CMSC423: Bioinformatic Algorithms, Databases and Tools Lecture 18

Gene finding

Admin

- Project 2- listed on the website
- Midterm answers
- Other questions?

Gene finding as machine learning

- Main question: does the ORF look like a gene?

- Given a set of examples – genes we already know
- and a string of DNA (e.g. ORF)
- compute the likelihood that the ORF is a gene.
- Note: more complex than motif finding

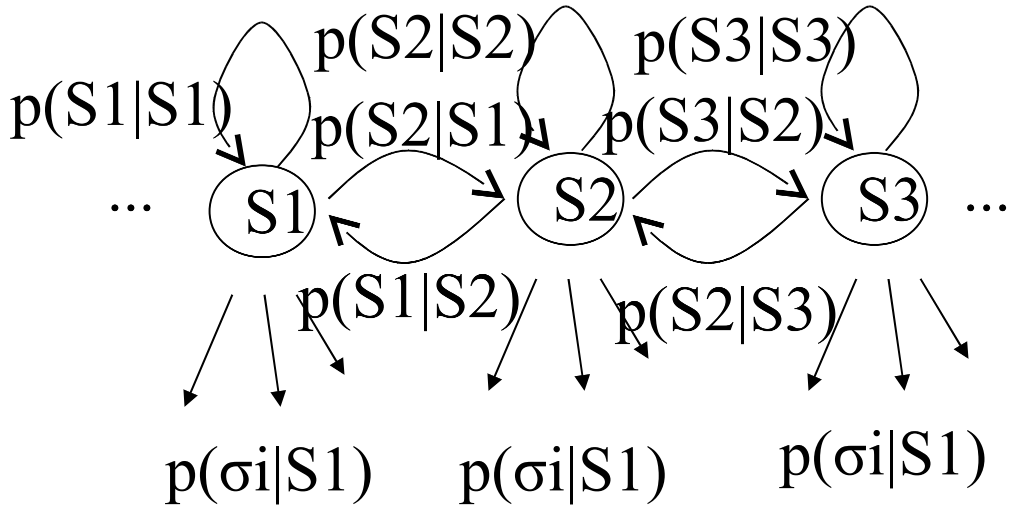
- Codon usage bias – not all codons for a same amino-acid are equally likely
- K-mer (e.g. 6-mer) frequencies (instead of single-base frequencies in motif finding)

Bacillus anthracis codon usage

UUU	F	0.76	UCU	S	0.27	UAU	Y	0.77	UGU	C	0.73
UUC	F	0.24	UCC	S	0.08	UAC	Y	0.23	UGC	C	0.27
UUA	L	0.49	UCA	S	0.23	UAA	*	0.66	UGA	*	0.14
UUG	L	0.13	UCG	S	0.06	UAG	*	0.20	UGG	W	1.00
CUU	L	0.16	CCU	P	0.28	CAU	H	0.79	CGU	R	0.26
CUC	L	0.04	CCC	P	0.07	CAC	H	0.21	CGC	R	0.06
CUA	L	0.14	CCA	P	0.49	CAA	Q	0.78	CGA	R	0.16
CUG	L	0.05	CCG	P	0.16	CAG	Q	0.22	CGG	R	0.05
AUU	I	0.57	ACU	T	0.36	AAU	N	0.76	AGU	S	0.28
AUC	I	0.15	ACC	T	0.08	AAC	N	0.24	AGC	S	0.08
AUA	I	0.28	ACA	T	0.42	AAA	K	0.74	AGA	R	0.36
AUG	M	1.00	ACG	T	0.15	AAG	K	0.26	AGG	R	0.11
GUU	V	0.32	GCU	A	0.34	GAU	D	0.81	GGU	G	0.30
GUC	V	0.07	GCC	A	0.07	GAC	D	0.19	GGC	G	0.09
GUA	V	0.43	GCA	A	0.44	GAA	E	0.75	GGA	G	0.41
GUG	V	0.18	GCG	A	0.15	GAG	E	0.25	GGG	G	0.20

A more general solution

- Hidden Markov models
- States, transition probabilities, emission probabilities



- $p(S_i|S_j)$ – probability of transitioning to state i if we are in state j
- $p(\sigma_i|S_j)$ – probability of emitting symbol σ_i if we are in state j

Why "Hidden"?

