))
  end
end

def align_rec(s,t)
  align_sub(s,t,s.length(),t.length())
end
```

g : ギャップのペナルティ

q(c, d) : c と d の類似度

文字列 s の長さ

文字列 t の
j-1 番目の
文字（コード）

align_rec.rb

ギャップのペナルティ

```
def g()
  -2
end

def q(c,d)
  if c == d
    return 2
  else
    return -1
  end
end
```

align_rec.rb

文字 c と文字 d の
類似度を返す

一致

不一致

いわずもがな、
x と y の最大値

```
def max(x,y)
    if x>y
        return x
    else
        return y
    end
end
```

max.rb

練習

- （max.rb を作る。）
- align_rec.rb をダウンロードし、align_rec("AAC","ATAG") を実行する。
- RNase_P.rb をダウンロードし、align_rec(seq0(),seq1()) を実行する。

動的計画法

- プロ野球日本シリーズにおける優勝の確率

  P(i, j): Aがシリーズに勝つまでにあと i 勝、
  Bはあと j 勝という状況で、
  最終的にAがシリーズに勝つ確率

$$
\begin{aligned}
P(i, j) &= 1 && i=0 \text{ かつ } j>0 \\
&= 0 && i>0 \text{ かつ } j=0 \\
&= (P(i-1, j) + P(i, j-1))/2 && i>0 \text{ かつ } j>0
\end{aligned}
$$

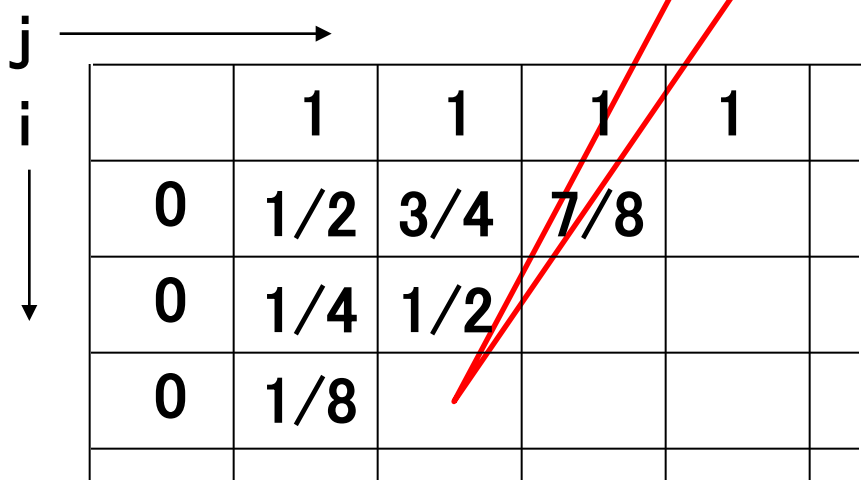
j →

i ↓

	1	1	1	1	
0	1/2	3/4	7/8		
0	1/4	1/2			
0	1/8				

AとBは
同じ強さと
仮定

1. 1/16
2. 3/16
3. 5/16
4. 7/16
5. 9/16



ここは？

