Bioinformatics

Computer science: graphs, HMMs

Martin Saturka

http://www.bioplexity.org/lectures/

EBI version 0.6

Computer science basics

From mathematical logic to complexity theories to software.

- data and algorithms work, roots in pure mathematics
- applications practical data stores and computation systems

Distinctions

- mathematical background
  - theories of data and computations
  - logic and recursion, Turing machines, calculi
- data and algorithms
  - data structures and algorithm development
  - complexity descriptions and charcterizations
- programming, computations
  - imperative (common) vs. declarative programming
  - functional, logic, constraint technics, templates

Logic

exact systems and their descriptions

theories metatheories

- propositional calculus - 'outer'
  - pseudoparadoxes: "this sentence is false"
- predicate (first order) calculus - 'inner'
  - quantifiers: axioms vs. deduction rules

- symbols, terms, formulae
- axioms, deduction rules, proofs

- theorems - provability, models - truth
- recursion and compuations, relation systems

Recursion

From the beginning to the end.

- self-similarity
  - function calls on reduced data sets
  - efficient for tail recursion
- inductive computing
  - to keep up an invariant - still valid
  - to go through, approaching a target
- example: square-2 covering
  - invariant: #black = #white - 2
  - decreasing the amount of free cells
- limitations
  - halting problem

Probability

stochastic systems: probability, degree, uncertainty

- probability space $(\Omega, \mathcal{F}, P)$
  - Ω - sample space of possible events
  - $\mathcal{F}$ - σ-algebra: complements and countable unions
  - P - probability measure
  - random variables - measurable functions

- density / mass functions
  - uniform: $1/(b-a)$, $1/n$
  - memoryless (exponential, geometric): $\lambda e^{-\lambda x}$, $(1-p)^{n-1}p$
  - normal: $1/(\sigma\sqrt{2\pi}) \cdot \exp[-(x-\mu)^2/(2\sigma^2)]$
  - homoschedastic variables $\rightarrow$ normal distribution

Conditional probability

Probabilities under some prior knowledge

- e.g. probabilities of symbols on a string on position n when symbols on positions $n-k$, ..., $n-1$ are known

- $\Pr(A|B) = \Pr(A \cap B) / \Pr(B)$
  - take the part of universe where B is valid as the new whole universe and compute the probability of A therein
  - $\Pr(A|B) = \Pr(A)$ iff A and B are independent random events

- Bayes' theorem
  - $\Pr(A_1|B) = \Pr(B|A_1) \cdot \Pr(A_1) \ / \ \Sigma_i[\Pr(B|A_i) \cdot \Pr(A_i)]$
    - $\Pr(A|B)\ \Pr(B) = \Pr(A \cap B) = \Pr(B|A)\ \Pr(A)$

explorative vs. confirmative statistics

- explorations: system characterizations
  - mean, median, variance, correlation, etc.
  - data-mining, search for interavariable relations
- confirmations: hypothesis testing
  - p-value - probability of wrong rejecting the null hypothesis
  - normal distribution of errors
    - parametric tests: based on means and variances
  - continuous distributions
    - (robust) non-parametric tests: based on quantiles
- multivariate tests
  - elimination of the growing probability of wrong H_0 rejecting
  - resempling: bootstrap, permutation tests
  - Bonferroni corrections

complexities of structural descriptions

- Shannon entropy: $H = -k \sum_1^n p_i \log p_i$
  - the least amount of bits for an encoding
  - axioms:
    - defined and continuous
    - growing in variables of the discrete uniform distribution
    - branching into subsystems

- Fisher information: $I(\theta) = -\int \frac{\partial^2}{\partial \theta^2} \ln f(X;\theta)\, df(X;\theta)$
  - definition: the variance of the score
  - suitability of an observable with respect to an unobservable

- Kolmogorov complexity
  - definition: the shortest specification of an object
  - Berry pseudoparadox:
    - ”the smallest positive integer not definable in 11 words”

Complexity

computational complexity

- asymptotic notation:
  - $f(n) \in O(g(n))$ ... upper bound
    - $\limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \infty$
  - $f(n) \in \Theta(g(n))$ ... tight bound
    - $0 < \liminf_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| \leq \limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \infty$
- Computation resources
  - time complexity
  - space complexity
- Complexity classes
  - polynomial: linear, loglinear, quadratic, ...
  - NP i.e. 'non-deterministic polynomial'
  - other: probabilistic, exponential, etc.

