Neuro-Dynamic Programming for Fractionated Radiotherapy Planning

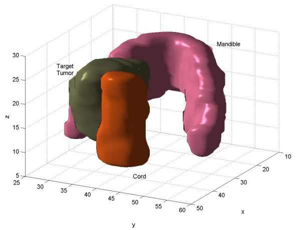
Geng Deng   Michael C. Ferris

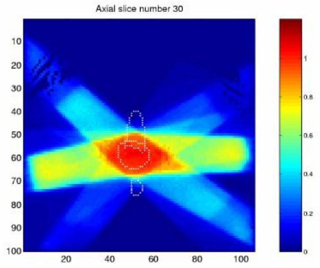
University of Wisconsin at Madison

Conference on Optimization and Health Care, Feb, 2006

Background

- Optimal delivery plan





- Deliver ideal dose on the target while avoid the critical organs and normal tissues.

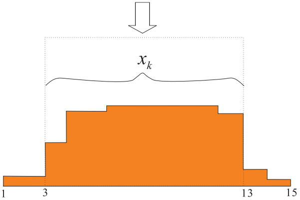
Fractionated radiotherapy (Dynamic problem)

- Treatments usually last several weeks
  - Limits burning
  - Allows healthy tissue to recover
- Types of day-to-day error: Registration error, internal organ motion, tumor shrinkage, and non-rigid transformation.
- Current approach: constant policy.
- New option: True dose delivered can be measured during individual treatments.
  - Update treatment plan day-to-day (online policy)
  - Compensate for errors

Problem overview

- State and state transition:

$$x_{k+1}(i) = x_k(i) + u_k(i + \omega_k), \forall i \in \mathcal{T}. \quad (1)$$



- Consider simple shifts in each direction
- Known error distributions
- Accumulation of errors
- Determine dose (u_k) to apply to minimize final error

Dynamic programming formulation

Minimize the cost-to-go function starting at x_0:

$$
\begin{aligned}
J_0(x_0) = \min \quad & \mathbb{E}\left[\sum_{k=0}^{N-1} g(x_k, x_{k+1}, u_k) + J_N(x_N)\right] \\
s.t. \quad & x_{k+1}(i) = x_k(i) + u_k(i+\omega_k), \\
& u_k \in U(x_k), k = 0, 1, \cdots, N-1.
\end{aligned} \tag{2}
$$

$J_N(x_N)$ is final cost function:

$$J_N(X_N) = \sum_{i \in \mathcal{T}} c(i)|x_N(i) - T(i)|$$

$g(x_k, x_{k+1}, u_k)$ is the immediate cost delivered outside the target:

$$g(x_k, x_{k+1}, u_k) = \sum_{i+\omega_k \notin \mathcal{T}} c(i+\omega_k) u_k(i+\omega_k)$$

An iterative formulation

The cost-to-go function at stage k can be formulated as:

$$J_k(x_k) = \min_{u_k \in U(x_k)} \mathbb{E}\left[g(x_k, x_{k+1}, u_k) + J_{k+1}(x_{k+1})\right]$$

Bellman's equation!

This is a finite horizon dynamic programming problem.

Existing policies

We will compare the following policies:

- Constant policy

$$u_k = T/N$$

- Reactive policy (Online policy)

$$u_k = \max(0, T - x_k)/(N - k)$$

- Modified reactive policy (Online policy)

$$u_k = a \cdot \max(0, T - x_k)/(N - k)$$

Why do we use NDP?