動的計画法

- テーブルを用意して、
  再帰的な計算の重複を避ける。
- テーブルの中身を順に埋めることにより、
  求める値を計算する。
- 各種の最適化問題に適用。
  - アラインメント
  - DNA・RNAのエネルギー最小化
  - 構文解析

アラインメントの動的計画法

- 二つの配列 s と t の間の類似度
- a[i][j] : s の部分配列 s[0],...,s[i-1] と
  t の部分配列 t[0],...,t[j-1] の間の類似度
- a[i][j] = max{a[i][j-1] + g(),
  a[i-1][j-1] + q(s[i-1], t[j-1]),
  a[i-1][j] + g() }

  g() : ギャップのペナルティ（負の数）

  q(c, d) : c と d の類似度

  - c と d が等しければ適当なスコア（正）
  - 似ていればそれなりのスコア
  - 似ていなければ不一致のペナルティ（負）

境界条件

- a[0][0] = 0
- a[0][j] = a[0][j-1] + g() (j>0)
- a[i][0] = a[i-1][0] + g() (i>0)

例えば g() = -2

- 結局

  a[0][j] = j*g()

  a[i][0] = i*g()

- となる。要するに、ギャップばかり。

スコア:　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

	t	**A**	**T**	**A**	**G**
s	0	−2	−4	−6	−8
A	−2				
A	−4				
C	−6				

スコア:　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

	t	**A**	**T**	**A**	**G**
s	0	−2	−4	−6	−8
A	−2	2			
A	−4				
C	−6				

スコア:　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

	t	**A**	**T**	**A**	**G**
s	0	−2	−4	−6	−8
A	−2	2	0		