Graphs

- $G = (V, E)$
  - V vertices, E edges between the vertices
  - connected, un/directed graphs, weighted graphs
  - degrees of vertices of a graph
- $n \times n$ matrices
  - adjacency matrix - a_{ij} count of edges between i, j
  - degree matrix - non-zeros on diagonal: degree of a_{ii}
  - Laplacian matrix $l_{ij} = d_{ij} - a_{ij}$
- basic notions
  - (simple) path - a sequence of adjoint vertices
  - cycle - a closed path
  - tree - a graph without a cycle
  - spanning tree - with all the vertices
- Euler paths
  - to meet all the edges without an edge repeat
  - iff: all (-2) the vertices with even degrees

Algorithms

standard graph algorithms

- traversal
  - a simple pass through a graph: DFS, BFS
- minimum spanning trees
  - Borůvka's, Jarník's/Prim's, Kruskal's
- shortest paths
  - Dijkstra's, Bellman-Ford, Floyd-Warshall algorithm
- network flows
  - Ford-Fulkerson, MPM, Goldberg
- sort
  - topological, scheduling

Traversal

- BFS - breadth-first search
  - usage of a FIFO queue
  - start: put a (root) vertex into the queue
  - next: repeat until the queue is empty
    - take the first vertex from the queue
    - put all its (free) adjecent vertices into the queue
- DFS - depth-first search
  - usage of a LIFO stack
  - start: put a (root) vertex into the stack
  - next: repeat prolong / backtrack
    - push first available adjacent vertices into the stack
    - remove vertices when there's no (free) prolongation
- complexity
  - time: $|V| + |E|$ for both
  - space: DFS more efficient than BFS
  - DFS usually better for heuristics

Components

- graph structure
  - connected components
  - biconnected components
    - without a separation vertex, iff all edge pairs in a cycle
  - strongly connected components
    - connected ($\leftrightarrow$) components of a directed graph
- connected components
  - just traversals of a graph until met all the vertices
- biconnected components of graph G
  - DFS on the graph G $\rightarrow$ proxy graph F
    - vertices of F are edges of G
    - connected components of F are the biconnected ones of G
- strong components
  - subsequent DFS on the reverted graph (G^R)
  - DFS on G from its sink vertex - a source vertex of G^R
  - result is a DAG of strongly connected components of G

Shortest paths

- all vertices: Floyd-Warshall algorithm
  - use of adjacency matrix, without negative cycles
  - time complexity is $\Theta(|V|^3)$
  - search the shortest paths through limited sets of other verteces
  - consequently compare distances through the newly added vertices
- single-source: Dijkstra's algorithm
  - for positive weights, complexity $O(|E| + |V|log|V|)$
  - set all the verteces as open
  - set infinite distance for all but the source vertex
  - start with the source vertex, set its distance to 0
  - repeat:
    - take the open vertex with least distance
    - check distances of paths to verteces from the taken vertex
    - set the taken vertex closed
- the algorithms make use of separation of the problems onto smaller data subsets

Divide et Conquer

- Fermat's principle
  - the optical path is the extremal one
  - subpaths are extremal too $\rightarrow$ accessible to decompositions
- shortest path
  - subsequently optimal solutions up to limited data sets
  - the 'divide' part is linear, i.e. the recursion easily iterated
- common scheme
  - forward search for optimal sub-solutions
  - backward reconstruction of the final solution

