Hidden Markov Model II

A brief review of HMM

- HMM is used to model sequential data. Observed data are assumed to be "emitted" from hidden states, where the hidden states is a "Markov chain".
- A HMM is characterized by initial/emission/transition probabilities.
- Difference between HMM and mixture model is the correlations between hidden states.
- The goals of HMM include (1) parameter estimation; (2) underlying states estimation; (3) determine the best path.
- We have covered an EM with "forward-backward" algorithm for parameter estimation.
- We will cover **dynamic programming** and **Viterbi algorithm** in this lecture.

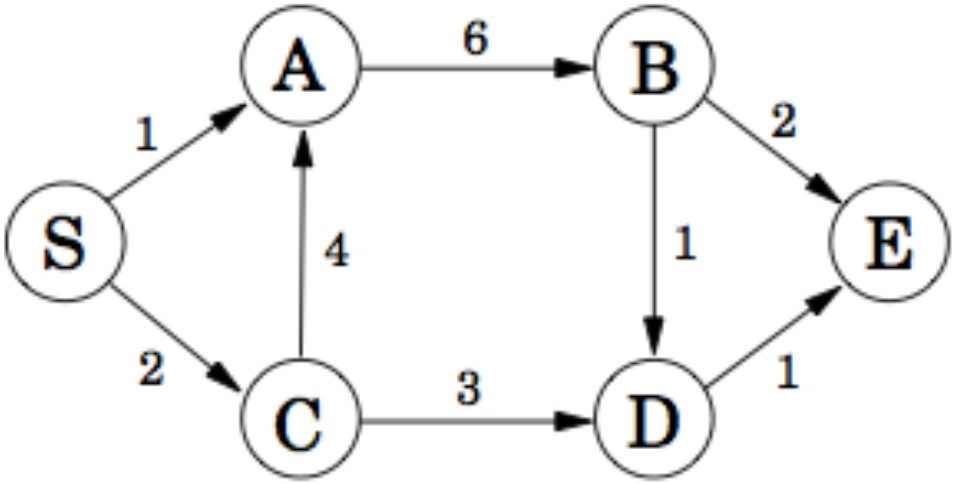
Dynamic Programming (DP)

"*Two sledgehammers of the algorithms craft: dynamic programming and linear programming*"

- DP is a general optimization algorithm.
- Breaking the overall optimization problem into overlapping smaller problems.
- Solve each sub-problem once, and reuse the results, thus reducing the computing cost (dramatically).
- Often working backward.

A simple DP example

Find the shortest path from S to E in the directed acyclic graph below.

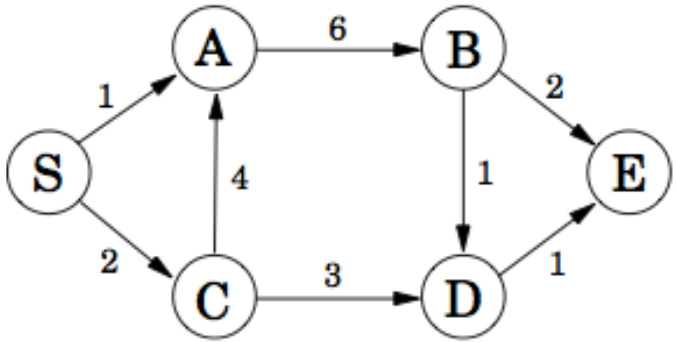


The problem can be solved backward. Take node D as an example. The way to get to D is through B or C. So, $\text{dist}(D) = \min\{\text{dist}(B) + 1, \ \text{dist}(C) + 3\}$.

Then work our way backward, we can find the best path.

Compare with exhaustive approach

Using exhaustive search , one has to do:



- SABE: 1+6+2
- SCABE: 2+4+6+2
- SCDE: 2+3+1
- SCABDE: 2+4+6+1+1

Total is 11 additions. The complexity grows exponentially with the size of graph

Using DP, do:

- Dist(A)=min(1, 2+4)=1
- Dist(C)=2
- Dist(B)=dist(A)+6=1+6=7
- Dist(D)=min(dist(B)+1, dist(C)+3)=min(7+1, 2+3)=5
- Dist(E)=min(dist(B)+2, dist(D)+1)=min(7+2, 5+1)=6

Total is 6 additions. The complexity grows linearly with the size of graph.

