HMM_8

April 17, 2019

1 HMM for bioinformatics

Given the parameters of an HMM, we now know how to compute the probability that the model M emits any given sequence S, as well as how to identify the most probable path ϕ through the model whereby S could have been generated.

We turn now to the problem of constructing an HMM and setting its parameters so as to most accurately model the class of sequences in which we are interested. This is the problem of parameter estimation, or of training the HMM.

Let us suppose we wish to train a 3-state HMM from the following labeled sequence:

$$CGATATTCGATTCTACGCGCGTATACTAGCTTATCTGATC$$
$$01111111222222211111122221111111222211111 0$$

where we have given the state labels below the corresponding symbols in the sequence

1.1 Basic facts

:1. It is easy to see that we observe the number of times state label i is followed by label j and then convert this to a relative frequency by;

:2. In the case of the emission probabilities, we merely count the number of times each symbol s_i was observed in association with state q_j and then normalize this into a probability

:3. Obviously, these values are the maximum likelihood estimates for the true parameters of the HMM assumed to have generated the training data.

1.2 Stupid example

```
In [7]: genom=c("C" , "G" , "A" , "T" , "A" , "T" , "T" , "C" , "G","A" ,"T" , "T", "C","T","A
        hidden=c(1,1,1,1,1,1,1,2,2,2,2,2,2,2,1,1,1,1,1,1,2,2,2,2,1,1,1,1,1,1,2,2,2,2,1,1,1,1
        dane=list(hidden, genom)
```

1.3 Emissions

```
In [8]: names(dane)=c("hidden","genom")
        E=as.matrix(table(dane))
        E=(1/rowSums(E))*E
        E
```

```
      genom
hidden         A         C         G         T
     1 0.2400000 0.2800000 0.2000000 0.2800000
     2 0.2000000 0.2000000 0.1333333 0.4666667
```

1.4 Transitions

```
In [9]: Ptemp=matrix(0,nrow=2,ncol=2)
        for(i in 1:(length(hidden)-1)){
            Ptemp[hidden[i],hidden[i+1]]=Ptemp[hidden[i],hidden[i+1]]+1
        }
        P=matrix(0,nrow=3,ncol=3)
        P[2:3,2:3]=Ptemp
        P[1,2]=1
        P[2,1]=1/40
        P=(1/rowSums(P))*P
        P
```

0.000000000	1.0000000	0.0000000
0.001040583	0.8740895	0.1248699
0.000000000	0.2000000	0.8000000

```
In [1]: source("hmm.R")
        H=c("q_0","q_1","q_2")
        O=c("E","I")
```

```
In [4]: library("igraph")
        result <- graph_from_adjacency_matrix(P,mode="directed",weighted = TRUE)
        plot.igraph(result,vertex.label=H)
```

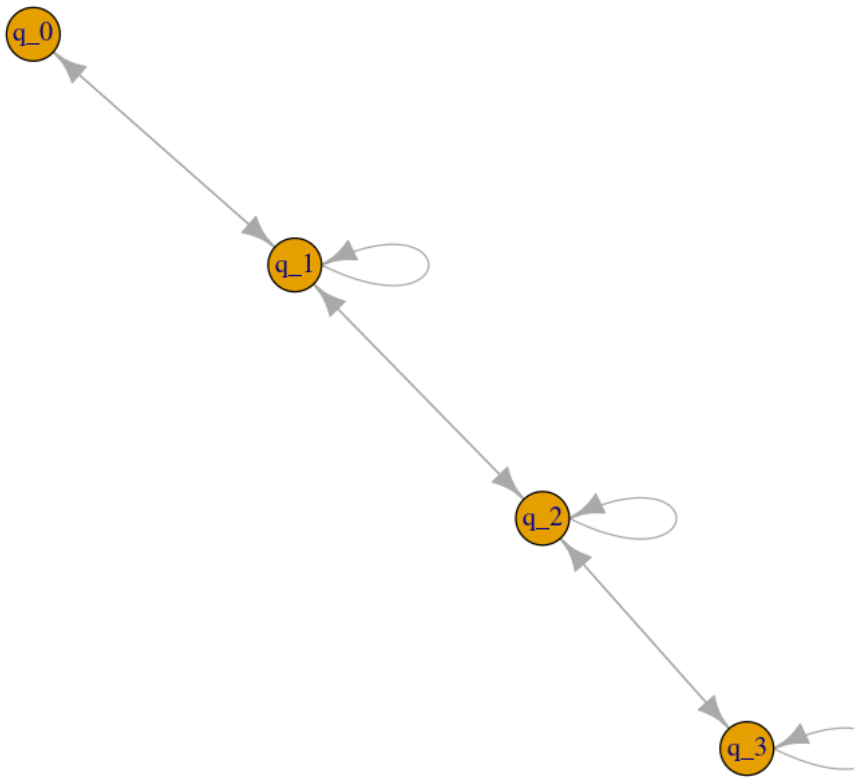
```
Attaching package: igraph

The following objects are masked from package:stats:

    decompose, spectrum

The following object is masked from package:base:

    union
```