A	−4	0			
C	−6				

スコア:　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

	t	**A**	**T**	**A**	**G**
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

	t	**A**	**T**	**A**	**G**
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	

A-A
ATA

A-A-
ATAG

AAC
ATA

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

	t	**A**	**T**	**A**	**G**
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	1

A-A
ATA

A-A-
ATAG

AAC
ATA

A-AC
ATAG

```
def align(s,t)
  m = s.length()
  n = t.length()
  a = make2d(m+1,n+1)
  a[0][0] = 0
  for j in 1..n
    a[0][j] = a[0][j-1] + g()
  end
  for i in 1..m
    a[i][0] = a[i-1][0] + g()
  end
  for i in 1..m
    for j in 1..n
      # ここを埋めてみよう
    end
  end
  a
end
```

align.rb

境界条件

スコア：　一致　+2
不一致　−1
ギャップ　−2

ここは？

	t	**A**	**T**	**A**	**C**
s	0	−2	−4	−6	−8
A	−2				
T	−4				
C	−6				

1. 1
2. 2
3. 3
4. 4
5. 0
6. -1
7. -2
8. -3
9. -4

スコア：　一致　+2
不一致　−1
ギャップ　−2

ここは？

	t	A	T	A	C
s	0	−2	−4	−6	−8
A	−2				
T	−4				
C	−6				

1. 1
2. 2
