Recap: Graphical Models

- How to handle high-dimensional data and high-dimensional models with hidden variables?
- The global distribution is intractable, but with graphical models it is approximated by a product over tractable distributions.
- Bayesian network: directed acyclic graph, each node denotes a variable.
- Global factorization property:

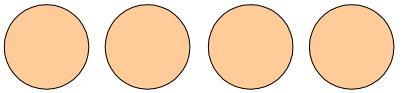
$$p(X_{1,}\ldots,X_N)=\prod_i p(X_i|X_j;j\in N^-(i))$$

- See slides 2-3 for more information.

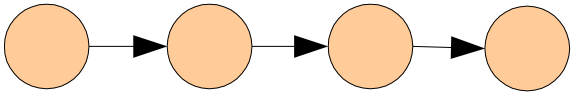
Increasingly Complex Models for Sequences

1. Simplest: single-die model
   - trivial, used as a background model for comparison

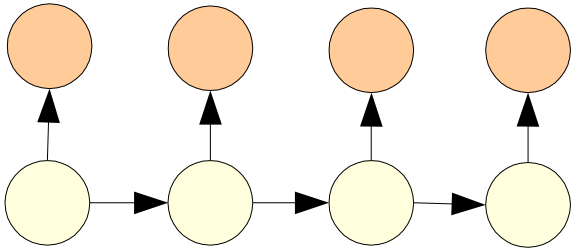
2. Separate dice at each position
   - profiles from multiple alignments



3. Markov models
   - higher-order models: dependencies many steps in the past
   - parameter space grows quickly with the chain degree
   - orders up to 6 used in e.g. gene finding