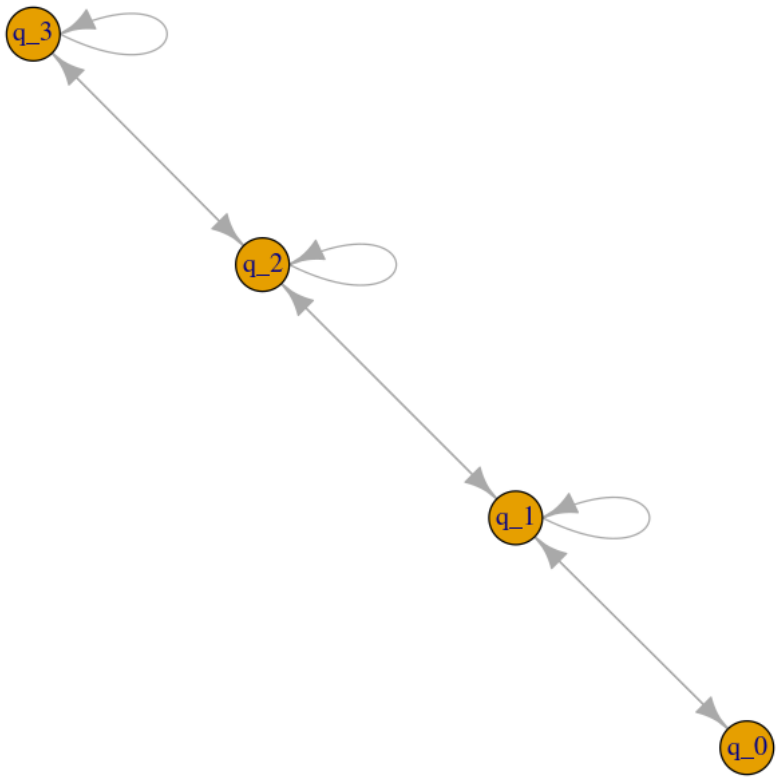
1.5 Example: Building an HMM for gene finding

We now possess all the necessary tools for designing, training, and deploying a rudimentary HMM-based gene finder. We will begin with the simple 4-state HMM.

```
In [2]: P=matrix(c(0,1,0,0,1,1,1,0,0,1,1,1,0,0,1,1), nrow=4,byrow=TRUE)
        P
```

0	1	0	0
1	1	1	0
0	1	1	1
0	0	1	1

```
In [5]: H=c("q_0","q_1","q_2","q_3")
        result <- graph_from_adjacency_matrix(P,mode="directed",weighted = TRUE)
        plot.igraph(result,vertex.label=H)
```




```
In [7]: source("skrypt.R")
```

```
In [8]: save(nazwy,tablica, test_set, file="Athaliana.Rdata")
```

1.6 GFF File

```
In [2]: load("Athaliana.Rdata")
```

```
In [9]: head(tablica)
```

V1	V2	V3	V4	V5	V6	V7	V8	V9
g10044	GlmrTrainData	exon	22266	22691	.	+	.	transgrp=1;
g10044	GlmrTrainData	exon	26242	26600	.	+	.	transgrp=2;
g10044	GlmrTrainData	exon	27248	27393	.	+	.	transgrp=2;
g10044	GlmrTrainData	exon	27470	28092	.	+	.	transgrp=2;
g10044	GlmrTrainData	exon	29283	29613	.	+	.	transgrp=3;
g10044	GlmrTrainData	exon	30084	30232	.	+	.	transgrp=3;