Recursion examples

recursion iteratization

- tail recursion $\rightarrow$ simple iteratization
  - Dijkstra's algorithm
- complex recursion schemes hard to iteratize
  - median search

- median search - linear time
  - medians of n-tuple subsets
  - recursively medain of the n-tuple medians
  - the current median based separation of the current set
  - recursively search in the selected subset

- remeber: many concrete recursion algorithms lack efficacy

Markov systems

the future depends on the past through the present

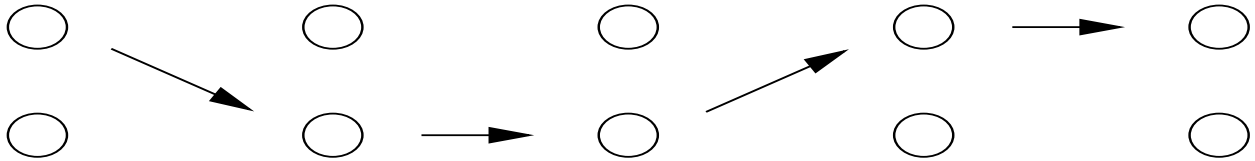
$$\Pr(X_{n+1} = x | X_n = x_n, \ldots, X_0 = x_0) = \Pr(X_{n+1} = x | X_n = x_n)$$

- i.e. future and past states are conditionally independent
  - possible domain separations
  - physics - time, valid
  - other - approximations

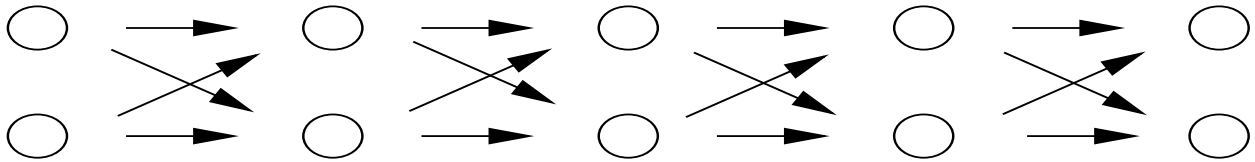
- generalizations
  - N-th order Markov process - relevant last N timestamps
  - parametrization: time, sequences, etc.

Hidden systems derivation

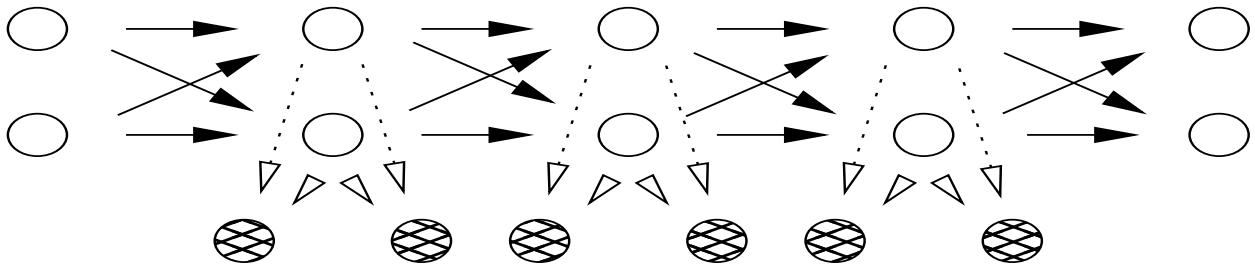
- deterministic system (visible states)



- stochastic system (visible states, probabilities)



- stochastic hidden process (states, probabilities, outputs)



Simple examples

hidden inner states - known outer manifestations

static - an examination:

- inner: understanding a topic
- outer: answers on questions

dynamic - an islamist behavior:

- inner: being offended current day
- outer: some (number of) burned flags
- a case: 'flames'-'flames'-'flames'-'nothing'-'flames'
  - for a high transition probability of 'offended' $\rightarrow$ 'offended', it should be rather probable 'being offended' all the time (e.g. just out of flags for the one day)

