Optimizations for statistical applications II

Stat 580: Statistical Computing

- Theme: Black - White
- Printable version

Combinatorial optimization

- Consider maximizing $f(\boldsymbol{\theta})$, $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, where $\boldsymbol{\Theta}$ is a finite set of candidate solutions.

Example

Variable selection in multiple linear regression

- Available predictors $x_1, \ldots, x_p$
- Each candidate model is
$$y = \sum_{j \in S} \beta_j x_j + \varepsilon$$
where S is a subset of $\{1, 2, \ldots, p\}$.

Example

- Akaike information criterion (AIC) for selecting a model is
$$\mathrm{AIC} = n \log \frac{\mathrm{RSS}}{n} + 2|S|,$$
where RSS is residual sum of squares and n is sample size. (Alternative?)
- Best model is chosen as the minimizer of AIC.
- To represent a model, set $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ such that $\theta_j = 1$ if x_j is included in the model, 0 otherwise.
- Then $\boldsymbol{\Theta} = \{0, 1\}^p$ and $\boldsymbol{\theta}$ has 2^p possible values.
- Common strategies?

Local search

- First, define a neighborhood $N(\boldsymbol{\theta})$ for $\boldsymbol{\theta}$.
- Why? $N(\boldsymbol{\theta})$ contains candidate solutions that are near $\boldsymbol{\theta}$, and we also want to reduce the number of changes to the current $\boldsymbol{\theta}$.
- $\boldsymbol{\theta}_0 \rightarrow \boldsymbol{\theta}_1 \rightarrow \boldsymbol{\theta}_2 \rightarrow \ldots$ where each time we re-define neighborhoods as $N(\boldsymbol{\theta}_0) \rightarrow N(\boldsymbol{\theta}_1) \rightarrow N(\boldsymbol{\theta}_2) \rightarrow \ldots$
- e.g., if $\boldsymbol{\Theta} = \{0, 1\}^p$, one can define $N(\boldsymbol{\theta})$ as the set of $\boldsymbol{\theta}$'s which differ from $\boldsymbol{\theta}$ in exactly one coordinate.

Local search

- Then at iteration t, choose $\boldsymbol{\theta}_{t+1}$ from $N(\boldsymbol{\theta}_t)$ according to a certain rule:
  1. Steepest ascent: choose $\boldsymbol{\theta}_{t+1} = \text{argmax}_{\boldsymbol{\theta} \in N(\boldsymbol{\theta}_t)} f(\boldsymbol{\theta})$
  2. Ascent algorithm: choose $\boldsymbol{\theta}_{t+1} \in N(\boldsymbol{\theta}_t)$ uphill from $\boldsymbol{\theta}_t$; i.e., $f(\boldsymbol{\theta}_{t+1}) \geq f(\boldsymbol{\theta}_t)$.
- Forward, backward, stepwise selections with model selection criterion are special cases of local search.

Local search

- To avoid being trapped in local maxima, one could do:

1. Random starts local search: repeatedly run an ascent algorithm to termination from a large number of randomly chosen starting points.

1. Steepest ascent/mildest descent: set to the least unfavorable $\boldsymbol{\theta}_{t+1} \in N(\boldsymbol{\theta}_t)$.
    - If $\boldsymbol{\theta}_t$ is a local maximum, $\boldsymbol{\theta}_{t+1}$ is the one with least decrease.
    - Otherwise, $\boldsymbol{\theta}_{t+1}$ is the one with largest increase.

Example

To select a linear model to minimize AIC:

1. randomly select a set of predictors
2. at each iteration, decrease AIC by adding a new predictor or by deleting a predictor that has been selected. Continue iteration until AIC cannot be decreased
3. repeat the above steps many times, choose the model at termination that has the lowest AIC

Example: traveling salesman problem

- A salesman must visit p cities exactly once and return to his starting city, using the shortest total travel distance.