3. 3
4. 4
5. 0
6. -1
7. -2
8. -3
9. -4

スコア：　一致　+2
不一致　−1
ギャップ　−2

	t	**A**	**T**	**A**	**C**
s	0	−2	−4	−6	−8
A	−2				
T	−4				
C	−6				

ここは？

1. 1
2. 2
3. 3
4. 4
5. 0
6. -1
7. -2
8. -3
9. -4

トレースバック

- 最大の類似度を与えるアラインメントを提示するために、配列の最後から、それまでの選択を振り返る（トレースバック）。
- ギャップの入った文字列を（配列にして）二つ返す。

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

→j

↓i

	t	A	T	A	G
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	1

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

→j

↓i

	t	A	T	A	G
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	1

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

→j

↓i

	t	A	T	A	G
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	1

A-A
ATA

A-AC
ATAG

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

→j

↓i

	t	A	T	A	G
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	1

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

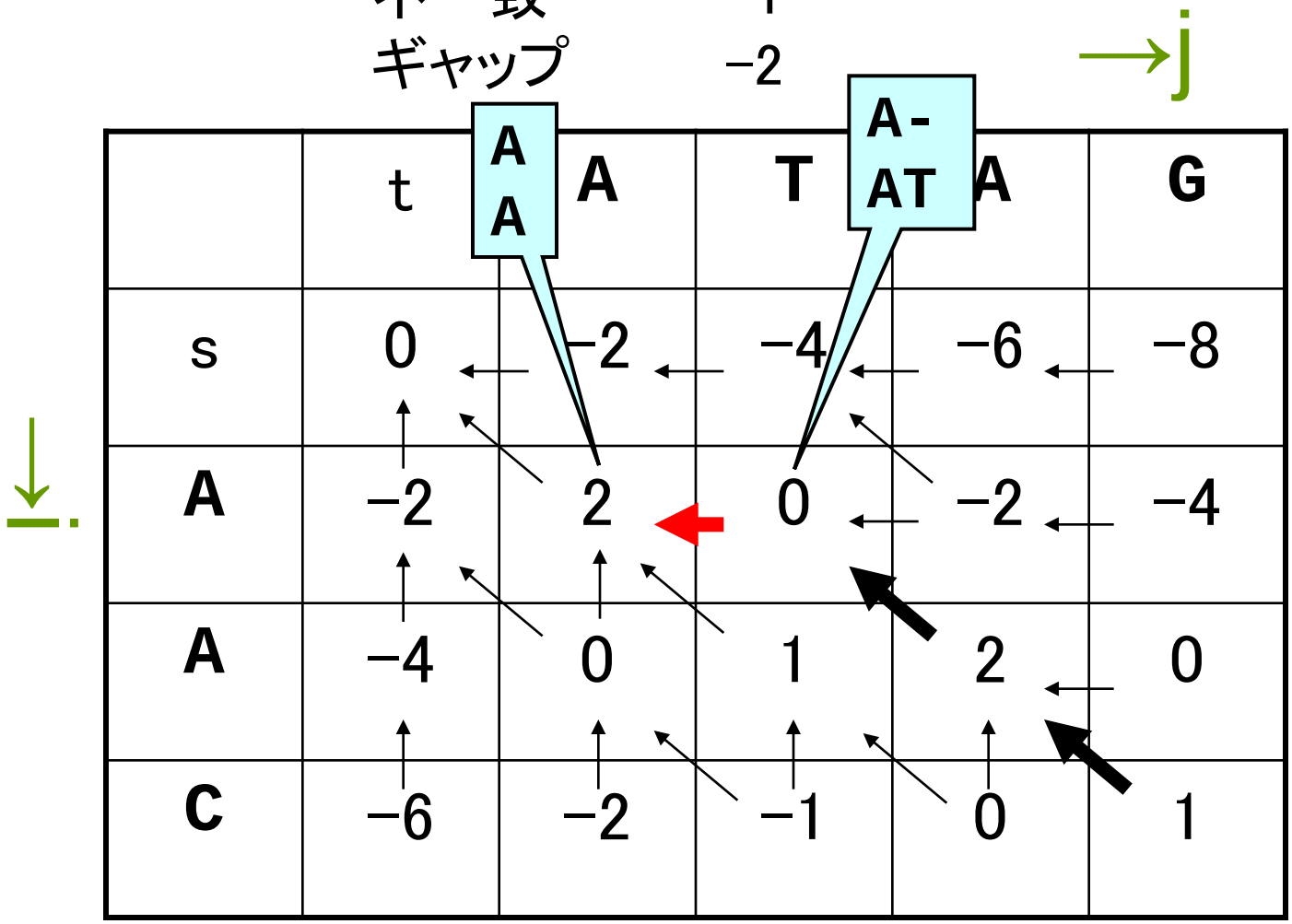
→j

↓i

	t	A	T	A	G
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	1