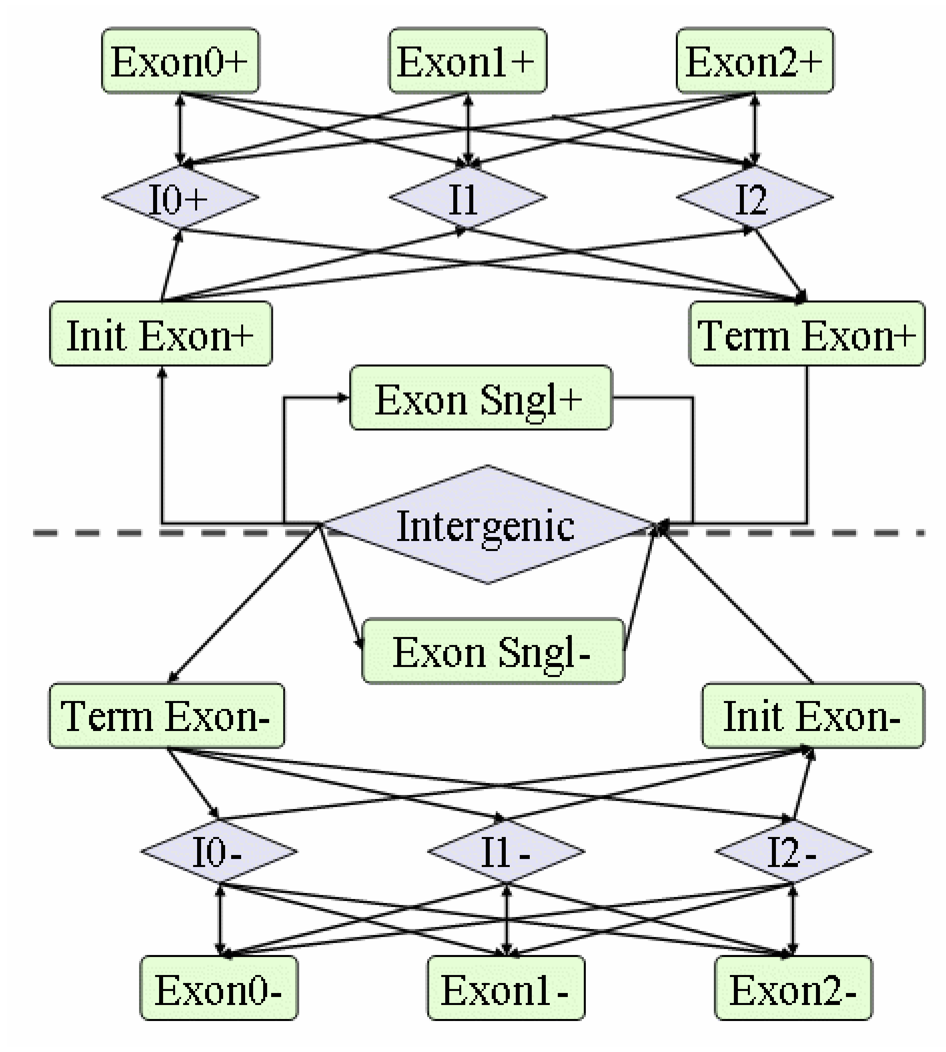
- Observers can see the emitted symbols of an HMM but have no ability to know which state the HMM is currently in.
- Thus, the goal is to infer the most likely hidden states of an HMM based on the given sequence of emitted symbols.