- A candidate solution is
$$\boldsymbol{\theta} = (i_1, i_2, i_3, \ldots, i_p, i_1),$$
where $i_1, i_2, \ldots, i_p$ is a permutation of $1, \ldots, p$.

- The objective function to minimize is
$$f(\boldsymbol{\theta}) = d(i_1, i_2) + d(i_2, i_3) + \cdots + d(i_p, i_1),$$
where $d(j, k)$ is the travel distance between cities j and k.

Example: traveling salesman problem

- There are in total $\frac{(p-1)!}{2}$ possible routes.
- For $p = 20$ cities, there are more than 6×10^{16} possible routes.
- To use local search, one could define $N(\boldsymbol{\theta})$ as the set of sequences that only differ from $\boldsymbol{\theta}$ at two entries.
- e.g., if $\boldsymbol{\theta} = (1, 2, 5, 4, 6, 3, 1)$, then $(1, 2, 3, 4, 6, 5, 1)$ is in $N(\boldsymbol{\theta})$.

Simulated annealing

- Can be thought of as a randomized local search algorithm.
- Uses a "temperature" parameter to control the randomness of the search.
- It starts with a high temperature, and cools down gradually to locate a global optimum, similar to annealing of metal and glass.
- Suppose we want to minimize f. The algorithm runs in stages -- all iterations with the same stage share the same temperature τ_j.

Simulated annealing

Suppose iteration t belongs to stage j.

- Step 1. Sample a candidate $\boldsymbol{\theta}^* \in N(\boldsymbol{\theta})$ according to a proposal density $g_t(\boldsymbol{\theta}|\boldsymbol{\theta}_t)$ ($g_t(\boldsymbol{\theta}|\boldsymbol{\theta}_t)$ can be different for different t). Choice of $g_t(\boldsymbol{\theta}|\boldsymbol{\theta}_t)$ as a uniform distribution may be a good candidate.
- Step 2. Let $\Delta = f(\boldsymbol{\theta}^*) - f(\boldsymbol{\theta}_t)$.
  - If $\Delta \leq 0$, set $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}^*$.
  - If $\Delta > 0$,

$$\text{set } \boldsymbol{\theta}_{t+1} = \begin{cases} \boldsymbol{\theta}^* & \text{with probability } e^{\frac{-\Delta}{\tau_j}}, \\ \boldsymbol{\theta}_t & \text{otherwise.} \end{cases}$$

Simulated annealing

- Step 3. Repeat steps 1 and 2 m_j times.
- Step 4. Update $\tau_{j+1} = \alpha(\tau_j), m_{j+1} = \beta(m_j)$ and move to stage $j + 1$.
  - $\alpha(\tau)$ governs how to cool down the temperature; e.g., $\alpha(\tau) = p\tau, 0 < p < 1$ (decreasing function)
  - $\beta(m)$ governs how long to stay at a given temperature (increasing function)

Example: image denoising

- I_0 is an image consisting of pixels $I_0(i,j), i,j = 1, \ldots, N$.
- Observe $J = I_0 + Z$, where $Z(i,j)$ is noise corrupting $I_0(i,j)$.
- Minimize $f(I) = \sum\{J(i,j) - I(i,j)\}^2 + K(I)$, where $K(I)$ is a function that is large if I has many "irregularities" (i.e., I is non-smooth).
- Idea: the true image should be close to J and contains little irregularities.

Example: image denoising

- To use simulated annealing to minimize $f(I)$:
  - 1: At each iteration, only one pixel of I can be updated (defining a neighborhood).
  - 2: Choose a pixel $I(i, j)$ and select a candidate value for it.
  - 3: Update $I(i, j)$ according to step 2 of simulated annealing and move to the next iteration.
- τ_j should slowly decrease to zero.
- m_j should be large and increasing in j
- Reheating strategies can be used to prevent being trapped in local minima at low temperatures.

