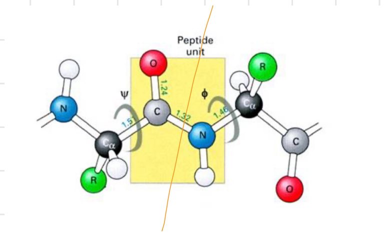
Review

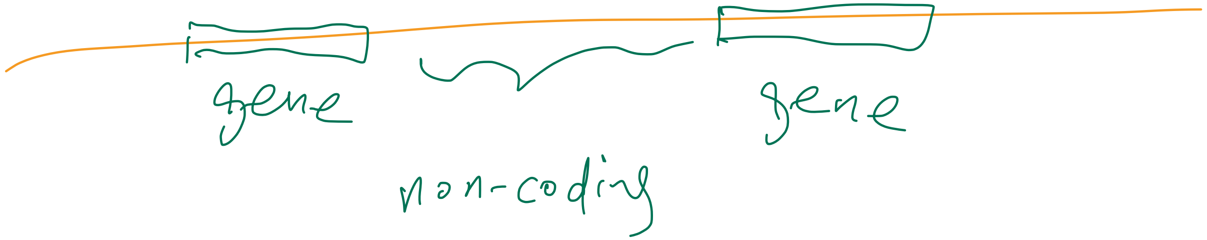
- protein structure prediction.
  - Torsion angels.



- Free energy
- Co-evolution
- contact map.
- Res Net
- AlphaFold2.

Hidden Markov Model

gene prediction:



HMM

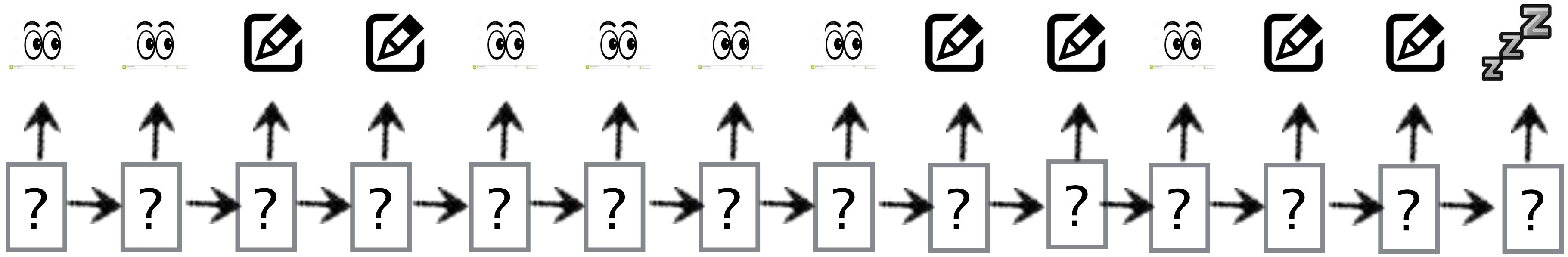
- Hidden Markov model was first invented in speech recognition. But are widely used in many other areas including bioinformatics.
- An automata that has “hidden states”. At each time point, it emits a symbol, and change a state with certain probability.
- We want to derive the hidden states by the emitted symbols.

Classroom example

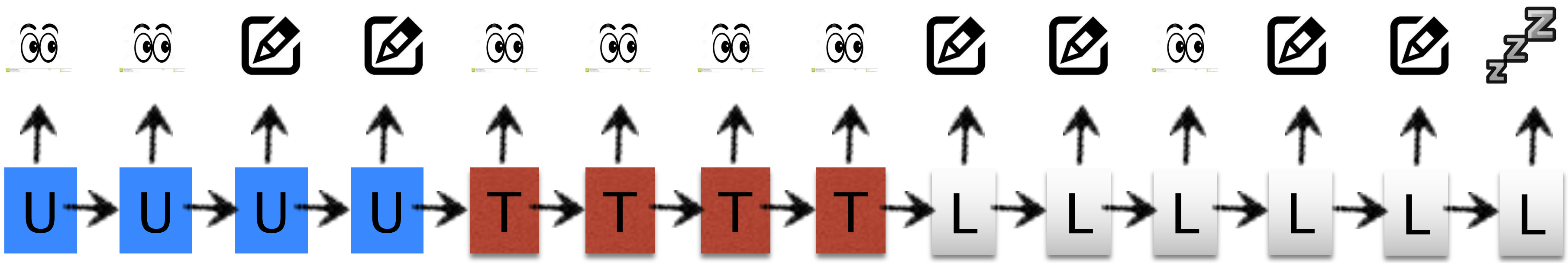
- Think of a student in classroom.
- At any minute, a student is in one of 3 hidden *states* that I try to figure out:
  - U: understands
  - T: does not understand but tries to understand
  - L: is lost completely and does not try to understand
- Meanwhile, the student emits one of 3 *symbols* that I can observe
  - Look at me
  - Write/Type
  - Sleep