HMM Parameters

- Σ: set of emission characters.
  - Ex.: Σ = {H, T} for coin tossing
  - Σ = {1, 2, 3, 4, 5, 6} for dice tossing
  - Σ = {A, C, T, G} for DNA

- Q: set of hidden states, each emitting symbols from Σ.
  - Q={Fair,Biased} for coin tossing
  - Q={gene, not gene} for bacteria
  - Q={exon, intron, intergenic) for eukaryotes

GlimmerHMM model



Questions we can ask with HMMs

- Given an observed sequence of emitted characters (a string of DNA), what is the most likely sequence of states that generated the observed sequence?
  - given a string of DNA and the model, break it up into genes
  - solved by Viterbi algorithm

- Given an observed sequence of emitted characters, what is the most likely state the model was in at time t?
  - given a string of DNA, how likely is it that a certain location is inside a gene?
  - solved by forward-backward algorithm

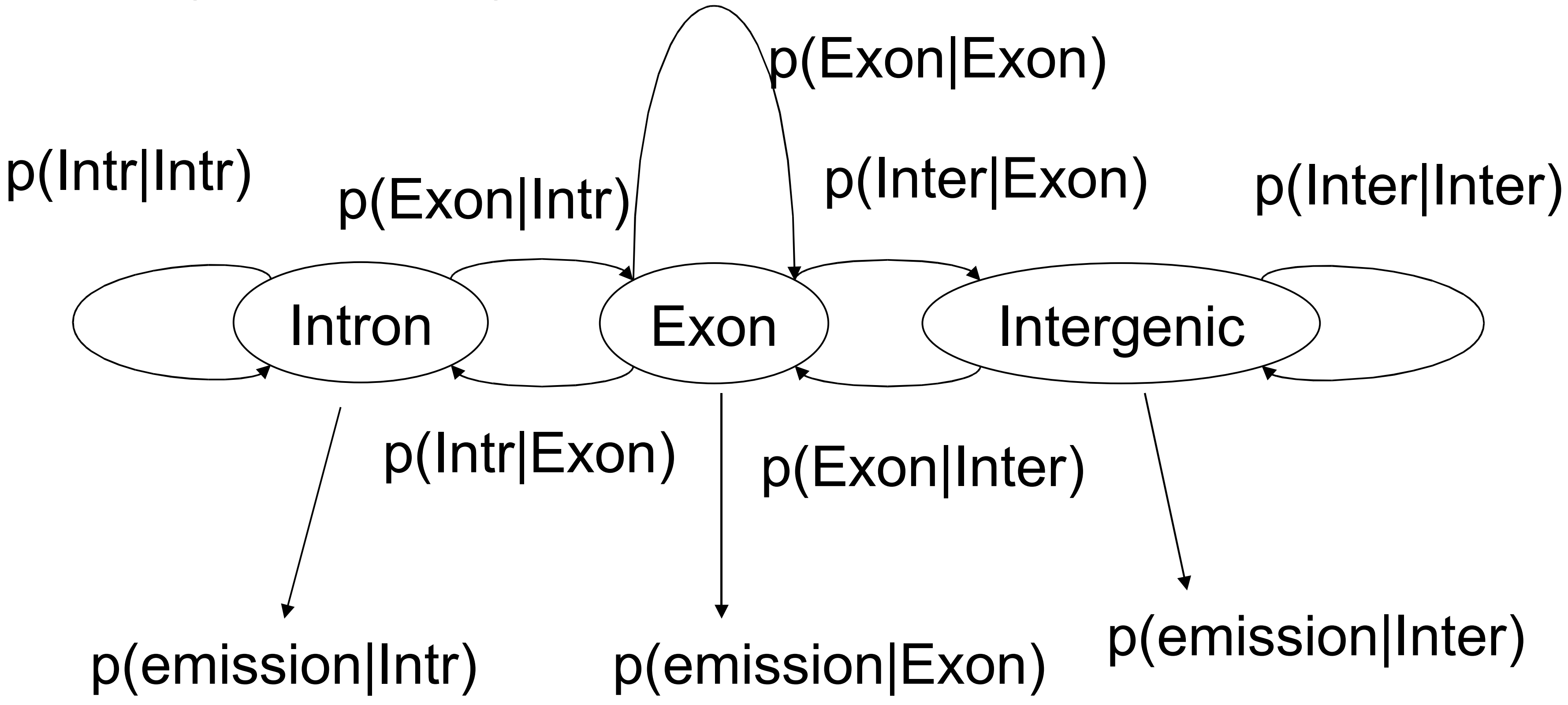
Training – the key to HMMs

- So far we've assumed that all probabilities are known.
- The training problem:
  - given an HMM (just the states and connections)
  - given several examples (e.g. known genes and intergenic regions)
  - compute the transition and emission probabilities

