Example: In the human genome, the dinucleotide (that is, two base pairs next to each other in the sequence) CG, written CpG to distinguish it from the nucleotide pair C-G, is subject to methylation. Methylation changes the G to a T. That means we see the CpG dinucleotide less frequently than we would expect by considering the individual frequencies of C and G. In some functional regions of the genome, such as promoter and start regions of genes, methylation is suppressed and CpG occurs at a higher frequency than elsewhere. These are called CpG islands.

To detect these regions we model each region of a sequence as having either high or low CG content. We label the regions H and L. In high CG regions, nucleotides C and G each occur with probability 0.35 and nucleotides T and A with probability 0.15. In low CG regions, C and G are probability 0.2 and T, A are 0.3. The initial state is H with probability 0.5 and L with probability 0.5. Transitions are given by $a_{HH} = a_{HL} = 0.5$, $a_{LL} = 0.6, a_{LH} = 0.4$. Use the Viterbi algorithm to find the most likely sequence staes given the sequence of symbols $x = GGCACTGAA$.

Solution: We work in the log space (base 2) so that the numbers don't get too small. We'll need the log values of each of the above probabilities which are best represented as matrices: set $\mathbf{A} = \log_2(a)$ where a is the transition matrix

$$\begin{array}{cccc} & & H & L \\ \mathbf{A} = & H & \log(a_{HH}) = -1 & -1 \\ & L & -1.322 & -0.737 \end{array}$$

and if $\mathbf{E} = \log_2(e)$ where e is the matrix if emission probabilities then

$$\begin{array}{cccccc} & & A & C & G & T \\ \mathbf{E} = & H & \log(e_H(A)) = -2.737 & -1.515 & -1.515 & -2.737 \\ & L & -1.737 & -2.322 & -2.322 & -1.737 \end{array}$$

The matrix V, with row indices k and column indices i (so that the (k,i)th element is $V_k(i)$) along with the pointers to allow traceback is

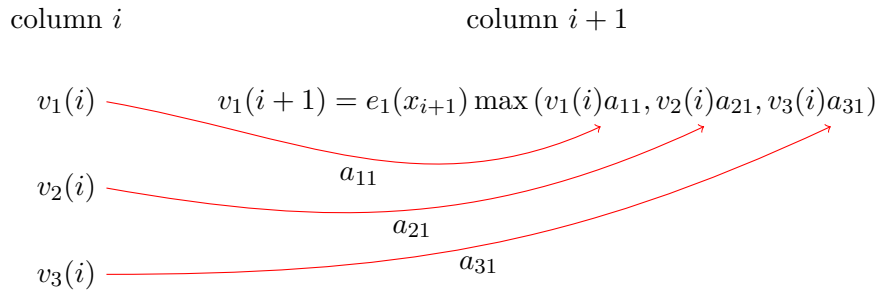
	–	G	G	C	A	C	T	G	A	A
0	0									
H	$-\infty$	-2.51	-5.03	-7.54	-11.28	-13.12	-16.85	-18.65	-22.39	-25.41
L	$-\infty$	-3.32	-5.84	-8.35	-10.28	-13.34	-15.81	-18.87	-21.35	-23.82

The first column in this matrix is simple: every sequence is in state 0 at step 0, so $V_0(0) = \log(1) = 0$ while other states have $V_H(0) = V_L(0) = \log(0) = -\infty$.

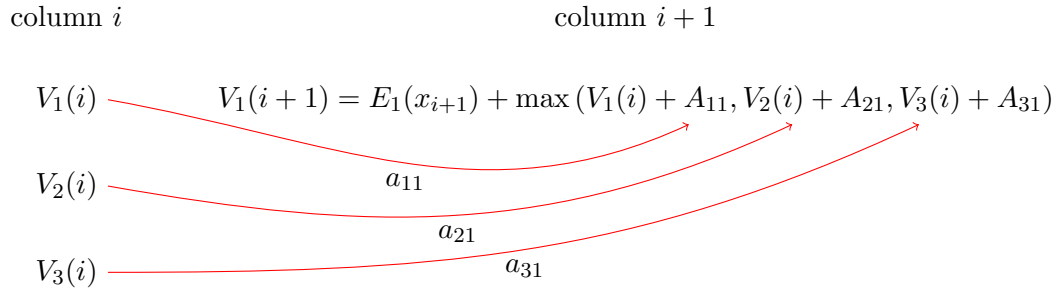
The second column is derived from the first as follows:
$V_H(1) = \log(e_H(G)) + \max\{V_0(0) + \log(a_{0H}), V_H(0) + \log(a_{HH}), V_L(0) + \log(a_{LH})\} = -1.515 + (V_0(0) + log(a_{0H})) = -1.515 - 1 = -2.515$ and similarly for $V_L(1)$.