Classroom Example

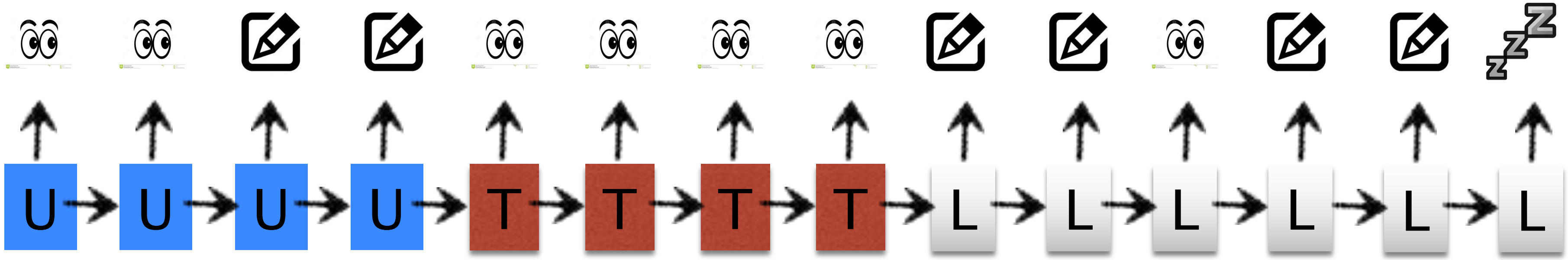
- Now suppose I see a student's behavior is the following in the past several minutes. What is his internal states at each minute?



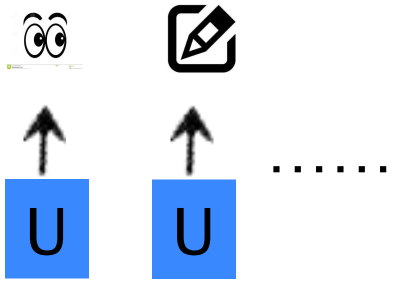
Classroom Example



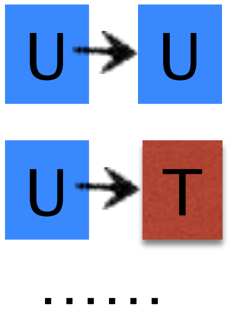
Classroom Example



Emissions

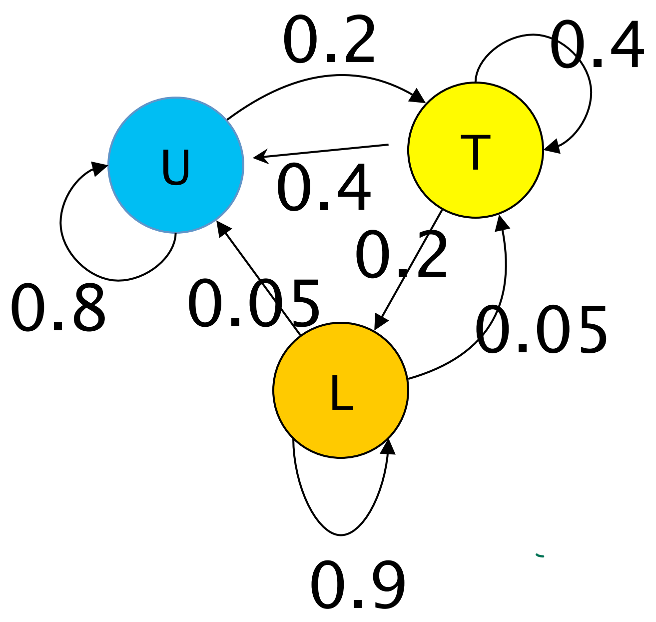


Transitions



Typically, HMM assumes that emission probability depends only on current state; and current state only depends on previous state. We want to find the most likely path of states given the symbols (observations).

