AI = Function Approximation
AI = Probability
AI = Optimization

AI = Optimization: Issues

- Optimizing a function vs. optimization over time: static/one-shot vs. dynamic
- Discrete vs. continuous: Is there smoothness?
- Is data expensive? Is computation expensive?
- Do I have a model? Can I build a model?
- Are there safety issues? Very expensive function evaluations?
- Feasible vs. optimal
- Globally optimal?
- What should be optimized, anyway?

Policies vs. Planning

Static/one-shot/temporally-independent/noisy-function optimization

Achieving a goal: Root finding

- World: $y = f(x,u) + \text{noise}$
- Choose u so that $E(y) = y_d$
- This is basically trying to hit a target. Archery example.
- Continuous x, u; continuous f(x,u)
- Numerical Recipes in X is excellent resource for noiseless case:
- Noiseless 1D, can bracket root and use bisection algorithm
- See Numerical Recipes in X for more sophisticated noiseless 1D search algorithms

Achieving a goal: Root finding (2)

- Approach 1: Learn model of inverse of f:

$u = f_m^{-1}(x,y_d)$

- May have problems if inverse does not exist: no u works, or more than one u works
- Approach 2: Search using forward model:

$y_p = f_m(x,u)$

- Grid search, random search, gradient descent, second order method, minimization

Search using forward model

- $y = f(x,u)$, $A = \partial f/\partial x$, $B = \partial f/\partial u$
- Gradient descent: $\Delta u = B^T(y_d - y)$

Safe, local minima, slow/many function evals

- Second order method: $\Delta u = B^{-1}(y_d - y)$

(actually solve $B\Delta u = (y_d - y)$ for Δu)

Fast, local minima, may fail badly, see Levenberg-Marquardt or trust region methods for sophisticated versions

- Minimization: minimize

$C = (y_d - y)^T Q(y_d - y) + u^T R u$

Explicit tradeoff of accuracy and "effort". See next section

- Examples where each method will work well, and examples where they will fail or work badly

Optimizing a function

- Maximize reward r(x,u) or minimize cost c(x,u) with respect to u. Let's minimize c()
- Discrete u or discontinuous c(): See discrete optimization section.
- Continuous u and c(x,u)
- Discuss linear programming later
- Noiseless case: see Numerical Recipes in X

Noiseless case, no derivatives, Random approaches

- Simulated annealing: general, good first try
- Genetic Algorithms: need to think about representation

Noiseless case, no derivatives, Deterministic approaches

- Simplex, Amoeba: general, good first try
- Powell's method

In theory, …

- $B = \partial c/\partial u$
- Gradient descent: $\Delta u = -\varepsilon B$
- Picture of ellipse with failure of gradient descent
- $H = \partial^2 c/\partial^2 u$
- 2nd order method: $\Delta u = -H^{-1}B$, actually solve $H\Delta u = -B$
- Sophisticated methods use 1D line minimization and maintain useful set of directions, aligned with problem

Noiseless case, First derivatives

- Conjugate gradient: Good for many parameters
- Quasi-Newton

Noiseless case, 2nd order
These are good when close to local optimum

- Levenberg Marquardt
- Trust Region

Noisy Case

- Stochastic gradient descent
- Simplex, Amoeba w/ multiple measurements
- Response surface methods: experiment design, go for best guess
- PMAX, IEMAX: active learning: go for best combination of performance and uncertainty reduction
- Q2: automate as much as possible
- More info on auton lab web page.