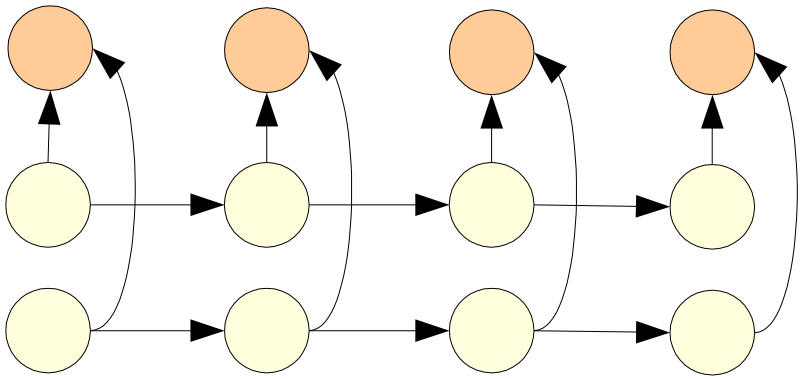


4. Hidden Markov models
   - can take into account insertions, deletions

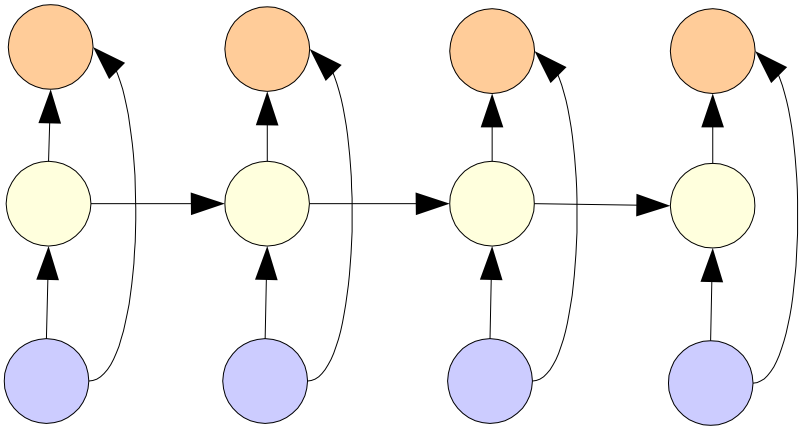


More Complex Markovian Models

Factorial HMMs: output depends on several Markov chains



Input-Output HMMs: output and hidden states depend on input too

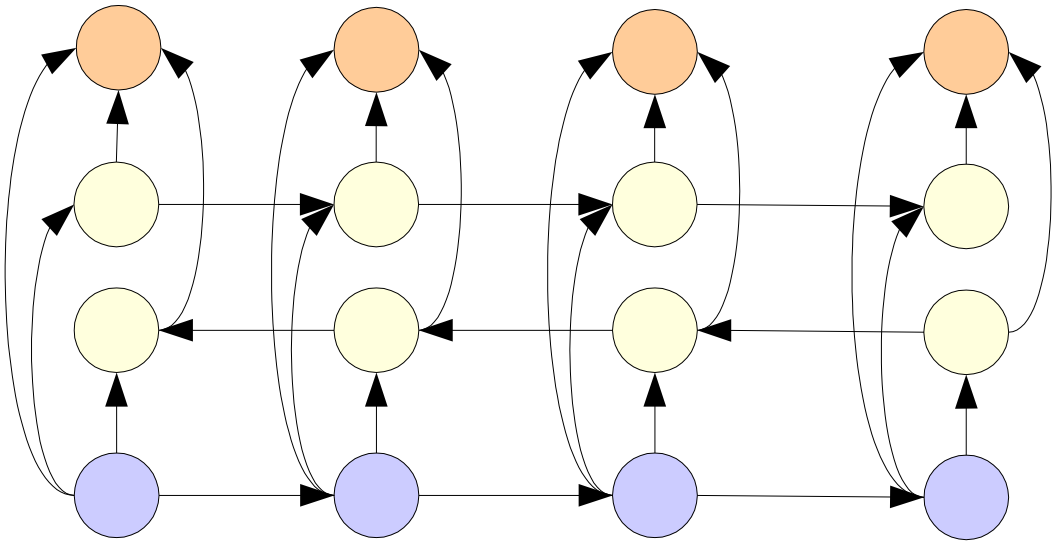


BIOHMMs

- Biological sequences have spatial structure, not temporal!
  → "Future" information could be used.

- Just add backward Markov chains? No, equivalent to forward chains.
- Reversing a chain in a factorial HMM doesn't help; identical result.

- But: adding a backward chain to an input-output HMM
  → BIOHMM.



Output

Forward Markov chain

Backward Markov chain

Input

Application: Investigating the Parity Rule

- Chargaff's 1st parity rule:
  In double-helical DNA, n. of A = n. of T
  n. of C = n. of G

- Chargaff's 2nd parity rule: the above works (approximately) with just 1 strand!

- Does that hold for e.g. dinucleotides too?
  (AA etc., equivalent to 2nd order Markov model)

- Markov model of order N for DNA: 4^N transition probabilities.
  Larger data sets needed to fit higher order models.

Complementarity and Symmetry

- Complementarity: Markov model on one strand defines a model on the reverse complement.
- Symmetric model: $P(X_{1,}\ldots,X_N)=P(\bar{X}_N,\ldots,\bar{X}_1)$
- If a high-order model is symmetric, so are lower-order projections, but it does not work the other way around!
- Residual symmetry: correlation of ratios between N-mers and their reverse complements
- Chargaff's 2nd parity rule is remarkably valid. Complex influences at different scales?
- Forces that don't distinguish between strands contribute to 2nd parity rule. Mutations, replication machinery (optimized for same number of base pairs).

Gene Finders

- GeneMark, GeneMark.hmm, GLIMMER, GRAIL, GenScan, ...
- Modular architecture, same basic strategies
- Elementary modules: detect **boundary elements** or **variable length regions**

(see GeneMark.hmm illustration in Baldi and Brunak, page 237)

Gene Finder Modules

- Detecting boundary elements:
  - splice sites, start & stop codons, protein binding sites, ...
  - consensus sequences, profiles, weight matrices (special cases of 1^{st} order Markov models), Markov models, neural networks

- Detecting variable length regions:
  - exons, introns, intergenic regions
  - Markov methods.
  - Coding regions: account for 3- and 6-periodicities
  - Exon models: account for reading frames

- Scan and parse large genomic regions (Viterbi paths)