1.7 fasta file

```
In [19]: dane[[1]][1:100]
```

1. 'a' 2. 'a' 3. 'g' 4. 'c' 5. 't' 6. 't' 7. 't' 8. 'g' 9. 't' 10. 't' 11. 'a' 12. 'c' 13. 'a' 14. 't' 15. 'a' 16. 'g' 17. 'c' 18. 't' 19. 'g' 20. 't' 21. 't' 22. 'g' 23. 'a' 24. 'c' 25. 't' 26. 'a' 27. 'c' 28. 'g' 29. 'a' 30. 'g' 31. 'c' 32. 'a' 33. 'a' 34. 'g' 35. 'a' 36. 'a' 37. 'a' 38. 't' 39. 'g' 40. 'g' 41. 'a' 42. 'g' 43. 'a' 44. 'a' 45. 'g' 46. 'g' 47. 'c' 48. 't' 49. 'a' 50. 'c' 51. 'a' 52. 'a' 53. 'c' 54. 'g' 55. 'a' 56. 'g' 57. 'c' 58. 't' 59. 'c' 60. 'g' 61. 'g' 62. 'c' 63. 'a' 64. 'a' 65. 't' 66. 'c' 67. 'g' 68. 'a' 69. 't' 70. 'a' 71. 'g' 72. 'a' 73. 'a' 74. 'c' 75. 'c' 76. 't' 77. 'a' 78. 'c' 79. 'g' 80. 'a' 81. 'g' 82. 'c' 83. 't' 84. 't' 85. 'c' 86. 'c' 87. 'g' 88. 'g' 89. 'a' 90. 't' 91. 'g' 92. 'g' 93. 't' 94. 'c' 95. 'a' 96. 'a' 97. 'g' 98. 't' 99. 'g' 100. 'a'

1.8 Training sequence

```
In [10]: library("seqinr")
         n2s(test_set[[1]]$genom[1:100])
         test_set[[1]]$hidden[1:100]
```