HMMs on biopolymers

- detection of a protein domain
  - inner: inside / outside of the domain motif
  - outer: actual aminoacyls
- sequence profile matching
  - inner: insert / delete / match to a position of the profile
  - outer: actual aminoacyls
- search for gene sequences inside genomes
  - inside / outside of a gene
  - actual nucleotides

The inner states can be / need not be parametrized.

like 'being offended' vs. 'being offended in a given day'

HMMs themselves

Finite automaton extension

- finite automaton
  - states, transition function
- FA inner states
  - structural states - lesser amount of states
    - ⟳ outside a gene $\leftrightarrow$ inside a gene ⟳
  - linearized states - greater amount of states
    - $\rightarrow$ states for position n $\rightarrow$ states for position $n+1$ $\rightarrow$
- linear passing through the inner states
  - conditional independence parametrization: time, sequence
- linear outer sequences of visible events
  - stochastic regular grammars

HMMs usage

three types of problems:

- most probable path for a HMM, sequence pair given
  - one concrete derivation path
  - sequence alignment
- probability of a given output sequence on a HMM
  - overall probability of a sequence
  - domain detection
- motif HMM parameter profiling for a sequence set
  - iterative parameter adjusting
  - HMM training

HMM algorithms outline

paths processing:

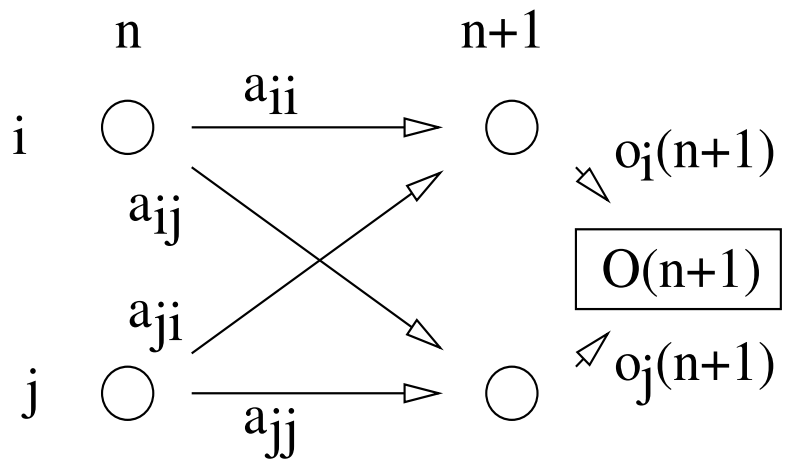
- assumption of Markovian processes
  - $\longrightarrow$ dynamic programming
  - tail recursion, simple linear iteration
- passing through the HMM states
  - computing maximal or overall probabilities along the steps
  - maximal derivation: Viterbi algorithm
  - overall derivation: forward algorithm
- the forward algorithm used as a part of HMM profiling

HMM algorithms step

stepwise probability products

- invariant assumption on each (relevant) inner state:
  - maximal / overall probabilities for inner states are set
- take next observable object
  - for each currently accessed inner state
    - multiply its probability for each accessible state
      with the transition probability times the output probability
  - for each recently accessed inner state
    - sum the new probabilities (forward algorithm)
    - choose the greatest new probability (Viterbi algorithm)

HMM probabilities



i, j	inner states
n	outer string postions
$O(\cdot)$	outer string symbols
$a_{\cdot\cdot}$	transition probabilities
$o_{\cdot}(\cdot)$	output probabilities