Genetic Algorithms

- Loosely, it starts with *a set of* random candidate solutions, and *combine them in a special manner* as iteration goes on.
  - many explorers
  - special ways to pass information to next iterations
- Assume each solution $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ can be represented as a string of alphabets $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_c)$, where each θ_i is a symbol from a finite set such as $\{0, 1\}$, $\{0, 1, 2\}$, or $\{a, b, c\}$.
- Each possible $\boldsymbol{\theta}$ is called a chromosome.
- Each θ_i is called a gene.

Algorithm

- In general:
  - Step 1: Generate a set of P chromosome (usually in a random fashion); this is the first generation.
  - Step 2: Draw chromosomes from the current generation as parents.
  - Step 3: Produce offspring from the selected parents; the next generation is formed once P offspring chromosomes are produced
  - Step 4: Repeat steps 2 and 3 until convergence.

- Two issues:
  1. How to draw the parents
  2. How to produce offspring from the parents

Issue 1: How to draw the parents

- Define a fitness function $f(\boldsymbol{\theta})$ (usually the function to be maximized) for the chromosomes.
- Let r_j be the rank of the fitness value of the j-th chromosome (the higher the rank, the better the $\boldsymbol{\theta}$ is for maximization).
- Probability for drawing the j-th chromosome as a parent is proportional to r_j; i.e.,

$$\frac{r_j}{\sum_{j=1}^{P} r_j} = \frac{2r_j}{P(P+1)}.$$

Issue 2: How to produce offsprings from the parents

- Operation 1: Crossover
  - Two parents are selected: $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_c)$ and $\boldsymbol{\theta}' = (\theta'_1, \theta'_2, \ldots, \theta'_c)$.
  - Randomly select two gene locations and perform crossover; e.g., locations 3 and $c - 2$:

$$\begin{aligned} \text{Offspring } 1: &\quad (\theta_1, \theta_2, \theta'_3, \ldots, \theta'_{c-2}, \theta_{c-1}, \theta_c) \\ \text{Offspring } 2: &\quad (\theta'_1, \theta'_2, \theta_3, \ldots, \theta_{c-2}, \theta'_{c-1}, \theta'_c) \end{aligned}$$

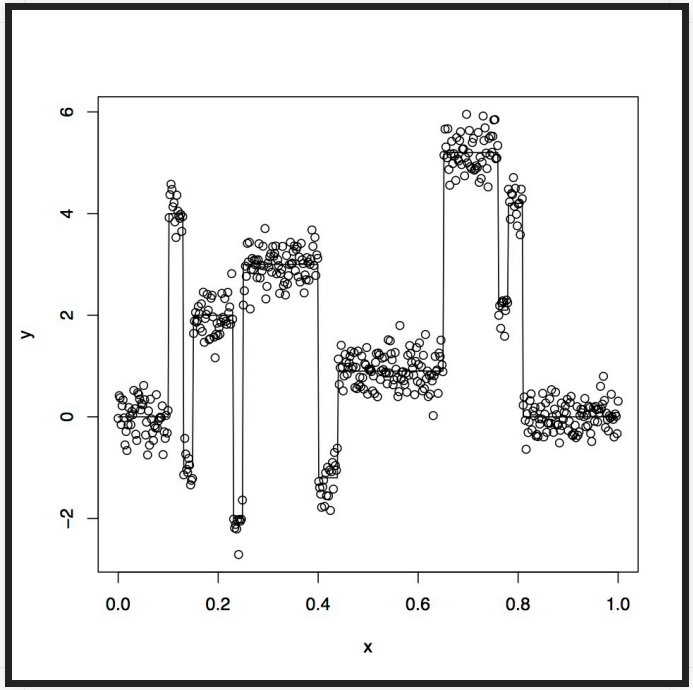
  - Alternatively, an offspring has equal probability of getting the gene from each of the parents.
  - Crossover allows the possibility of combining the best parts from different parents. It is a unique feature of genetic algorithms that makes them different from other optimization methods.

Issue 2: How to produce offsprings from the parents

- Operation 2: Mutation
  - Only one parent is required.
  - Each gene has a small probability (mutation rate) to change to a different gene value.
  - Mutation is used to prevent being trapped in local maxima.