- Performance has improved

Hybrid Models

Combine HMMs and neural networks (NNs)

1. NNs as front-end processors (extract features for HMMs), separate training

2. NNs classify likelihood pattern produced by many HMMs

3. Hybrid architectures
   - HMM and NN not separable
   - NN reparameterizes and modulates HMM
   - unified training

Hybrid Framework

- Address overfitting (too complex model) and underfitting (too complex data)
- Single-model case: reparameterize a complex model $M(\theta)$ with a simple vector w: $\theta = f(w)$
- Multiple-model case: modulate parameters with the input: $\theta = f(I)$
- Or both together: $\theta = f(w, I)$
- f can be in another model class

Single-Model Case for HMMs

- Emission/transition probabilities depend on state only: $\theta = f(i)$
- Compute f by a neural network
- Normalized exponential reparameterization:

$$e_{iX} = \frac{e^{w_{iX}}}{\sum_{Y} e^{w_{iY}}}$$

- Compute f by a small NN, 1 bias, no hidden layer, softmax output units. w_{iX}: input-output connections

Single-Model Case, continued...

- Generalizations: more complex NNs, link NNs for different states (states can be partitioned to groups, e.g., 1 NN for insert states, 1 for main states, ...)
- Parameter reduction: small hidden layers, $|H|$ small in comparison to N, $|A|$. Roughly $|H|N$ params. Number of hidden units can be adaptively adjusted to training set sizes.
- All NN techniques can be used.
- Prior information (substitution matrices) can be implemented.
- Continuous emission distributions can be used.

Single-Model Case, continued...

- What emission distributions can be achieved depends on connections from hidden layer to output. Hidden layer activities: vector in $[-1,1]^{|H|}$.

- Emission distributions have the form

$$e_{iX} = \frac{\prod_{h \in H} [\, p_X^h \,]^{\mu_h}}{\sum_{Y \in A} \prod_{h \in H} [\, p_Y^h \,]^{\mu_h}} \qquad \text{where} \qquad p_X^h = \frac{e^{-\beta_{Xh}}}{\sum_{Y \in A} e^{-\beta_{Yh}}}$$

- Combinations of $|H|+1$ basic distributions (1/hidden unit), but not convex linear combinations.

- Learning done by ML or MAP estimation.

- EM algorithm: M step cannot be done analytically → gradient descent, Viterbi learning (GEM algorithms).

Multiple-Model Case for HMMss

In the single-model case, the final model is just one HMM.

How to account for long-range dependencies?

1. Higher-order HMMs: intractable

2. Markov models with varying memory length?
   must have a method for learning the lengths from data,
   number of relevant contexts should be small

3. Instead of a different model class, more general HMM/NN architectures. Underlying model: set of HMMs.

Multiple-Model Case, continued...

Example:

	i		j	
. . .	X	. . .	Y	. . .
. . .	X'	. . .	Y'	. . .

Must have two HMMs, switch depending on the sequence.
Or: one HMM with emissions modulated by the input

- Emission depends on additional information I, $\theta = f(i,I)$
  - f can be computed by NN
  - I can be a "context" variable, or even the current sequence, or e.g. protein secondary structure.
- Example: mixture of HMM experts, $P(O|M) = \sum_{i=1}^{n} \lambda_i P(O|M_i)$