Classroom Example



U: Understands
T: Tries to understand
L: Lost completely

(T) Transition matrix

	U	T	L
U	0.8	0.2	0
T	0.4	0.4	0.2
L	0.05	0.05	0.9

add up to 1

(E) Emission matrix

	Look	Write	Sleep
U	0.6	0.35	0.05
T	0.9	0.1	0
L	0.1	0.6	0.3

add up to 1

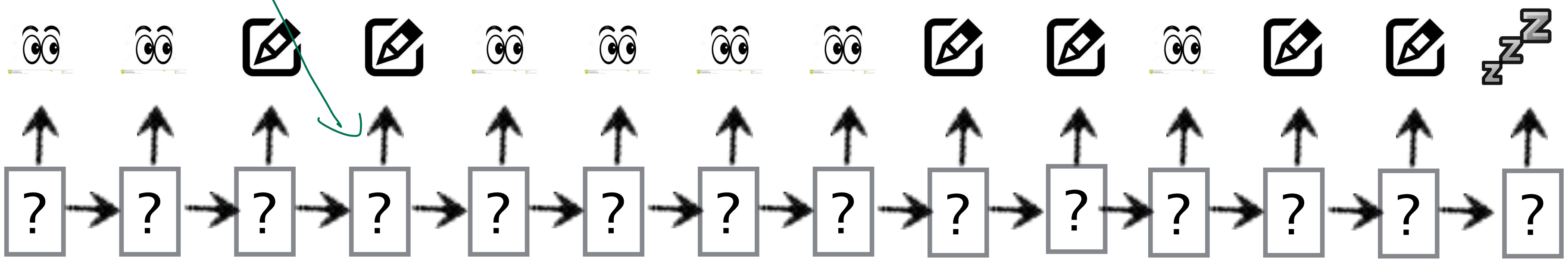
Classroom Example

- $S=S_1S_2\ldots S_n$: sequence of symbols;
- $P=P_1P_2\ldots P_n$: path of states.
- We want to maximize Pr(P|S) = Pr(P,S) / Pr(S).
- Therefore, we want to maximize

transission emission

$Pr(S_i|P_i)$

$$\Pr(P,S)=\prod_i \Pr(P_i\mid P_{i-1})\Pr(S_i\mid P_i)$$

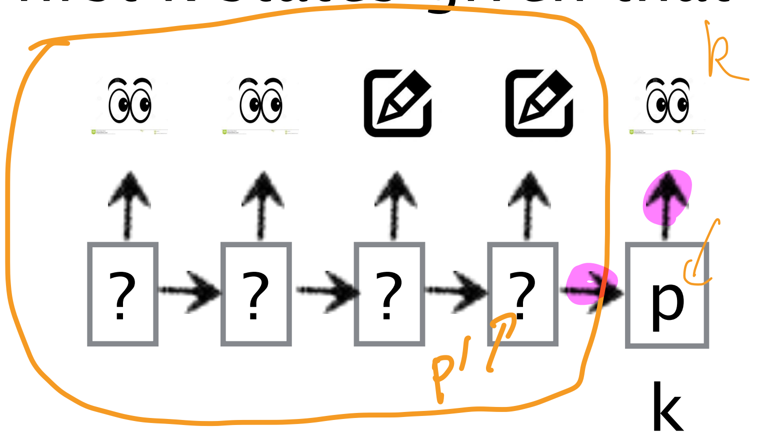


$Pr(P_i|P_{i-1})$.

* Note: To deal with the first state, we can define $\Pr(P_1|P_0)=1$ in above formula.

Solving HMM

- We use dynamic programming again. Define D[k,p] be the maximum probability achieved by first k states given that the last state is p.