Finding most likely hidden state sequence

Under the notations:

- Observed data: $\boldsymbol{u} = \{u_1, u_2, \ldots, u_T\}$.
- Hidden states: $\boldsymbol{s} = \{s_1, s_2, \ldots, s_T\}$
- Model parameters: $\boldsymbol{\lambda} = \{\pi_k, b_k(u), a_{k,l}\}$.

We want to find the most possible "path": $\hat{\boldsymbol{s}} = \text{argmax}_s \, Pr(\boldsymbol{s}|\boldsymbol{\lambda}, \boldsymbol{u})$. This is called the rule of *Maximum A Posteriori* (MAP) (mode of the posterior probability).

Since we have:

$$Pr(\boldsymbol{s}|\boldsymbol{\lambda}, \boldsymbol{u}) = \frac{Pr(\boldsymbol{s}, \boldsymbol{u}|\boldsymbol{\lambda})}{Pr(\boldsymbol{u}|\boldsymbol{\lambda})}$$

The denominator doesn't involve $\boldsymbol{s}$. So

$$\underset{\boldsymbol{s}}{\text{argmax}} \, Pr(\boldsymbol{s}|\boldsymbol{\lambda}, \boldsymbol{u}) = \underset{\boldsymbol{s}}{\text{argmax}} \, Pr(\boldsymbol{s}, \boldsymbol{u}|\boldsymbol{\lambda})$$

In other words, to maximize the conditional probability, we can simply maximize the joint probability.

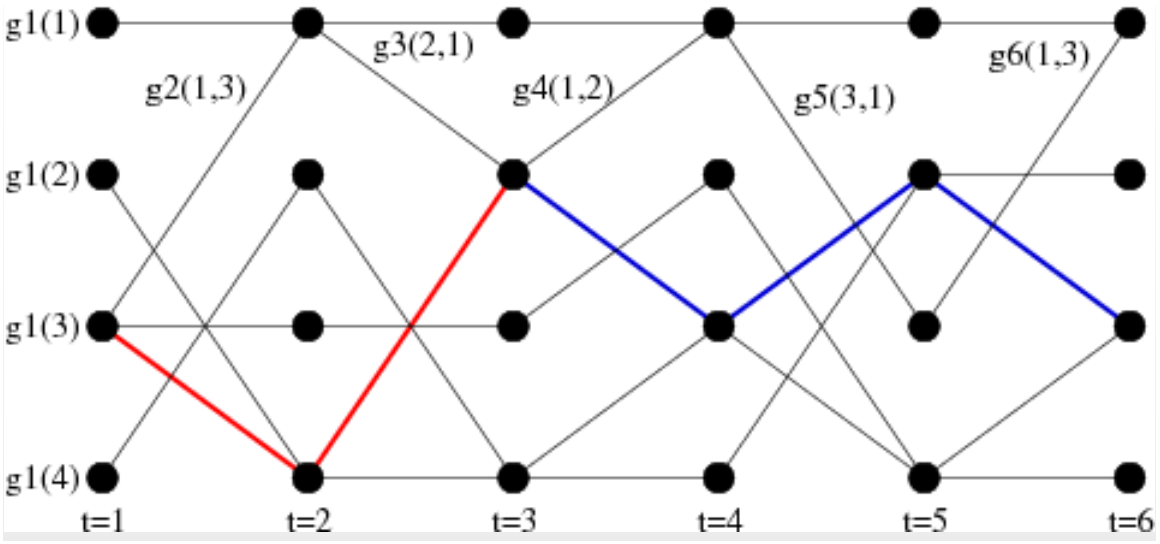
The Viterbi Algorithm

- The Viterbi algorithm maximizes an objective function $G(\boldsymbol{s})$, where $\boldsymbol{s} = \{s_1, \ldots, s_T\}$ is a sequence of categorical values: $s_t \in \{1, \ldots, M\}$.
- $G(\boldsymbol{s})$ satisfies following special property:

$$G(\boldsymbol{s}) = g_1(s_1) + g_2(s_2, s_1) + g_3(s_3, s_2) + \ldots + g_T(s_T, s_{T-1}).$$

  So the objective function can be expressed as sum of functions depending one state and its preceding one.