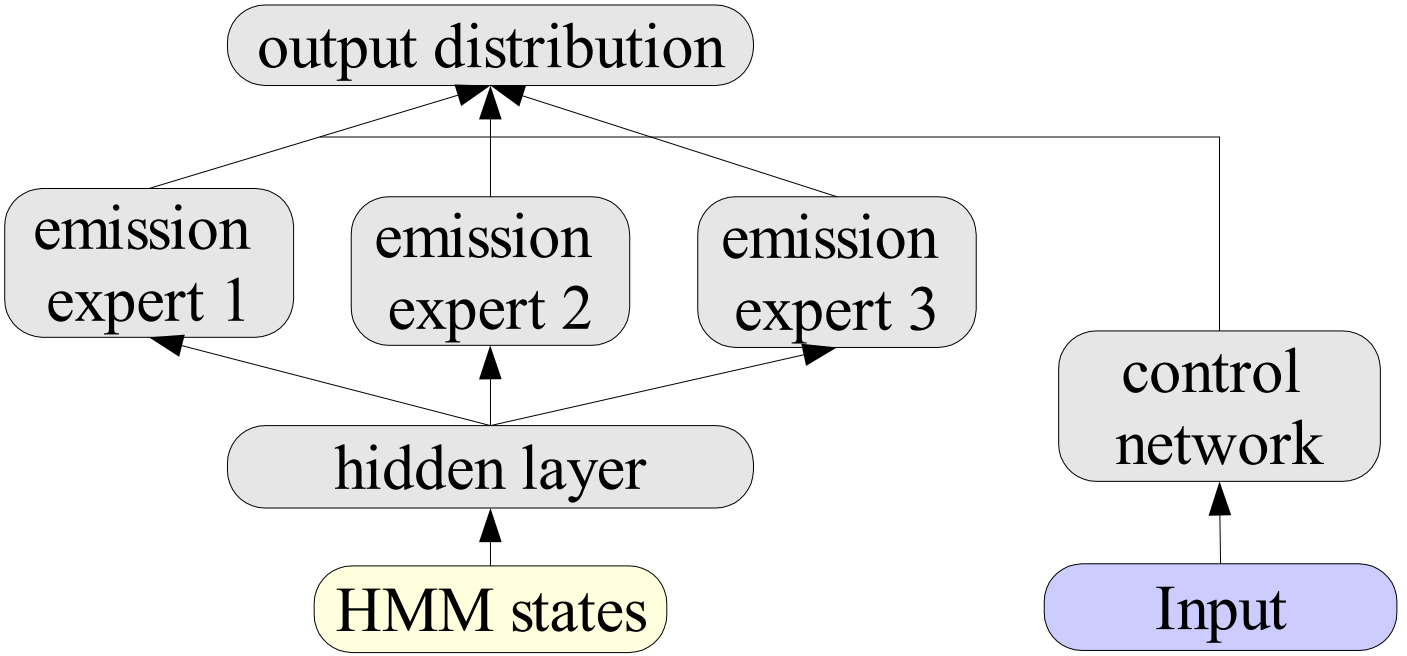
Example: Mixture of Emission Experts

Modulate emission probabilities with additional input I:

$$e_{iX} = P(i, X, I) = \sum_{j=1}^{n} \lambda_j(i, X, I) P_j(i, X, I)$$

Sometimes λ_j doesn't depend on X or i, and P_j not on I:

$$e_{iX} = P(i, X, I) = \sum_{j=1}^{n} \lambda_j(i) P_j(i, X)$$



Learning a Mixture of Emission Experts

- Given parameter settings, observations, and input I, hybrid architectures reduce to one HMM.

- Sequence likelihood can be computed, and its gradient backpropagated through the NN.

- If the value of I is unknown, it can be learned by Bayesian inversion:

$$P(I|O,w)=\frac{P(O|I,w)P(I)}{P(O|w)}$$

- Parameter probability is then (with independent observations):

$$P(w|O)=\frac{\left[\prod_O P(O|w)\right]P(w)}{P(D)}$$

  ... which can be optimized by gradient descent. P(O|w) and derivatives can be computed by Monte Carlo.

- Book: example on immunoglobulins

Protein Secondary Structure Prediction with BRNNs

- Protein secondary structure prediction: learn a translation from amino acid strings to structural category strings.
- Sequences are spatial → BIOHMMs are an alternative to a fixed width input window.
- BIOHMMs have undirected loops → need junction tree for evidence propagation. Inference complexity $O(n^3)$ per time step.
- Speedup: reparameterize with feedforward and recurrent networks → Bidirectional Recurrent Neural Nets (BRNN)

BRNN Basics

- At each position t in a protein sequence, output membership probabilities o_{it} in secondary structure classes $i=1,2,3$.