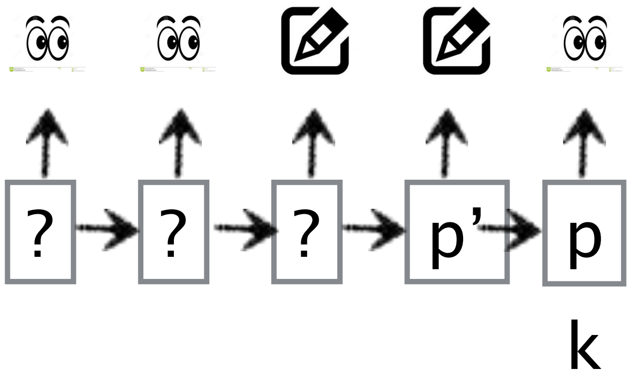
$D[k,p] = \max_{p'} \left(D[k-1,p'] \cdot T[p',p] \cdot E[p,S_k] \right)$

- Then $\max_p$ D[n,p] is the maximum probability achieved by the complete path, which is what we want to compute.
- It is not hard to obtain a recurrence Relation:

Definition of D[k,p] →

$$D[k,p] = \max_{P[1..k];P[k]=p} \prod_{1 \le i \le k} T[P_{i-1},P_i] E[P_i,S_i]$$

Solving HMM

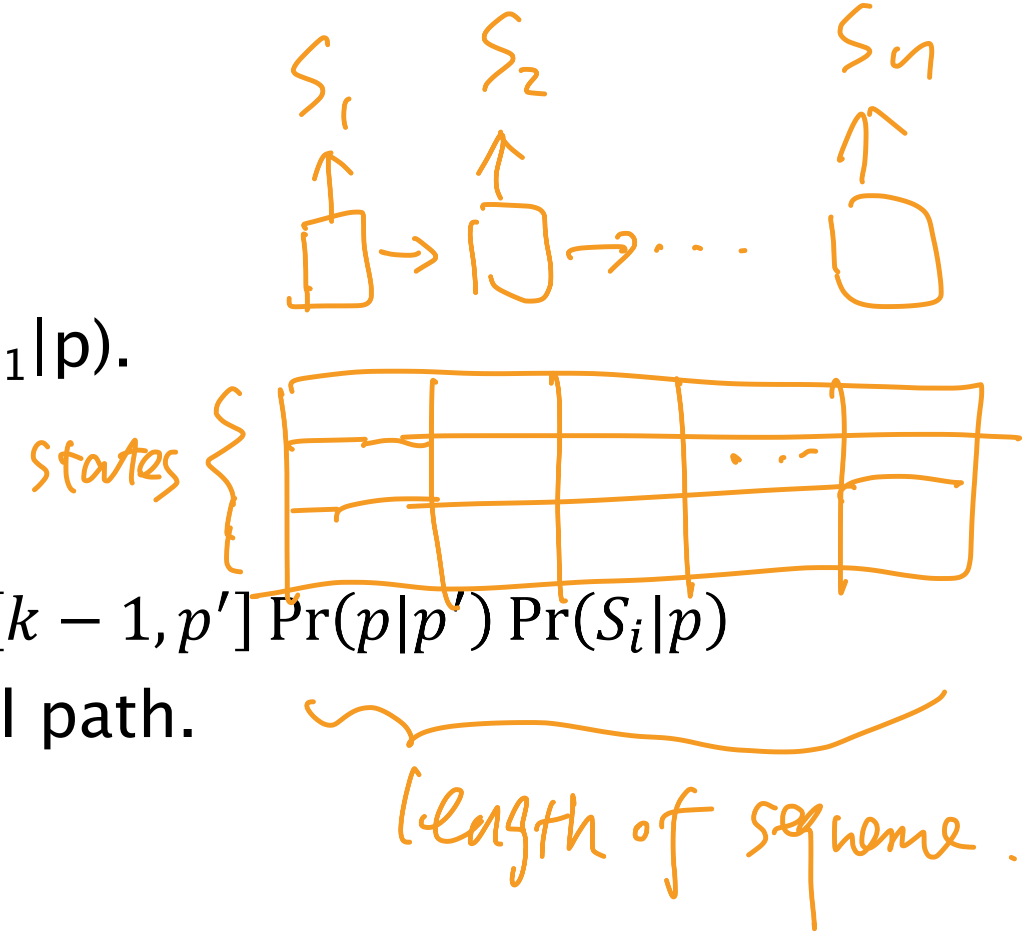


$$D[k,p] = \max_{p'} \underbrace{D[k-1,p']}\, \underbrace{\Pr(p|p')}_{T[p',p]}\, \underbrace{\Pr(S_i|p)}_{E[p,S_i]}$$