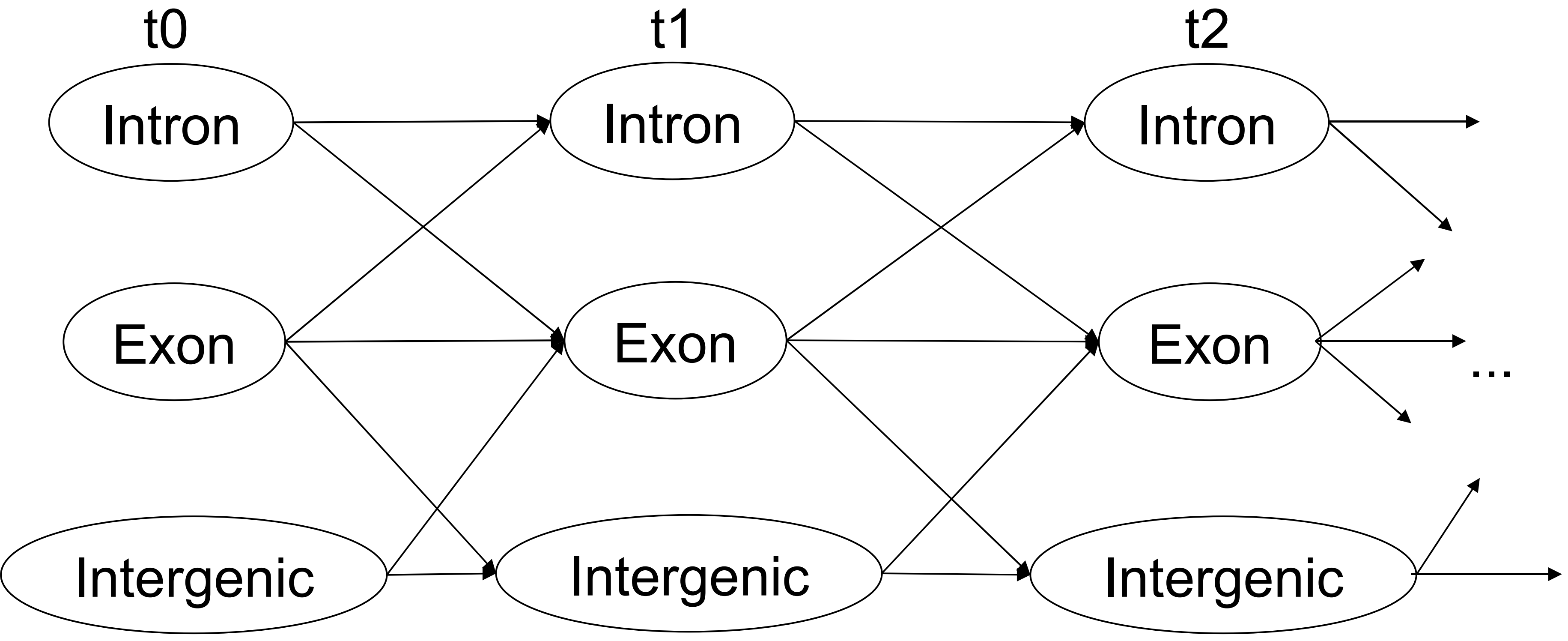
- Training is difficult!!
- Baum-Welch algorithm – iterative optimization
  - start with estimates of the probabilities
  - run model with training data
  - re-estimate probabilities based on performance on training data

Viterbi algorithm

- Given an HMM and an output string, compute the most likely path through the HMM that would result in the given string



Viterbi algorithm



Observations:

x0 x1 x2

maximize $\prod_{0}^{n} e_{statej}(x_j)\, p(state_j \mid state_{j-1})$ over all possible state paths

dynamic programming algorithm

Viterbi algorithm

- S(k,i) – most likely path for $x_0 .. x_i$ ends in state k
- $S(l, i + 1) = \max_k \{ S(k, i) * p(l|k) * p(\text{emission of } x_{i+1}|l)\}$
  $= p(\text{emission of } x_{i+1}|l) * \max_k \{S(k,i) * p(l|k)\}$

- The optimal path is found by back-tracking
- Note: Viterbi is equivalent to finding longest path in a graph
- Implementation problem: underflow – many products of very small values
- Solution: work in log-space
  - instead of probabilities use logarithm of probabilities
  - instead of products use sums

Forward-backward algorithm

- Given an HMM and an output string of length n, what is the probability that the HMM was in state k at time i < n?

