Bayesian Evolutionary Computation, Importance Sampling, Sequential Monte Carlo, Particle Filters, and Active Learning

2001 Bioinformatics Course
Supplement

SNU Biointelligence Lab
http://bi.snu.ac.kr/

Evolutionary Computation: A (Very) Brief Introduction

Outline of Evolutionary Computation

- A Brief History of Evolutionary Computation
- Properties of Evolutionary Algorithms
- Terminology
- Evolutionary Operators
- Simple Evolutionary Algorithm
- Bayesian Evolutionary Algorithms

Evolutionary Computation

- Evolutionary computation is inspired from natural selection and evolution in the nature.
- 1960s, John Holland: Genetic Algorithms
  - To study the phenomenon of adaptation as it occurs in nature (not to solve specific problems)
- 1960s, Fogel, Owens, and Walsh: Evolutionary Programming
  - To find finite-state machines
- 1960s, Rechenberg: Evolution Strategies
  - A method used to optimize real-valued parameters for devices

Properties of Evolutionary Algorithms

- Global population-based random search algorithms
- Simulate the principle of evolution (survival of the fittest).
- Maintain a population of potential solutions (individuals) through repeated application of some "evolutionary" operators.
- Yield individuals with successively improved fitness, and converge, hopefully, to the fittest individuals representing optimum solutions.
- Although GS, EP and ES were originally developed for different purposes, they all have been successfully applied to solve global optimization problem.

Terminology

- *Chromosome*
  - Candidate solution to a problem
  - Often encoded as a bit string
- *Genes*
  - Single bit or short blocks of adjacent bits that encode a particular element of the candidate solution
- *Crossover*
  - Exchanging genetic material between two single chromosome parents
- *Mutation*
  - Flipping the bit at randomly chosen locus

Evolutionary Operators

- Selection
  - Select chromosomes in the population for reproduction.
  - The fitter the chromosome, the more times it is likely to be selected to reproduce
- Crossover
  - Randomly choose a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring
- Mutation
  - Randomly flips some of the bits in a chromosome.

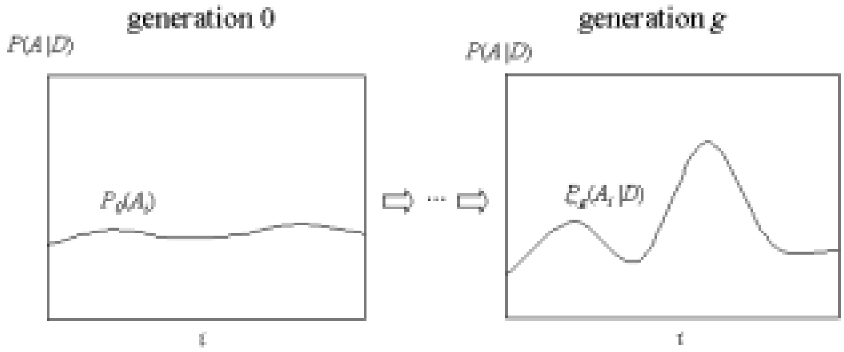
A Simple Evolutionary Algorithm

1. Start with a randomly generated population
2. Determine the fitness of each individual in the population
3. Repeat the following steps
   a) Perform crossover with probability p_c
   b) Perform mutation the two offspring with probability p_m
   c) Determine the fitness of each individual
   d) Perform selection

   until some stopping criterion applies

Bayesian Evolutionary Computation

- A probabilistic model of evolutionary computation for learning and optimization [Zhang, 1999]
  - [Zhang & Shin, 2000] [Cho & Zhang, 2000][Shin, Cho, & Zhang, 2001][Lee & Zhang, 2001]
- Explicitly estimate the posterior distribution of the individuals and then sample offspring from the distribution



BEC: History and Current Issues

- Evolutionary computation as Bayesian inference [Zhang & Muehlenbein, 1993]
- Distribution estimation algorithms [Baluja & Caruana, 1995][Muehlenbein & Paass, 1996]
- Bayesian genetic programming [Zhang, 1995, 1997]
- Evolving neural trees by BEA [Zhang & Joung, 1998]
- Bayesian evolutionary algorithms for learning and optimization [Zhang, 1999]
- Helmholtz machines for distribution estimation [Zhang & Shin, 2000] [Shin, Cho, & Zhang, 2001]
- Probabilistic PCA for estimation [Cho & Zhang, 2001]
- Connections to particle filters [Lee & Zhang, 2001]
- Evolving neural trees by BEC with PPCA [Cho & Zhang, 2001]