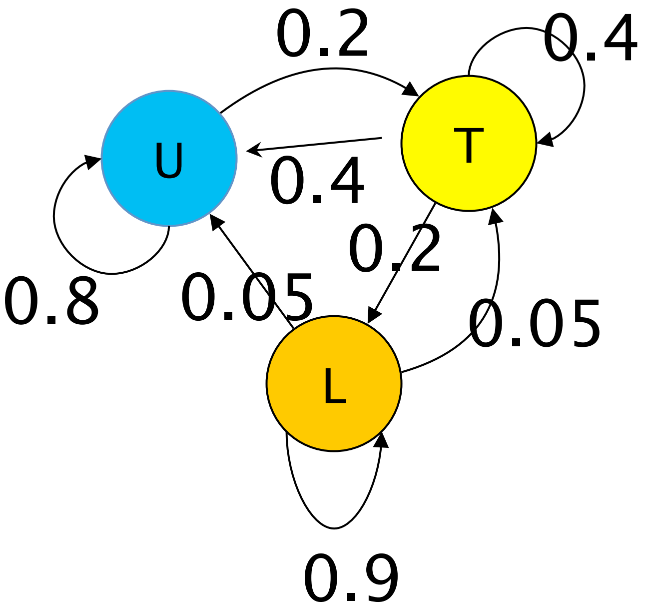
k

Solving HMM

- The algorithm:
- Input: $S = S_1S_2\ldots S_n$
- Output: $P=P_1P_2\ldots P_n$
- 1. for every state p, let $D[1,p] = \Pr(S_1|p)$.
- 2. for k from 2 to n,
- 2.1 for every state p,
- 2.1.1 let $D[k,p] = \max_{p'} D[k-1,p'] \Pr(p|p') \Pr(S_i|p)$
- 3. backtrace to compute the optimal path.



Example



(E) emission matrix

	Look	Write	Sleep
Understand	0.6	0.35	0.05
Try	0.9	0.1	0
Lost	0.1	0.6	0.3

$$D[k,p] = \max_{p'} D[k-1,p'] \Pr(p|p') \Pr(S_i|p)$$

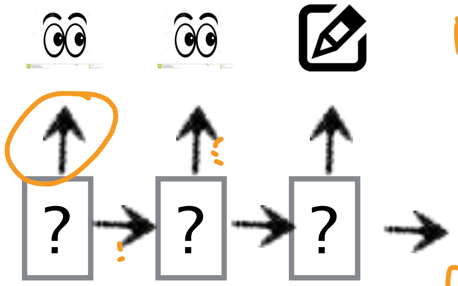
D[2,U]

case 1: p' = U

D[2,U] = D[1,U]·T[U,U]·E[U,L]

= 0.6×0.8×0.6

= 0.288.



case2: p' = T.

0.9×0.4×0.6 = 0.216

U	0.6	0.288 0.216 0.003	
T	0.9		
L	0.1		

case 3: p' = L

0.1×0.05×0.6 = 0.003

Notes

- Do not multiply
  - because soon the numbers become so small that the double precision will give you value 0.
  - Do a logarithm and use additions instead.

$$D[k,p] = \max_{p'} D[k-1,p'] \Pr(p|p') \Pr(S_i|p)$$

$$\Downarrow$$

$$\log D[k,p] = \max_{p'} (\log D[k-1,p'] + \log \Pr(p|p') + \log \Pr(S_i|p))$$

Parameter Estimation

- All of our computation depends on the transition probabilities and emission probabilities. How do we estimate these parameters?

Parameter Estimation

- If we have an annotated sequence with both symbols and states, then these can be trained by counting.
- If we do not, then we can start with a reasonable guess of the parameters and annotate the sequence.
- Then we use the annotation to train a new set of probabilities. Repeat until converge.
- There is some guarantee to the convergence. But does not guarantee this will converge to the right solution.