- Similar dynamic programming as Viterbi however done twice:
  - from t0 to ti (forwards)
  - from tn to ti (backwards)
- In Viterbi recurrence replace max with ∑
  - likelihood is a sum of probabilities - all possible paths that go through state k at time i

Notes on training an HMM

- Gene finder output
  - a set of predictions (exon, intron, intergenic, etc.)
  - a probability (likelihood) for each prediction
- In addition to setting parameters for the model you also need to pick a threshold – how high should the probability be before you "believe" it.

Picking the "right" threshold

- Cross-validation (hold-out cross validation)
  - divide training set into Training and Hold sets
  - train in "Training"
  - test result on "Hold" – adjust threshold until results look best

- k-fold cross-validation
  - divide training set into K sub-sets
  - train on K-1 sets and test on one of them
  - repeat for different choices of "test" set

Assessing accuracy

- Confusion matrix: compare predictions to truth

truth

prediction

	Gene	Not-gene
Gene	True positive	False positive Type I error
Not-gene	False negative Type II error	True negative

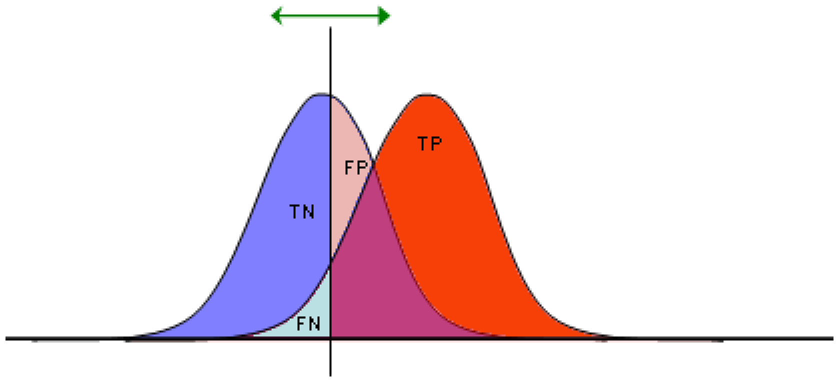
Measures of accuracy

- Sensitivity (Sn, recall) – TP/TP+FN
- Specificity (Sp) – TN/TN+FP
- Precision – TP/TP+FP

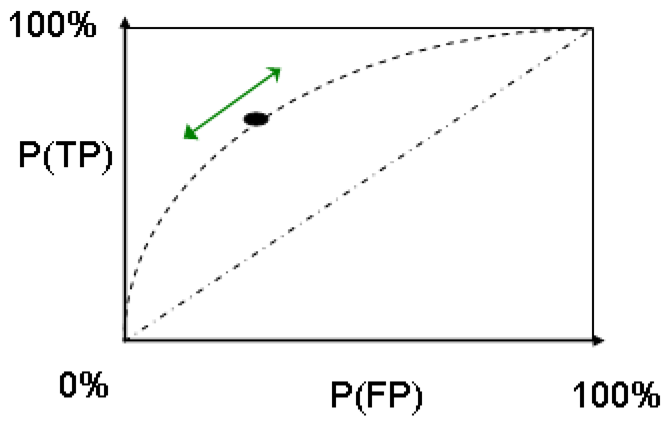
- Usually reported as (Sp, Sn), or (precision, recall).
- Also:
  F-score = 2*Precision*Recall/(Precision + Recall)

TP	FP
FN	TN

Receiver operating characteristic



TP	FP
FN	TN
1	1


P(TP)
P(FP)