Importance Sampling

Outline of Importance Sampling

- Generalized Importance Sampling
- Bayesian Importance Sampling
- Sequential Importance Sampling
- Choice of Proposal Distribution
- Selection (Resampling)
- Sequential Monte Carlo Methods (Particle Filters)
  - Brief History
- Particle Filter Algorithm

Generalized Importance Sampling (1/2)

- We are interested in computing expectation $E_{P(x)} f(x)$
- Draw independent points from a simpler "proposal" distribution Q
- Weight these points by $w(x) = P(x) / Q(x)$ to obtain a "fair" representation of P

$$E_{Q(x)} f(x)w(x) = \sum_{x\in X} [f(x)w(x)]Q(x) = \sum_{x\in X} \left[f(x)\frac{P(x)}{Q(x)} \right] Q(x)$$

$$= \sum_{x\in X} f(x)P(x) = E_{P(x)} f(x)$$

- If *Q*=*P*, above formula reduces to simple Monte Carlo estimation formula.

Generalized Importance Sampling (2/2)

- Importance sampling is effective when *Q* approximates *P* over most of the domain.
- There are no absolute requirements for how well *Q* approximates *P*, except that *Q* must not zero anywhere *P* is non-zero.
- It fails when *Q* misses high probability regions of *P* and systematically yields samples with small weights.
- To overcome this problem it is critical to obtain data points from importance regions of *P*.
- Explicitly searching for significant regions in the target distribution *P*.

Problem Formulation: Dynamic State Space Model

- Transition equation $p(x_t \mid x_{t-1})$ and measurement's equation $p(y_t \mid x_t)$
  - $x_t \in R^{n_x}$ denotes the states (hidden variables, parameter)
  - $y_t \in R^{n_y}$ observations at time t
- Goal
  - Approximate the posterior $p(x_{0:t} \mid y_{1:t})$
  - One of its marginals, the filtering density $p(x_t \mid y_{1:t})$

Bayesian Importance Sampling

- Impossible to sample directly from the posterior $p(x_{0:t} \mid y_{1:t})$
- Sample from an easy-to-sample, proposal distribution $q(x_{0:t} \mid y_{1:t})$

$$E(g_t(x_{0:t})) = \int g_t(x_{0:t}) \frac{p(x_{0:t} \mid y_{1:t})}{q(x_{0:t} \mid y_{1:t})} q(x_{0:t} \mid y_{1:t}) dx_{0:t}$$

$$= \int g_t(x_{0:t}) \frac{p(y_{1:t} \mid x_{0:t}) p(x_{0:t})}{p(y_{1:t}) q(x_{0:t} \mid y_{1:t})} q(x_{0:t} \mid y_{1:t}) dx_{0:t}$$

$$= \int g_t(x_{0:t}) \frac{w_t(x_{0:t})}{p(y_{1:t})} q(x_{0:t} \mid y_{1:t}) dx_{0:t}$$

(cont'd)

$$= \frac{1}{p(y_{1:t})} \int g_t(x_{0:t}) w_t(x_{0:t}) q(x_{0:t} \mid y_{1:t}) dx_{0:t}$$

$$= \frac{\int g_t(x_{0:t}) w_t(x_{0:t}) q(x_{0:t} \mid y_{1:t}) dx_{0:t}}{\int p(y_{1:t} \mid x_{0:t}) p(x_{0:t}) \frac{q(x_{0:t} \mid y_{1:t})}{q(x_{0:t} \mid y_{1:t})} dx_{0:t}}$$

$$= \frac{\int g_t(x_{0:t}) w_t(x_{0:t}) q(x_{0:t} \mid y_{1:t}) dx_{0:t}}{\int w_t(x_{0:t}) q(x_{0:t} \mid y_{1:t}) dx_{0:t}}$$

$$= \frac{E_{q(\cdot \mid y_{1:t})}(w_t(x_{0:t}) g_t(x_{0:t}))}{E_{q(\cdot \mid y_{1:t})}(w_t(x_{0:t}))}$$

$$\overline{E(g_t(x_{0:t}))} = \frac{1/N \sum_{i=1}^{N} g_t(x^{(i)}{}_{0:t}) w_t(x^{(i)}{}_{0:t})}{1/N \sum_{i=1}^{N} w_t(x^{(i)}{}_{0:t})}$$

$$= \sum_{i=1}^{N} g_t(x^{(i)}{}_{0:t}) \tilde{w}_t(x^{(i)}{}_{0:t})$$

Sequential Importance Sampling

- Proposal distribution

$$q(x_{0:t} \mid y_{1:t}) = q(x_0 \mid y_{1:t}) \prod_{j=1}^{t} q(x_j \mid x_{1:j-1}, y_{1:j})$$