Traceback begins in the final column where we see the state that maximises the joint probability is L. Following the pointers from this position and recording the state at each step gives us the state path with the highest probability is $\pi^* = HHHLLLLLL$. Note that π^* is built from right to left in the traceback procedure. □

The schematic below shows how the $(i+1)$th column is derived from the ith column in a Viterbi matrix. Here, there are 3 possible states, 1, 2 and 3. The 0 state is omitted in this diagram.



The same diagram shown using log units:



18.2 The forward algorithm and calculating P(x)

We have seen that it is easy to calculate $P(x, \pi)$. However, we usually only observe x so can't directly calculate $P(x, \pi)$. We could tackle this by finding a suitable state path, such as the Viterbi path, π^*, and calculate $P(x, \pi^*)$. But calculating $P(x, \pi^*)$ does not adequately tell us the likelihood of observing x which may have arisen from a large number of possible state paths.

What we really want is calculate is $P(x)$, the probability of observing x without taking any particular state path into account. This involves marginalizing over all possible paths: that is,

$$P(x) = \sum_{\pi} P(x, \pi).$$

The number of possible state paths grows exponentially so we cannot enumerate them all and naively calculate this sum. Instead, we use another dynamic programming algorithm called the forward algorithm and calculate $P(x)$ iteratively.

The forward algorithm iteratively calculates the quantity

$$f_k(i) = P(x_{1:i}, \pi_i = k),$$

the joint probability of the first i observations and the prob that $\pi_i = k$. The recursion used is that

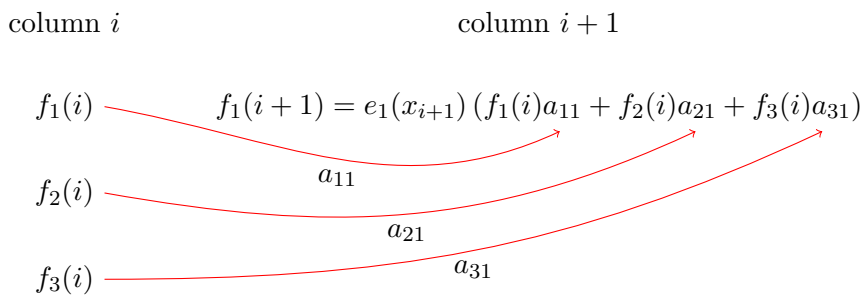
$$f_l(i+1) = e_l(x_{i+1}) \sum_k f_k(i) a_{kl}. \tag{1}$$

Initialisation $i = 0$**:** $f_0(0) = 1$, $f_k(0) = 0$ for $k > 0$.

Recursion $i = 1, \ldots, L$**:** $f_l(i) = e_l(x_i) \sum_k f_k(i-1) a_{kl})$.

Termination: $P(x) = \sum_k f_k(L) a_{k0}$.

If the end state 0 is not modelled, simply set $a_{k0} = 1$ to get $P(x) = \sum_k f_k(L)$.
Here's a diagram showing how to get the $(i+1)$th column from the ith column in the forward algorithm. The example shown has three states 1, 2 and 3.



Once again, we'll need to work with the log quantities as the qualities of interest get very small very fast. However, if we take the log of both sides of Equation **??**, the log of the sum on the right hand side does not simplify immediately.
Let $F_k(i) = \log(f_k(i))$ and $A_{kl} = \log(a_{kl})$. Then Equation **??** becomes

$$F_l(i) = \log[e_l(x_i) \sum_k f_k(i-1) a_{kl})] = \log[e_l(x_i)] + \log\left[\sum_k \exp(F_k(i-1) + A_{kl})\right]$$

Directly calculating a sum of the form $\log(c) = \log(e^a + e^b)$ requires calculating e^a and e^b which we were trying to avoid all along. Instead, note that $\log(e^a + e^b) = \log(e^a(1 + e^{b-a})) = \log(e^a) + \log(1 + e^{b-a}) = a + \log(1 + e^{b-a})$. If the difference $b - a$ is not too large, this method never need store an extremely large or small number so is numerically stable. This extends to finding the log of a sum of multiple logged numbers:

```
function logsum(x)
  return x[0] + log(sum(exp(x - x[0])))
```

or, if you are using $\log_2$,

```
function log2sum(x)
  return x[0] + log2(sum(2^(x - x[0])))
```

Example cont: For the CG island example above, use the forward algorithm to calculate the probability of the sequence $x = GGCACTGAA$.

Solution: The matrix produced by the forward algorithm is given below, in log units (base 2). The first column is based on the start state, 0. The first entry of the second column is $log(f_H(1)) = log(e_H(G)) + log(1/2)$

	–	G	G	C	A	C	T	G	A	A
0	0									
H	∞	-2.51	-4.49	-6.39	-9.51	-10.49	-13.55	-14.53	-17.58	-19.78
L	∞	-3.32	-5.08	-6.97	-8.27	-10.89	-12.30	-14.92	-16.33	-18.37

The log probability is thus $\log(P(x)) = \log(2^{-19.78} + 2^{-18.37}) = -17.91$.