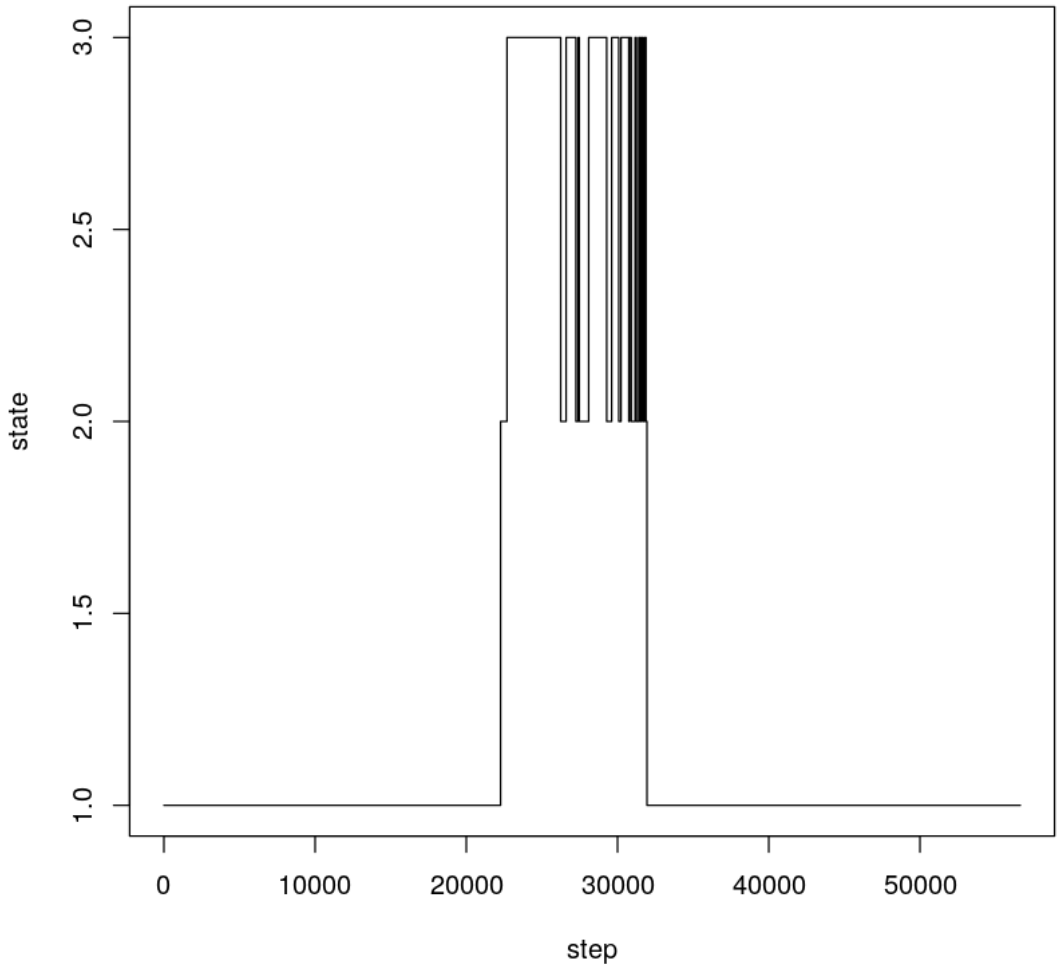
1. 'a' 2. 'a' 3. 'g' 4. 'c' 5. 't' 6. 't' 7. 't' 8. 'g' 9. 't' 10. 't' 11. 'a' 12. 'c' 13. 'a' 14. 't' 15. 'a' 16. 'g' 17. 'c' 18. 't' 19. 'g' 20. 't' 21. 't' 22. 'g' 23. 'a' 24. 'c' 25. 't' 26. 'a' 27. 'c' 28. 'g' 29. 'a' 30. 'g' 31. 'c' 32. 'a' 33. 'a' 34. 'g' 35. 'a' 36. 'a' 37. 'a' 38. 't' 39. 'g' 40. 'g' 41. 'a' 42. 'g' 43. 'a' 44. 'a' 45. 'g' 46. 'g' 47. 'c' 48. 't' 49. 'a' 50. 'c' 51. 'a' 52. 'a' 53. 'c' 54. 'g' 55. 'a' 56. 'g' 57. 'c' 58. 't' 59. 'c' 60. 'g' 61. 'g' 62. 'c' 63. 'a' 64. 'a' 65. 't' 66. 'c' 67. 'g' 68. 'a' 69. 't' 70. 'a' 71. 'g' 72. 'a' 73. 'a' 74. 'c' 75. 'c' 76. 't' 77. 'a' 78. 'c' 79. 'g' 80. 'a' 81. 'g' 82. 'c' 83. 't' 84. 't' 85. 'c' 86. 'c' 87. 'g' 88. 'g' 89. 'a' 90. 't' 91. 'g' 92. 'g' 93. 't' 94. 'c' 95. 'a' 96. 'a' 97. 'g' 98. 't' 99. 'g' 100. 'a'

1. 1 2. 1 3. 1 4. 1 5. 1 6. 1 7. 1 8. 1 9. 1 10. 1 11. 1 12. 1 13. 1 14. 1 15. 1 16. 1 17. 1 18. 1 19. 1 20. 1 21. 1 22. 1 23. 1 24. 1 25. 1 26. 1 27. 1 28. 1 29. 1 30. 1 31. 1 32. 1 33. 1 34. 1 35. 1 36. 1 37. 1 38. 1 39. 1 40. 1 41. 1 42. 1 43. 1 44. 1 45. 1 46. 1 47. 1 48. 1 49. 1 50. 1 51. 1 52. 1 53. 1 54. 1 55. 1 56. 1 57. 1 58. 1 59. 1 60. 1 61. 1 62. 1 63. 1 64. 1 65. 1 66. 1 67. 1 68. 1 69. 1 70. 1 71. 1 72. 1 73. 1 74. 1 75. 1 76. 1 77. 1 78. 1 79. 1 80. 1 81. 1 82. 1 83. 1 84. 1 85. 1 86. 1 87. 1 88. 1 89. 1 90. 1 91. 1 92. 1 93. 1 94. 1 95. 1 96. 1 97. 1 98. 1 99. 1 100. 1