- Assumptions
  - State: Markov process
  - Observations: independent given states

(cont'd)

- We can sample from the proposal and evaluate likelihood & transition probability, generate a prior set of samples and iteratively compute the importance weights w_t

$$w_t = \frac{p(y_{1:t} \mid x_{0:t})\, p(x_{0:t})}{q(x_{0:t-1} \mid y_{1:t-1})\, q(x_t \mid x_{0:t-1}, y_{1:t-1})}$$

$$= w_{t-1} \frac{p(y_t \mid x_t)\, p(x_t \mid x_{t-1})}{q(x_t \mid x_{0:t-1}, y_{1:t})}$$

Choice of Proposal Distribution

- Minimize variance of the importance weights

$$q(x_t \mid x_{0:t-1}, y_{1:t}) = p(x_t \mid x_{0:t-1}, y_{1:t})$$

- Popular choice

$$q(x_t \mid x_{0:t-1}, y_{1:t}) = p(x_t \mid x_{t-1})$$

- Move samples towards the region of high likelihood

Selection (Resampling)

- Variance of importance ratios increases stochastically over time.
- To avoid degeneracy of the Sequential Importance Sampling algorithm, a selection(resampling) stage may be used.
- Eliminate samples with low importance ratios and multiply samples with high importance ratios.
- Associate to each sample $x^{(i)}_{0:t}$ a number of children

$$N_i \quad s.t. \sum_{i=1}^{N} N_i = N$$

Sequential Monte Carlo Method (Particle Filtering): Brief History

- Basic sequential Monte Carlo methods introduced in the automatic control field in the late 1960s.
  - Handschin and Mayne (1969) tackled the problem of nonlinear filtering with a sequential importance sampling approach
- Many variations, but based on sequential importance sampling
  - Degenerate with time
- Inclusion of a resampling stage in early 1990s
- Many variations proposed in statistical and signal processing group
- Recently: Discovery of connections between sampling-importance resampling (SIR) and evolutionary algorithms.

Particle Filter Algorithm

- Sequential importance sampling step
  - For i=1,…,N sample: $(\hat{x}_t^{(i)}) \sim q(x_t \mid x_{0:t-1}^{(i)}, y_{1:t})$

  and set: $(\hat{x}_{0:t}^{(i)}) \cong (x_{0:t-1}^{(i)}, \hat{x}_t^{(i)})$
  - For i=1,…,N evaluate importance weights up to a normalizing constant:

$$w_t^{(i)} = \frac{p(\hat{x}_{0:t}^{(i)} \mid y_{1:t})}{q(\hat{x}_t^{(i)} \mid x_{0:t-1}^{(i)}, y_{1:t})\, p(\hat{x}_{0:t-1}^{(i)} \mid y_{1:t-1})}$$

  - For i=1,…,N normalize importance weights: $\tilde{w}_t^{(i)} = w_t^{(i)}[\sum_{j=1}^{N} w_t^{(j)}]^{-1}$
- Selection (resampling) step
  - multiply/suppress samples $(\hat{x}_{0:t}^{(i)})$ with high/low importance weights $\tilde{w}_t^{(i)}$,respectively, to obtain random samples $(\tilde{x}_{0:t}^{(i)})$ approximately distributed $p(\tilde{x}_{0:t}^{(i)} \mid y_{1:t})$
- MCMC step
  - Apply a Markov transition kernel with invariant distribution given by $p(x_{0:t}^{(i)} \mid y_{1:t})$ to obtain $(x_{0:t}^{(i)})$

What's Next? Active Evolution!!

- Learning and inference in probabilistic graphical models, such as Bayesian networks, are usually time-consuming.
- Active and adaptive sampling is necessary for efficient learning and inference.
- Active selection of training examples, a.k.a. active learning, may accelerate the learning process.
- MCMC provides a sound theoretical basis of learning and evolutionary algorithms for building probabilistic graphical models, including hidden Markov models, Bayesian networks, and Helmholtz machines.
- Sequential importance sampling as active learning or evolution.
- Active evolutionary algorithms? Natural evolution seems not very active or goal-directed, but systems evolving and learning actively seems at least more intelligent and more useful from the AI point of view.