Pseudocounts

- If the training data include no cases of a particular emission from a particular state, then its probability will be 0 in this model.
- That's no good.
- So we add pseudocounts to make the probabilities not zero when an event should be able to happen.

Higher Order HMM

- Think again the classroom example:

U: Understands
T: Tries to understand
L: Lost

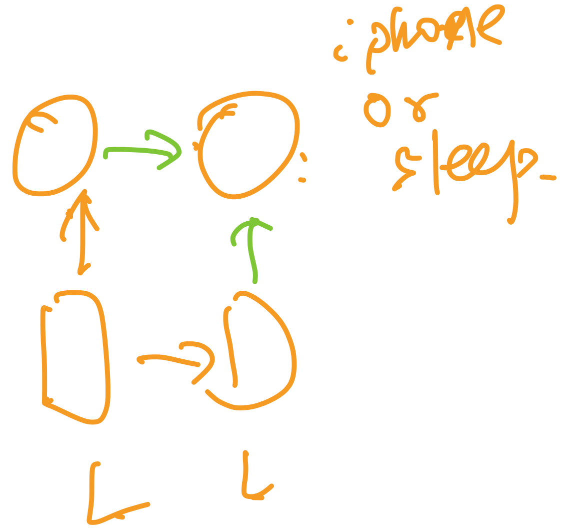
(E) Emission matrix

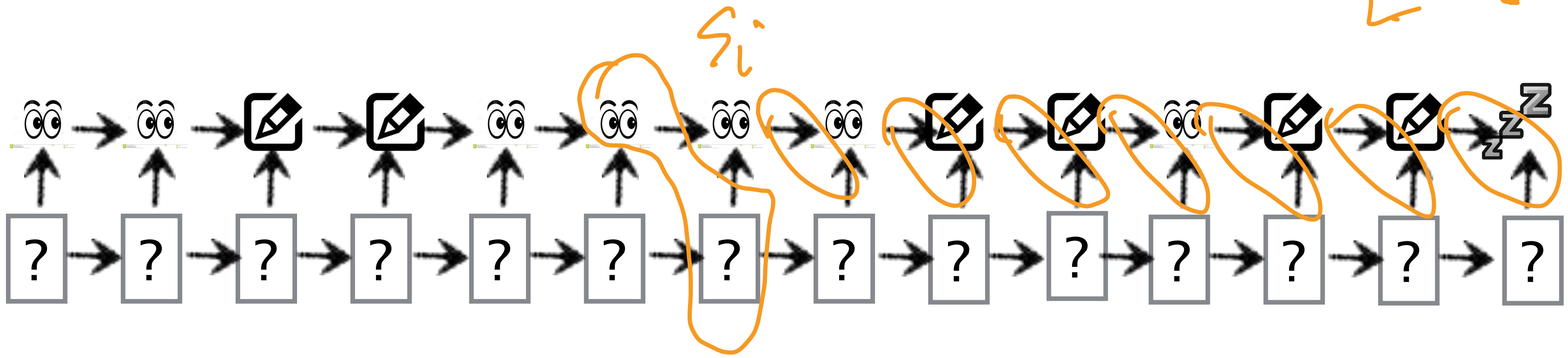
	Look	Write	Sleep
U	0.6	0.35	0.05
T	0.9	0.1	0
L	0.1	0.6	0.3

- The emission of a symbol should not only depend on current state, but sometimes also the previous symbol.
  - E.g. Sleeping at previous moment leads to a higher probability of sleeping now.

1st Order HMM

- To accommodate the correlation between the adjacent symbols, the emission matrix needs to be expanded.
- The emission matrix becomes $Pr(S_i \mid P_i, S_{i-1})$.





1st Order HMM

- Before

transion emission

$$\Pr(P,S)=\prod_i \Pr(P_i \mid P_{i-1})\Pr(S_i \mid P_i)$$

2nd order

- Now

same different

$$\Pr(P,S)=\prod_i \Pr(P_i \mid P_{i-1})\Pr(S_i \mid P_i, S_{i-1}) \rightarrow \Pr(S_i \mid P_i, S_{i-1}, S_{i-2}).$$

- To find the path P to maximize, we let D[k,p] be the maximum probability obtained by the first k states ending at p. We can obtain the following recurrence relation similarly as before.