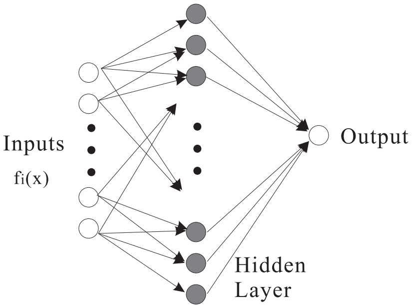
- Bellman's equation

$$\begin{aligned} u_k(x_k) &= \arg\min_{u_k \in U(x_k)} \mathbb{E}[g(x_k, x_{k+1}, u_k) + J_{k+1}(x_{k+1})] \\ s.t. \quad x_{k+1}(i) &= x_k(i) + u_k(i + \omega_k) \end{aligned} \tag{3}$$

- Dynamic programming method has difficulty to handle more than 4 stages, because of dimensionality.
- NDP approximates cost-to-go function $J_k(x_k)$ with a simple-structure function $\tilde{J}_k(x_k, r_k)$.
- NDP solves the problem fast.
- NDP obtains sub-optimal solutions.

Approximation architectures for $\tilde{J}(x, r)$

- Neural network (Input information are based on feature extraction $f_i(x)$)



- Heuristic mapping: $\tilde{J}(x, r) = r_0 + \sum_{i=1}^{I} r_i H_{u_i}(x)$. $H_{u_i}(x)$ is the heuristic cost-to-go applying policy u_i.

Approximate policy iteration

- Estimate parameters r_k.
- $x_k, \tilde{J}(\cdot, r_k) \xrightarrow{\text{Bellman's equation}} \hat{u}_k \xrightarrow{\text{Generate sample trajectories}}$ $\{x_{0i}, x_{1i}, \cdots, x_{Ni}\}, i = 1, \cdots, M \xrightarrow{\text{Evaluate costs}} c(x_{ki})$
- Solve least squares problem in r_k

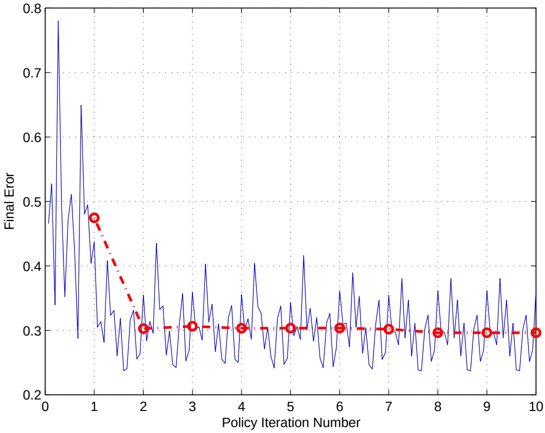
$$\min_{r_k} \sum_{i=1}^{M} \left| \tilde{J}_k(x_{ki}, r_k) - c(x_{ki}) \right|^2$$

- Simulation and evaluation steps alternate

Computational experiments

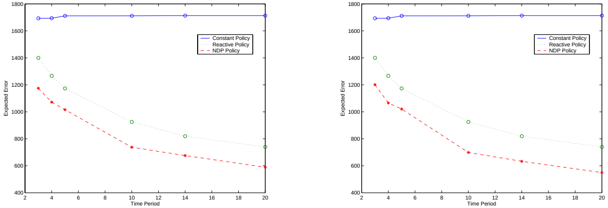
- Test a simple one dimensional case and a real problem: head and neck
- Use 5 candidate policies at each stage
- Test in high and low volatility scenarios
- Use two approximation architectures:
  - Neural network: features $(f_i(x_k))$ used are average dose, standard deviation of dose, and curvature of dose distribution
  - Heuristic mapping: Heuristic policies used are constant policy, reactive policy and modified reactive policy with $a = 2$.

Performance of approximate policy iteration



Comparison results in the head and neck problem

The figures show results for different policies in the high volatility case:



Neural network architecture (left) and HEuristic mapping architecture (right)

- NDP $>$ Reactive $>$ Constant
- Results of NN and HE are comparably the same, but HE takes much longer computation time
- Online policies require more computational effort

Conclusions

- Online policies with extra information outperform offline policies
- DP method is inapplicable in practice. NDP reduces computation time and produces "approximately" optimal policies
- Implemented on real patient data
- Future work:
  - Explore more policies
  - Consider different types of error
  - Fast computation