- In a diagram, let $g_t(k, l)$ be the distance from state l at $t-1$ to state k at t. At the starting node, use $g_1(k)$ for state k. The optimal path can be found through DP.



The Viterbi Algorithm in HMM

In a HMM, the distance between nodes are the transition probabilities. But we still need to consider emission probabilities.

Remember we want to find optimal sequence $\boldsymbol{s}^*$:

$$\boldsymbol{s}^* = \underset{\boldsymbol{s}}{\operatorname{argmax}}\, Pr(\boldsymbol{s}, \boldsymbol{u}|\boldsymbol{\lambda}).$$

The objective function can be expressed as:

$$\begin{aligned} G(\boldsymbol{s}) &= \log Pr(\boldsymbol{s}, \boldsymbol{u}|\boldsymbol{\lambda}) = \log[\pi_{s_1} b_{s_1}(u_1) a_{s_1,s_2} b_{s_2}(u_2) a_{s_2,s_3} \ldots a_{s_{T-1},s_T} b_{s_T}(u_T)] \\ &= [\log \pi_{s_1} + \log b_{s_1}(u_1)] + [\log a_{s_1,s_2} + \log b_{s_2}(u_2)] + \ldots + [\log a_{s_{T-1},s_T} + \log b_{s_T}(u_T)] \end{aligned}$$

If we define

$$\begin{aligned} g_1(s_1) &= \log \pi_{s_1} + \log b_{s_1}(u_1) \\ g_t(s_t, s_{t-1}) &= \log a_{s_{t-1},s_t} + \log b_{s_t}(u_t) \end{aligned}$$

then $G(\boldsymbol{s}) = g_1(s_1) + \sum_{t=2}^{T} g_t(s_t, s_{t-1})$, and Viterbi algorithm can be applied.

Viterbi training

- Notice that the Viterbi algorithm requires that the model parameters $\boldsymbol{\lambda}$ are known.
- "Viterbi training" algorithm can be applied to estimate $\boldsymbol{\lambda}$. The steps are:
  1. Choose initial values of $\boldsymbol{\lambda}$.
  2. Under current $\boldsymbol{\lambda}$, find the optimal path $\boldsymbol{s}^*$.
  3. Let $L_k(t) = \mathbb{1}(s_t^* = k)$ and $H_{k,l} = \mathbb{1}(s_{t-1} = k)\mathbb{1}(s_t = l)$, then update $\boldsymbol{\lambda}$ using the same M-step procedures derived before.
- Viterbi training replaces the step of computing forward and backward probabilities by finding the optimal path $\boldsymbol{s}^*$ under the current parameters using Viterbi algorithm.
- Basically, it uses "hard" classification (0/1) to replace the "soft" classification (probabilities).

Selecting the number of states (M)

- This is a model selection problem.
- Since the whole data likelihood $P(u)$ is available, this can be done by using BIC/AIC.
- With one more state, there are more parameters from initial probability, transition probabilities, and emission probability.
- However, based on my experience, BIC tends to select large M in real data, especially when the chain is long.
- Sometimes have to use arbitrary criteria.

Numerical issue

- When the HMM chain is long, the computation of forward/backward matrices must be carried in logarithm scale, i.e., the forward/backward matrices stores $\log(\alpha)$ and $\log(\beta)$. Otherwise the α and β values become 0 very quickly, since they will take values like 10^{-1000}.
- However, there are sums of likelihood (not log-likelihood) in computation, for example, $\alpha_k(t) = b_k(u_t) \sum_{l=1}^{M} \alpha_l(t-1) a_{l,k}$. How to compute these when we have $\log \alpha_l(t-1)$ instead of $\alpha_l(t-1)$?
- If one directly compute $\log(e^a + e^b)$, it will give negative infinity when a or b are negative number with large absolute values. Try to run following in R: `log(exp(-1000)+exp(-1000))`.
- This can be computed using the following trick:
  $log(e^a + e^b) = log(e^a(1 + e^{b-a})) = a + log(1 + e^{b-a})$
- Let it equals b when $b >> a$, and equals a when $b << a$. When the values of b and a are close, the computation is numerically stable. From this we get $\log(e^{-1000} + e^{-1000}) = -999.3069$.