$\Pr(S_i \mid P)$

$$D[k,p]=\max_{p'} D[k-1,p']\Pr(p|p')\Pr(S_i|p,S_{i-1})$$

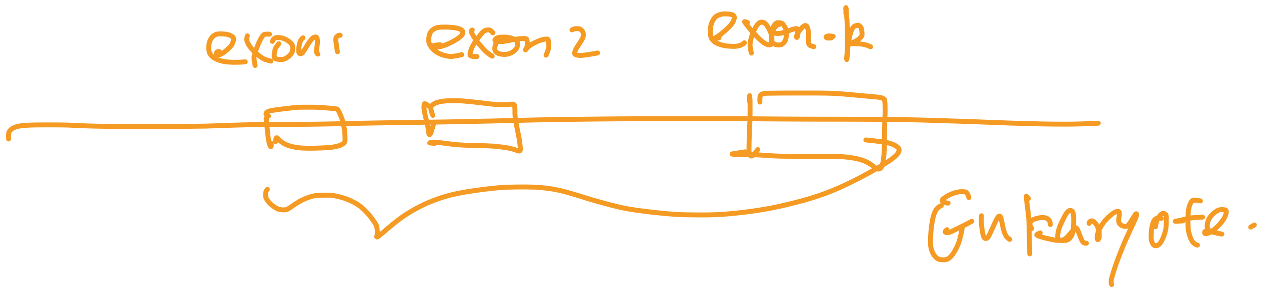
- We can still do dynamic programming.

Higher Order HMM

- To generalize, we can let the current emission depend on the current state, and previous k symbols.
- Then this is called the k-th order HMM.
- Solving such a HMM is similar as before. Running time not changed.
- The only difficulty is the parameter training because the emission matrix has many more parameters for larger k.

Prokaryote Gene Finding

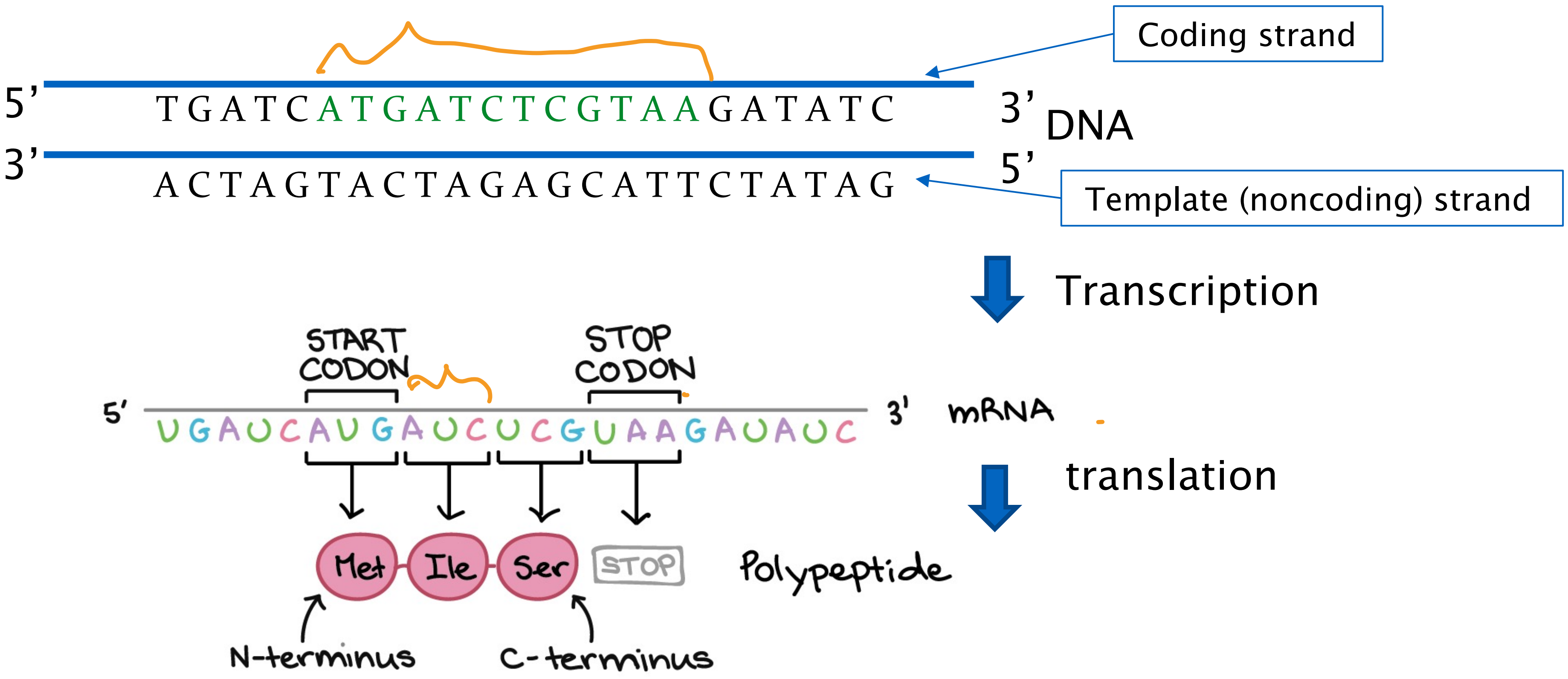
- The **prokaryotes** (pronounced /proʊ'kærioʊts/; singular **prokaryote** /proʊ'kæriət/) are a group of organisms that lack a cell nucleus (= karyon).
  - The opposite is the eukaryotes.
- Most of prokaryotes are unicellular.
- Prokaryote genes do not have introns. So their genes is a linear structure.