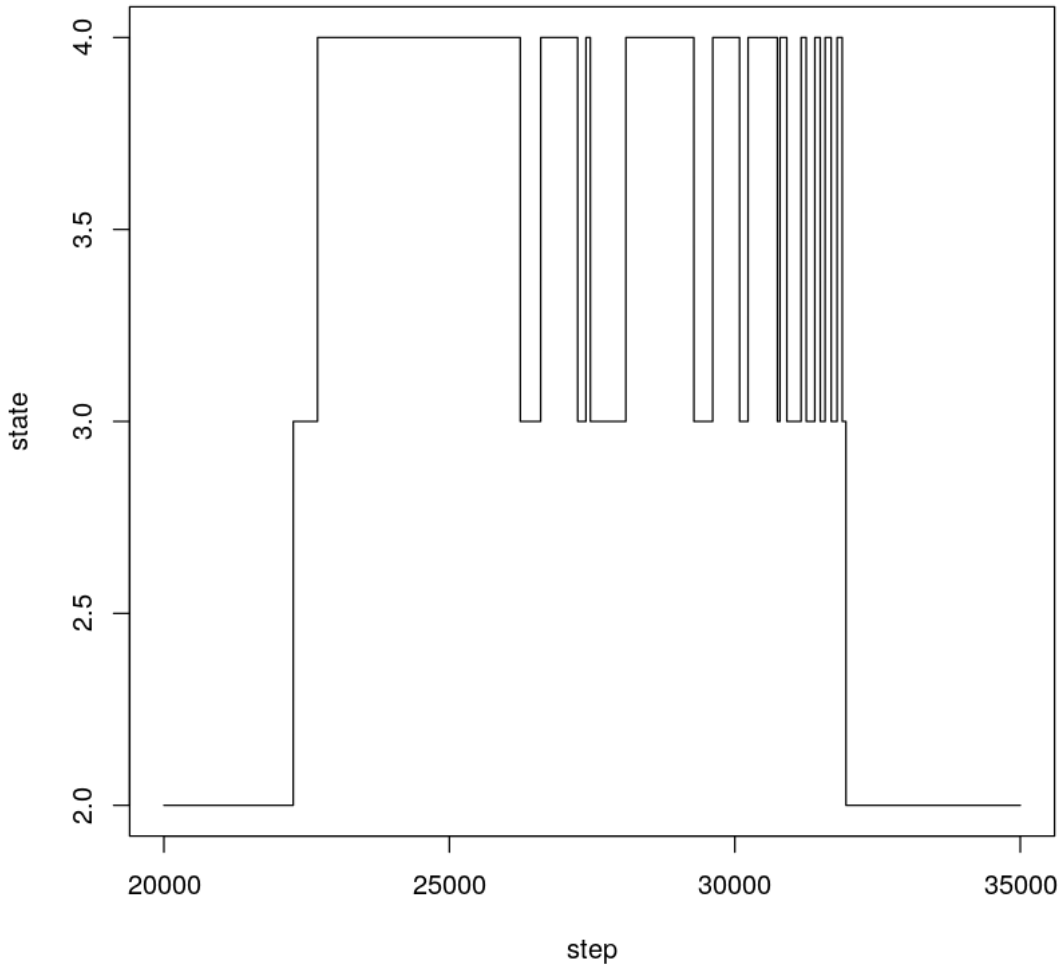
```
In [11]: size=length(test_set[[1]]$hidden)
         size
```

56635

```
In [12]: plot(1:size, test_set[[1]]$hidden, type="l", xlab="step", ylab="state")
```




```
In [13]: plot(20000:35000, test_set[[1]]$hidden[20000:35000]+1, type="l", xlab="step", ylab="st
```



1.9 HMM learning

```
In [23]: source("hmm.R")
         L=HMMlearn(test_set,1:500)

In [15]: P=L$P
         P
```

	q_0	N	E	I
q_0	0.000000e+00	1.0000000000	0.000000e+00	0.000000000
N	1.776808e-05	0.9999644638	1.776808e-05	0.000000000
E	0.000000e+00	0.0002805281	9.947349e-01	0.004984541
I	0.000000e+00	0.0000000000	4.515360e-04	0.999548464