Example

- Suppose n pairs of noisy measurements (x_i, y_i) are observed, with
$$y_i = f(x_i) + e_i, \quad x_1 < \cdots < x_n, \quad e_i \sim iid\ \mathcal{N}(0, \sigma^2), \quad i = 1, \ldots, n$$
- Assume that f is a piecewise constant function, but other details, such as the number of pieces, are unknown.
- The aim is to estimate f. (breakpoint detection)


y
6
4
2
0
−2
0.0
0.2
0.4
0.6
0.8
1.0
x

Example

- Let the (unknown) number of pieces be B, and the different pieces are joined at breakpoints $b_1, \ldots, b_{B-1}$.
- Without loss of generality, let $b_0 = 0 = x_1$ and $b_B = x_n + \delta = 1$ for a small $\delta > 0$ and assume $b_0 < \cdots < b_B$.
- The regression model for f is

$$f(x) = f_1 I_{\{b_0 \leq x < b_1\}} + \cdots + f_B I_{\{b_{B-1} \leq x < b_B\}}$$

  - I_E is the indicator function for the event E
  - f_j is the function value ("height") of the j-th piece of $f(x)$

Example

- We need to estimate $\boldsymbol{\theta}$:
  - the number of pieces B
  - the breakpoints $b_1, \ldots, b_{B-1}$
  - the heights $f_1, \ldots, f_B$
  - (and error variance σ^2)

Example

- Can we use MLE?
- One way of estimating $\boldsymbol{\theta}$ is to reconsider the problem as a model selection.
- Among the three types of parameters, which is the easiest one to estimate (given others)?
- Given B and $b_1, \ldots, b_{B-1}$, we can estimate the heights by

$$\hat{f}_j = \frac{1}{n} \sum_{\{i: b_{j-1} \le x_i < b_j\}} y_i$$

- Treat the selection of B and $b_1, \ldots, b_{B-1}$ as model selection via optimizing a model selection criterion
- We now show how to use genetic algorithm for such optimization.

Example

- Chromosome representation
- represent a candidate, i.e. $B, b_1, \ldots, b_{B-1}$
- constrain $b_1, \ldots, b_{B-1} \in \{x_1, \ldots, x_n\}$ (why?)
- an example of chromosome representation is
  - use "1" to denote a breakpoint gene and "0" to denote a normal gene
  - "00001000000100000000" is the chromosome when $n = 20, B = 3, b_1 = 5, b_2 = 12$
- Are crossover and mutation steps sensible?

Remarks

- One additional step (elitist step): always keep the best chromosome from the current generation to the next generation.
- Initialization: usually done in a random fashion.
- Need to choose some parameters:
  - $P \in [200, 300]$
  - $P(\text{crossover}) \approx 0.95$
  - Mutation rate ≈ 0.05.

Remarks

- Convergence:
  - Maximum number of iterations exceeds a threshold
  - The best chromosome does not change for, say, 20 generations.
- Island model
  - Several populations run in parallel and migration occurs after, say, every 5 generations.
  - We only migrate the best chromosome from each population.

A few examples

- Multiple linear regression
  - Suppose we have $\boldsymbol{\theta} = (0101100101)$, where 1 means that X_j is in the model and 0 means that X_j is not in the model.
- Regression mixture
  - In regression mixture, we have data points generated from two linear models, but we don't know which model each point is from.
  - Our solution may look like $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$, where $\theta_i \in \{1, 2, 3\}$ where 1 indicates that the point is from population 1, 2 indicates that the point is from population 2, and 3 indicates that the point is an outlier.

A few examples

- Time series
  - If we know that a time series is composed of segments described by different AR processes, we need to figure out where the breakpoints are, and the order of the AR process.
  - We can express a solution as, e.g., $\boldsymbol{\theta} = (1003000040000)$ where a positive integer indicates the order of the AR model, and the following zeros represent the number of points described by that AR model (minus one).