Intron video:

http://www.youtube.com/watch?v=o0BQJbLNYSg

From Gene to Protein (in *Prokaryotes*)



Genetic code

$4^3 = 64$
codons

Second letter

First letter	U	C	A	G	Third letter
U	UUU UUC Phenyl-alanine UUA UUG Leucine	UCU UCC UCA UCG Serine	UAU UAC Tyrosine UAA Stop codon UAG Stop codon	UGU UGC Cysteine UGA Stop codon UGG Tryptophan	U C A G
C	CUU CUC CUA CUG Leucine	CCU CCC CCA CCG Proline	CAU CAC Histidine CAA CAG Glutamine	CGU CGC CGA CGG Arginine	U C A G
A	AUU AUC AUA Isoleucine AUG Methionine; start codon	ACU ACC ACA ACG Threonine	AAU AAC Asparagine AAA AAG Lysine	AGU AGC Serine AGA AGG Arginine	U C A G
G	GUU GUC GUA GUG Valine	GCU GCC GCA GCG Alanine	GAU GAC Aspartic acid GAA GAG Glutamic acid	GGU GGC GGA GGG Glycine	U C A G

codons

$$
\begin{matrix} \cdot \\ \cdot \\ \left.\begin{matrix} A \\ T \\ T \end{matrix}\right\} \text{I} \\ \left.\begin{matrix} C \\ A \\ C \end{matrix}\right\} \text{H} \\ \left.\begin{matrix} A \\ G \\ T \end{matrix}\right\} \text{S} \\ \left.\begin{matrix} G \\ G \\ A \end{matrix}\right\} \text{G} \\ \cdot \\ \cdot \end{matrix}
$$

A Trivial Gene Finder

- Open Reading Frame (ORF) is a substring that
  - starts with a start codon
  - ends with a stop codon
  - no stop codon in the middle
- If ORF is long, then likely it is a gene or a part of a gene.
- Why?

e.g. 500 codons in an ORF.

$\frac{3}{64}$ stop

…TAG

…TGA

ATG… …TAA

start codon

stop codon

no stop codon in middle

Codon bias

- A codon XYZ occurs with different frequencies in coding regions and non-coding regions
  - different amino acids have different freq.
  - Diff. codons for the same amino acid have diff. freq.
  - In random regions approx. p(X)*p(Y)*p(Z)

Codon Bias Tables
(% of codons used for each residue)

Amino Acid	Codon		
Gly	GGG	2	25
Gly	GGA	0	25
Gly	GGU	59	16
Gly	GGC	39	34