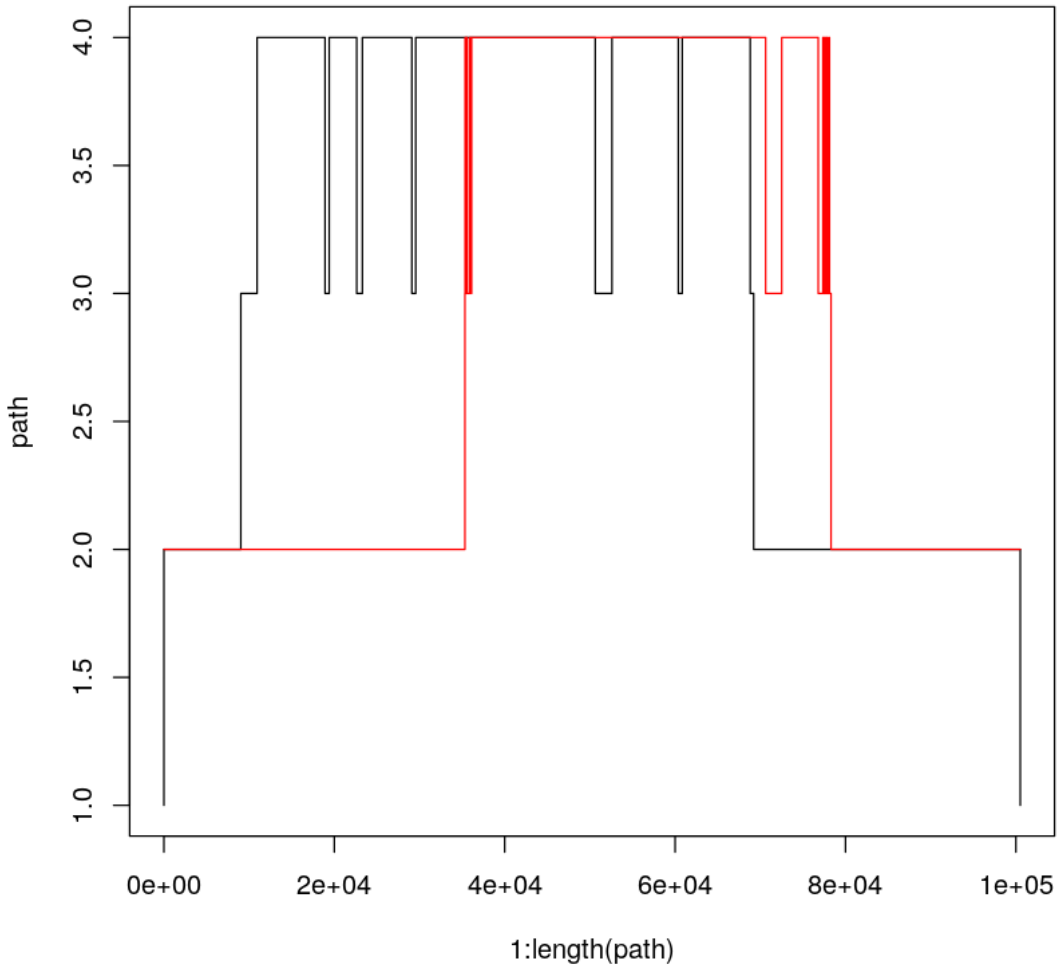
In [16]:
```
E=L$E
E
```

```
          A         C         G         T
N 0.3250217 0.1779100 0.1762130 0.3208554
E 0.2802003 0.2107475 0.2398604 0.2691918
I 0.3226070 0.1744013 0.1750941 0.3278976
```

1.10 Test

In [24]:
```
source("hmm.R")
seq=450
Sequence=test_set[[seq]]$genom+1
path=Viterbi(P,E,Sequence)

plot(1:length(path), path, type="l")
lines(1:length(Sequence),test_set[[seq]]$hidden+1, col="red" )
```


path
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
1:length(path)