- Functional form:

$$O_t = \eta(F_t, B_t, I_t), \qquad o_{it} = \frac{\exp(net_{i,t})}{\sum_{l=1}^{3} \exp(net_{l,t})}$$

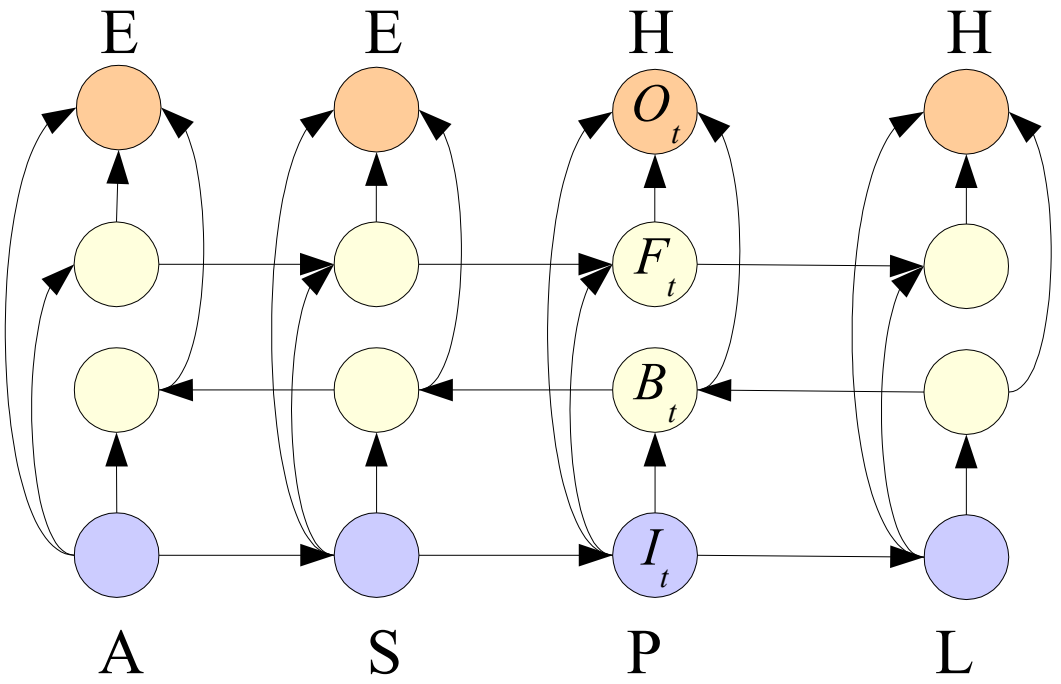
- I_t: external input at time t. $I_t \in R^k$; at simplest an orthogonal binary encoding of one amino acid.

- η: calculated by a neural network

BRNN Basics, continued

- F_t: forward context, information to the left of time t. Compare to internal state in normal RNNs.
- B_t: backward context, information to the right of time t.
- Recurrent bidirectional equations: $F_t = \phi(F_{t-1}, I_t)$
  $B_t = \beta(B_{t+1}, I_t)$
- ϕ and β are realized by two neural networks.
- Boundary conditions: $F_0 = B_{N+1} = 0$.

BRNN Basics, continued



Output: secondary structure symbols

Membership probabilities

Forward Markov chain

Backward Markov chain

Input

whole protein sequence

- Global mapping is like in BIOHMMs but internal relationships are deterministic, not probabilistic. (Overall model is probabilistic.)
- Connection weights do not change with t (weight sharing; reduces overfitting risk).

Training BRNNs

- The graph is acyclic, so nodes can sorted and unambiguously processed.
- The prediction algorithm uses the network “unrolled in time”, updates F_t left to right, and B_t right to left. Complexity $O(NW)$.
- Learning is maximum likelihood estimation: $$l = \sum_{sequences} \sum_{t=1}^{N} z_{i,t} \log o_{i,t}$$
- Optimized by gradient ascent. Some specialities needed because of noncausal dependencies. Online weight updating used for speedup. Heuristic adaptive learning rate.
- Propagated error decays exponentially. Remote information cannot be learnt effectively. In practice, the total effective window size is about 31. Shortcut connections can be used.