- auxiliar probabilities (under a given HMM)
  - $\Pr_{ii}(n+1) = \Pr_i(n) \cdot a_{ii} \cdot o_i(n+1)$
  - $\Pr_{ij}(n+1) = \Pr_j(n) \cdot a_{ji} \cdot o_i(n+1)$
- most probable derivation (profile HMMs)
  - $\Pr_i(n+1) = \max(\Pr_{ii}(n+1), \Pr_{ij}(n+1))$
- overall derivations (domain HMMs)
  - $\Pr_i(n+1) = \Pr_{ii}(n+1) + \Pr_{ij}(n+1)$

Path algorithms for HMMs

easy to understand algorithms

- start with all (initial) inner states
  - initial states for parametrized FA
- set either initial or initial $\times$ output probabilities to the states
  - initial probabilities for special 'quiet' initial states
- complexity: $|\text{inner states}|^2 \times$ string length
  - inner states for a single step in the case of linearized states

update sequentially inner states:

- Viterbi algorithm
  - compute new maximal probabilities
- Forward algorithm
  - compute new overall probabilities

Search results

- each sequence has a derivation probability, which ones are those with a found domain?

- statistics on NLL scores
  - NLL = $-\log(Pr(\text{overall}))$
  - rather low probabilities $\rightarrow$ better to compute with log
  - log values roughly under normal distribution, with outliers
  - Z-scores, i.e. distances in units of standard deviations
  - smoothened deviations for sequences of similar lengths

- iterative outlier search
  - compute smoothened standard deviations
  - take out sequences with high (e.g. > 4) Z-scores
  - all the outliers are taken as with successful search

HMMs reestimations

- start of adjusting the transition probabilities
  - under a given sequence and running HMM parameters
  - forward probabilities computation:
    - $\alpha_i(n) = \Pr(O(1,...,n), inner(n) = i | HMM)$
      probability of being in the state i in the step n
      and with outputing the initial part of the string
  - backward probabilities computation:
    - $\beta_i(n) = \Pr(O(n+1,...,N) | inner(n) = i, HMM)$
      probability of outputing the terminal part of the string,
      starting from the state i in the n-th step
- normalization for reading $n+1$-th outer symbol
  - it has to pass from a state i into a state j
- $\zeta_{i,j}(n) = \alpha_i(n) \cdot a_{ij} \cdot o_j(n+1) \cdot \beta_j(n+1)$
- $\xi_{i,j}(n) = \zeta_{i,j}(n) / \sum_i \sum_j \zeta_{i,j}(n)$

- overall expected number of transitions i to j:
  $\xi_{i,j} = \sum_n \xi_{i,j}(n)$
  - to be continued

HMMs profiling

EM - expectation maximization

- start with apriori probability parameters
- subsequently use all the relevant output strings
  - stop the adjusting when small changes
- iterate the parameters adjusting
  - compute overall expected numbers of transitions and analogically symbol outputs from particular inner states
  - normalize to have total unite probabilites:
    $\Pr(i,j) = \xi_{i,j} / \sum_j \xi_{i,j}$
    $\Pr(s|i) = o_i(s) / \sum_s o_i(s)$
    - for i, j inner states, s possible output symbols
    - just rate probabilities of having the-vs.-any symbol outputs
- the new probabilities serve as the new parameters
  - backward only probabilities usage for linearized HMMs

Profiling problems

optimization class of algorithms

- local optimum
- $\longrightarrow$ to use a system of profilings
  - probabilistic climbing methods: simulated annealing, etc.

- overfitting
- $\longrightarrow$ to add noise and/or apriori knowledge
  - regularization to avoid biases of small training set cases

HMMs limitations

when to use something more profound / complex

- protein profiles - relatively sufficient
- gene finding - relatively sufficient
- RNA folding - highly insufficient
  - long-distance interactions

- MCMC and hierarchical HMMs
  - for protein domain characterizations
- Bayesian networks
  - for complex system development
- Stochastic context-free grammars and Covariance models
  - for RNA matching and folding predictions

Items to remember

Nota bene:

graphs, dynamic programming

- HMMs
  - definition
  - examples
- Algorithms
  - paths
  - limitations