Multivariate HMM

- So far we have discussed univariate HMM, e.g., u_t is a scalar.
- When observation is a random vector, it can be modeled as a multivariate HMM.
- The emission probability $b_k(u)$ becomes a multivariate distribution.
- There are more parameters need to be estimated, but the procedure is the same.

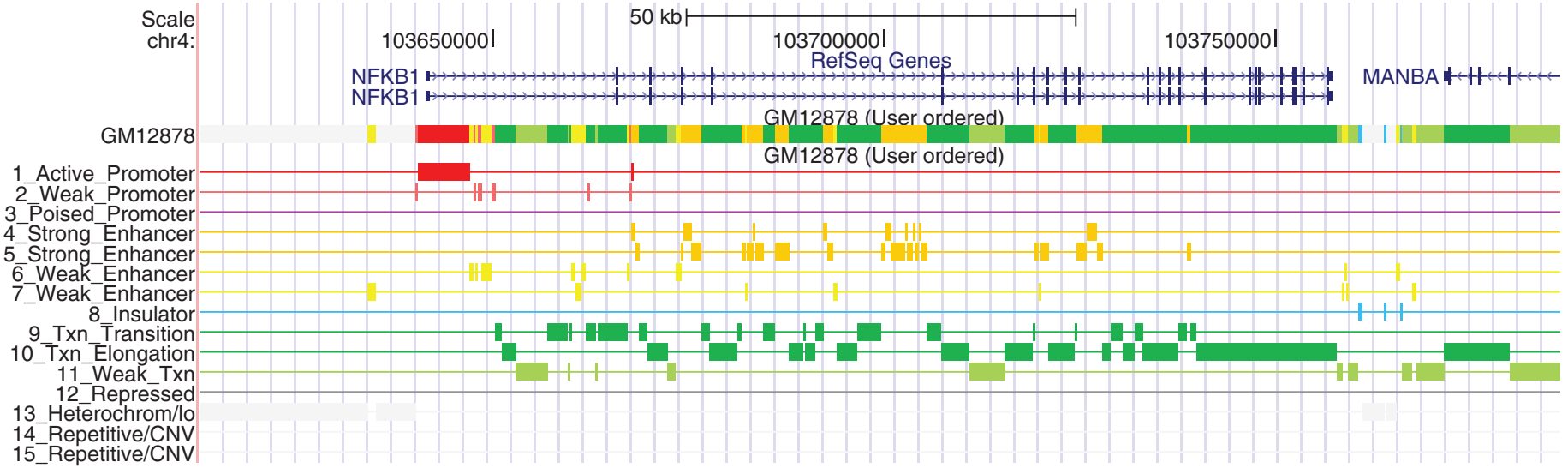
Multivariate HMM example: chromHMM

Ernst & Kellis, **Nature Method** 2012

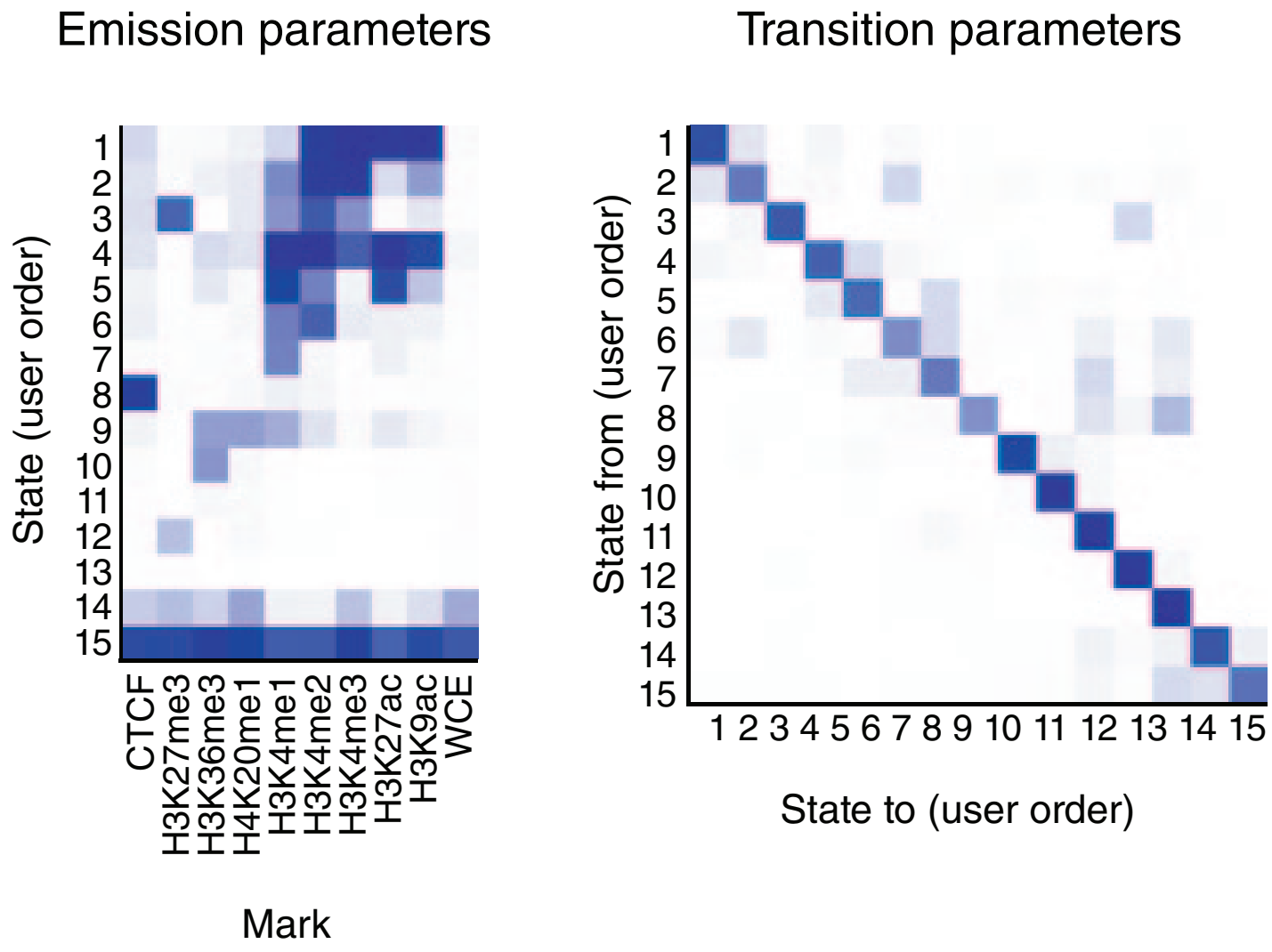
ChromHMM: automating chromatin-state discovery and characterization

- The observed data are multiple ChIP-seq datasets profiling histone modification and protein binding strengths.
- The data are measurements from 200 bp bins genome-wide. There are around 10 million bins (chain is 10 million long).
- The goal is to segment the whole genome into a number of “states”.
- They run Multivariate HMM, chains are assumed to be independent.
- Result segments the genome into 15 states.

chromHMM genome segmentation result on a gene:



chromHMM emission and transition probabilities:



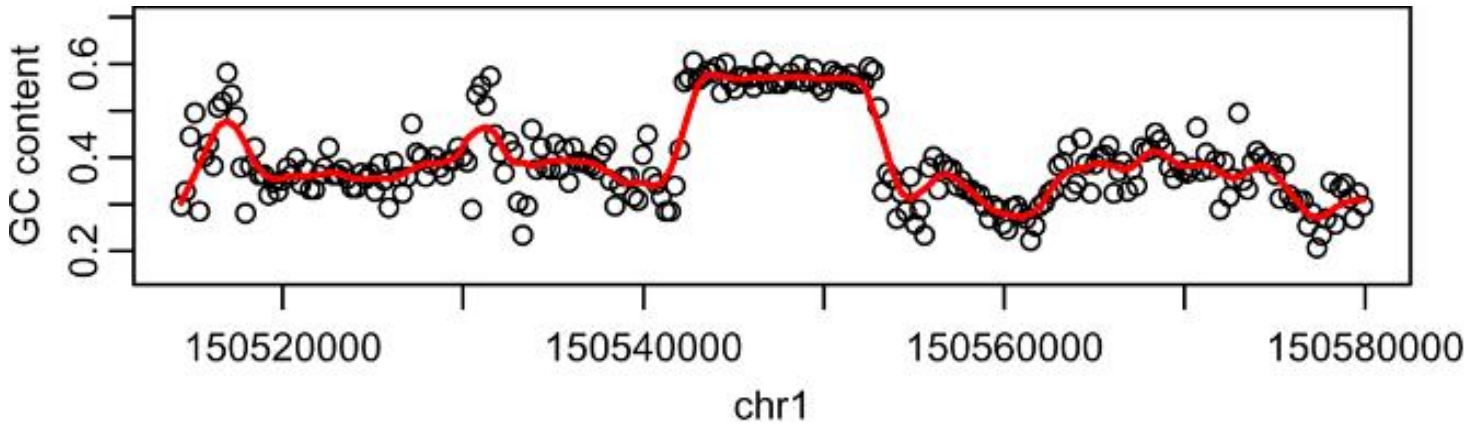
Auto-regressive HMM (ARHMM)

- The observations within a state have spatial correlations (remember HMM assumes observations are independent conditional on state).
- There are different ways to model the spatial correlations within a state, such as AR model or smoothing.
- Advantage over HMM: avoid over-fitting. The state segmentation of the chain is smoother.

Example – Define CpG island

- DNA sequence is a long (3 billion for human) string of four letters: A, C, G, T.
- The appearance of “CG” is rare, due DNA methylation, mutations and selections.
- However, there are regions where “CG” appears more frequently compared with overall. Such regions are called “CpG islands”, and they often mark important regions (such as gene promoters).
- The CpG islands can be detected by modeling the DNA sequence using HMM.
- Observed data are C+G content and CG appearance in 16bp bins.

GC content plot in a small region:



Our way to model this: the observation follows a smooth curve within each segmentation.