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG



u の前に - を追加

v の前に t[j-1] を追加

```
def traceback(a,s,t)
  u = ""
  v = ""
  i = s.length()
  j = t.length()
```

```
  while i>0 || j>0
    if j>0 && a[i][j] == a[i][j-1] + g()
      u = "-"           + u
      v = t[j-1 .. j-1] + v
      j = j - 1 # go left
    else
      if i>0 && j>0 &&
          a[i][j] == a[i-1][j-1] + q(s[i-1], t[j-1])
        # ここを埋めてみよう
      else
        if i>0 && a[i][j] == a[i-1][j] + g()
          # ここを埋めてみよう
        end
      end
    end
  end
  [u,v]
end
```

この + は文字列の連結

条件

align.rb

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

→j

↓i

	t	A	T	A	G
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	1

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

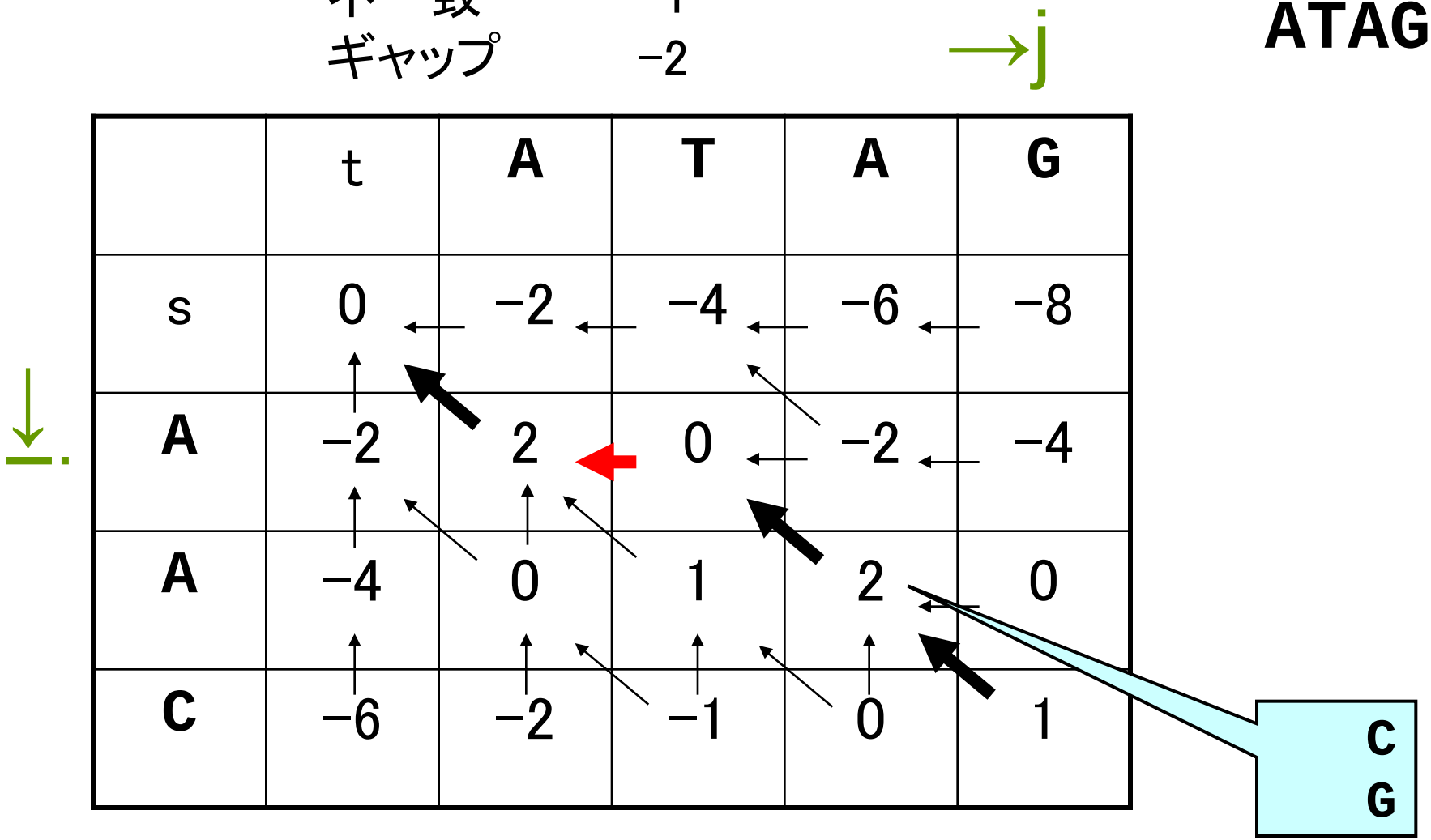
→j

↓i

	t	A	T	A	G
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	1

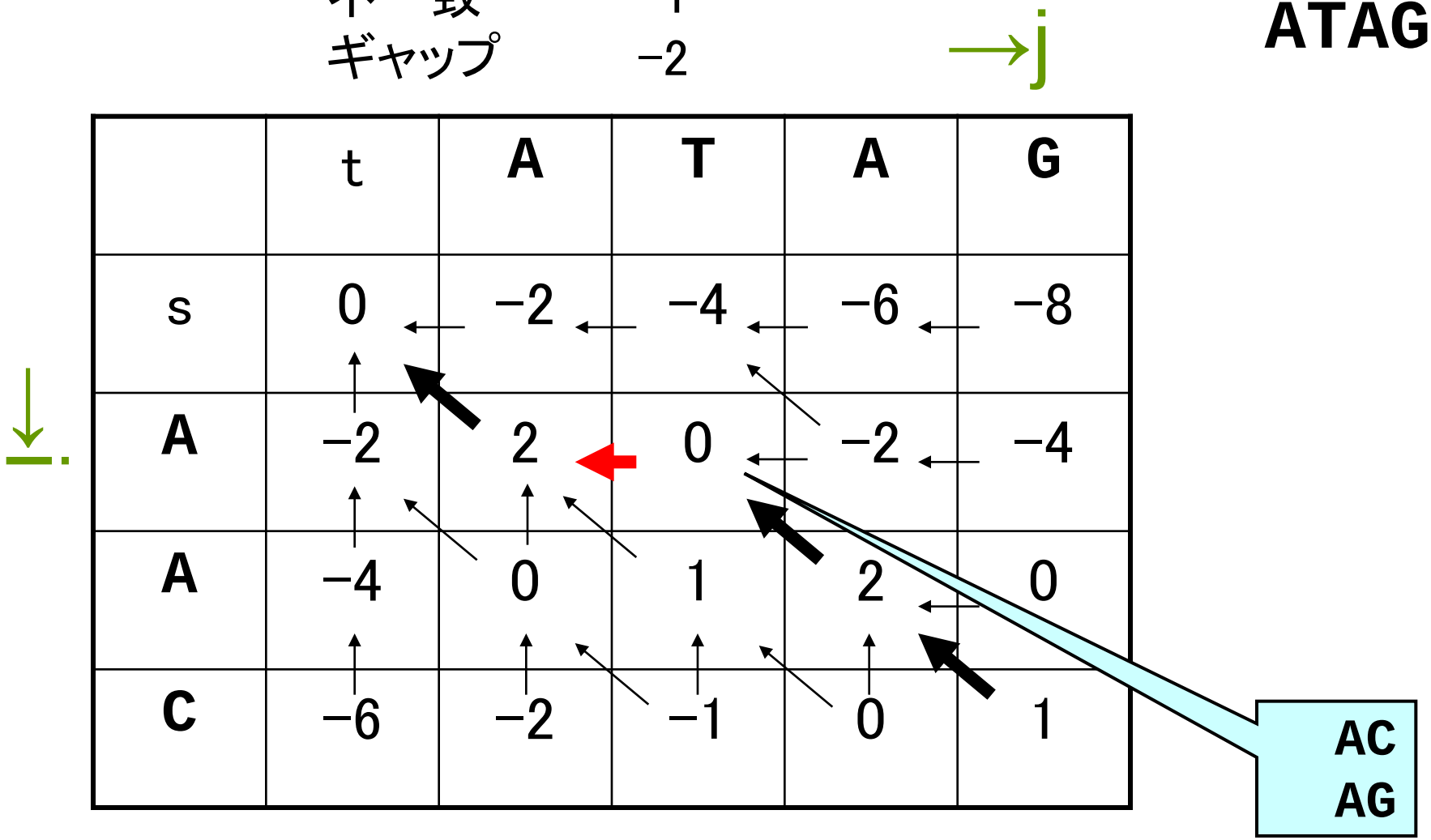
スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG



スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG



スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG

→j

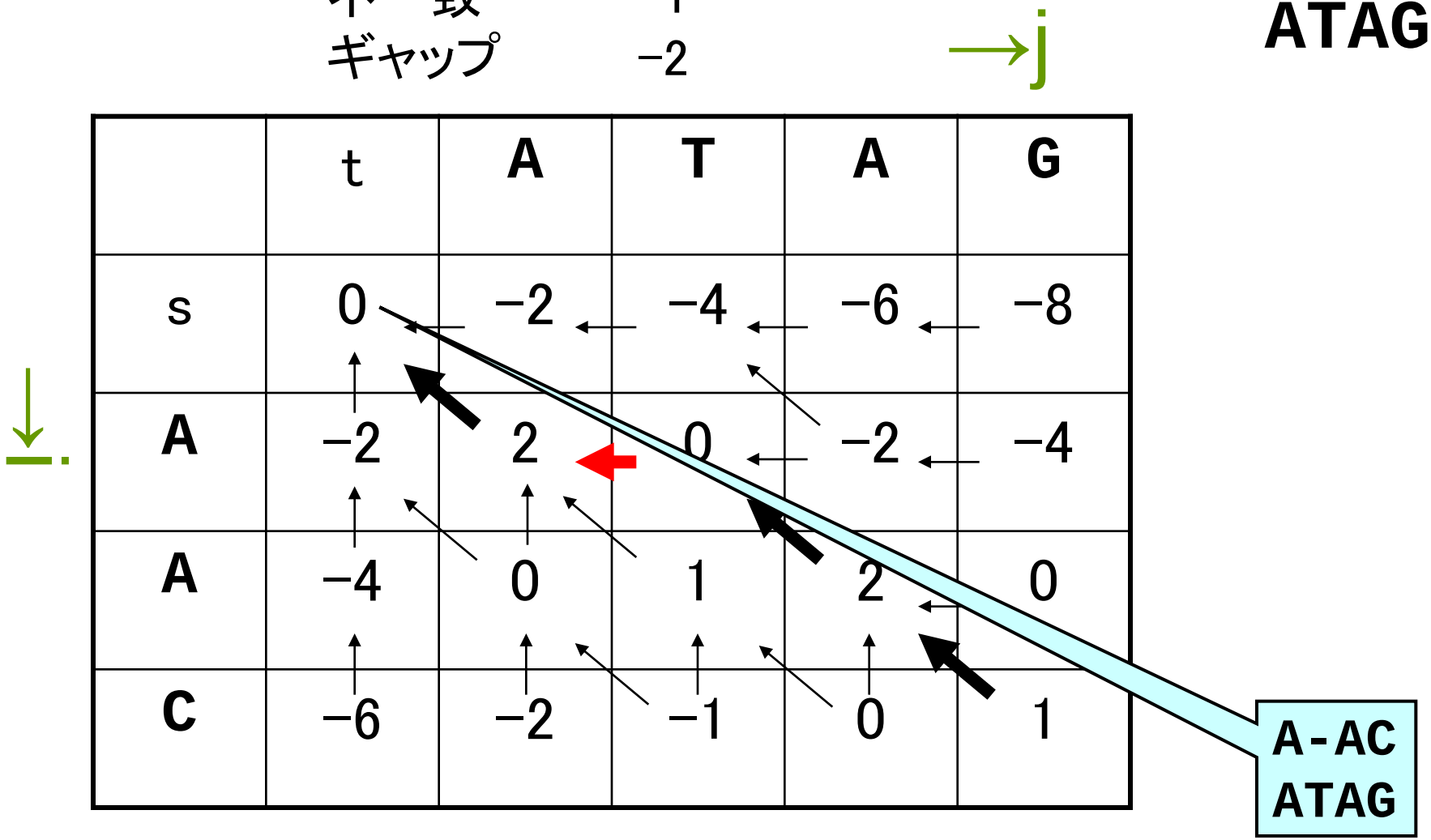
↓i

	t	A	T	A	G
s	0	−2	−4	−6	−8
A	−2	2	0	−2	−4
A	−4	0	1	2	0
C	−6	−2	−1	0	1

-AC
TAG

スコア：　一致　+2
不一致　−1
ギャップ　−2

A-AC
ATAG



スコア：　一致　+2
不一致　−1
ギャップ　−2

アラインメントは？

	t	A	T	A	C
s	0	−2	−4	−6	−8
A	−2				
T	−4				
C	−6				

1. -ATC
   ATAC
2. A-TC
   ATAC
3. AT-C
   ATAC
4. ATC-
   ATAC

練習7.4と7.5

- align.rb をダウンロードする。
- align を完成させて、
  align("AAC","ATAG") を実行する。
- align(seq0(),seq1()) を実行する。
- traceback を完成させて、
  align_dp("AAC","ATAG") を実行する。
- align_dp(seq0(),seq1()) を実行する。

進捗状況の確認

1. align_dp が正しく動いた時点で投票してください。
2. align_dp（traceback）が動かない。
3. align は動いたが、traceback がまだできていない。
4. align はできたが、うまく動かない。
5. align がまだできていない。

練習問題確認プログラム

- ex07.rb（練習7.4-5と7.7）の結果を以下のアドレスに送れ。

  is-komaba@lyon.is.s.u-tokyo.ac.jp

- Subject:欄は、

  is-komaba

  としてください。

- 〆切: 12月20日
  - 練習問題確認プログラムの結果は出席点の一部とします。結果の内容は問いません。（習得状況を知りたい。）