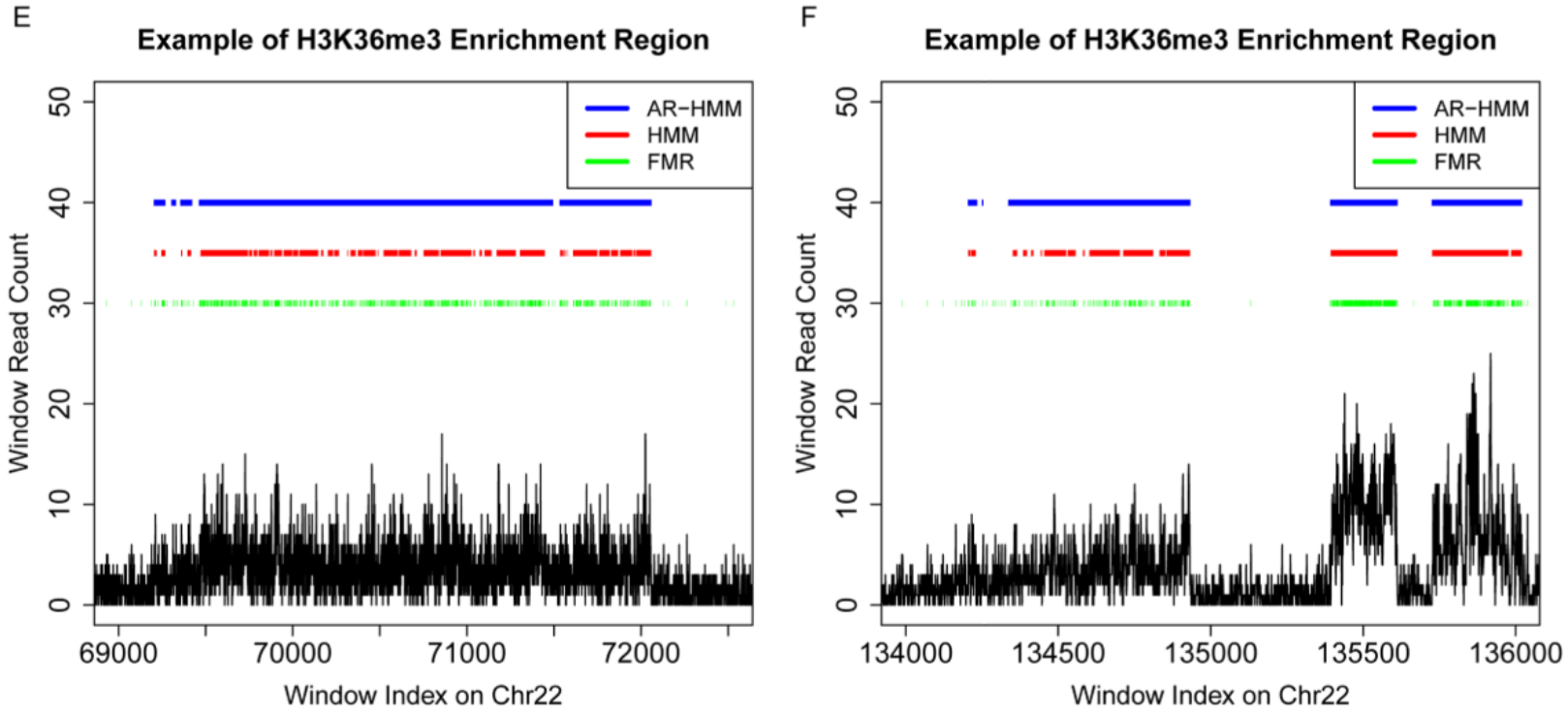
$$p(s) \mid s \in S_j \text{ and } X(M_j) = i \sim \text{Normal}\{c_i + f(s), \tau^2\},$$

For details, read Wu *et al.* (2010) *Redefining CpG islands using hidden Markov models*. Biostatistics.

Another example of ARHMM

Rashid *et al.* (2014) *Some Statistical Strategies for DAE-seq Data Analysis: Variable Selection and Modeling Dependencies among Observations*. JASA

- The goal is to detect long range histone modified regions.
- We want the regions to be long, but directly fitting a HMM often gives overly fragmented, short regions.
- Use AR model for the spatial dependence.



Other variations of HMM

Non-homogeneous HMM

Transition probability varies along the chain. Have to impose some constrains on the transition probabilities so that they can be estimated.

Higher-order HMM

Assume the hidden states are from a higher order Markov chain, e.g., the current state depends on several previous states.

Hierarchical HMM (HHMM)

Each state of the HHMM is itself an HHMM, e.g., the states of the HHMM emit sequences of observation that follows another HHMM.

2D HMM

Used in image segmentation. Inputs are 2D data emitted from a Markov random field. Need to model the transition from one observation to its neighbor. However a fully connected 2D HMM is NP-hard (computationally unsolvable). So different approximation is used (Pseudo 2D HMM).

Review

- HMM is used to model sequential data. Observed data are assumed to be "emitted" from hidden states, where the hidden states is a "Markov chain".
- A HMM is characterized by initial/emission/transition probabilities.
- Difference between HMM and mixture model is that HMM assumes correlations between hidden states, whereas mixture model assumes independence.
- The goals of HMM include (1) parameter estimation; (2) underlying states estimation; (3) determine the best path.
- Parameter estimation can be done by EM with "forward-backward" algorithm.
- Dynamic program (DP) is a general optimization method to find the shortest path in a directed acyclic graph.
- Viterbi algorithm is a DP algorithm applied in finding the optimal path for a HMM.
- Viterbi training is a simplified version of forward-backward algorithm. It uses hard classification to replace soft classification.