Problem Definitions

- Do we know what are good robot actions?
  - YES: Supervised learning
    Drive around with the training “on” to generate input/output pairs
  - NO: Unsupervised learning (critic based learning)
    Explore parameters till you find “right behavior”

- Online/Offline?
  - Offline: train/search for a complete solution before implementing
  - Online: take action, evaluate, take next action etc.
Example: Robot Control

- Robots observe environment through some sensors
- Sensors are inputed into a neural network
- Output of neural network determines direction/velocity of rover

Robot Control

Sensor Inputs
Desired Heading

Autonomous Robot

Heading
Robot Control

Sensor Inputs
Desired Heading

→ Neuro Controller

Steering

→ Autonomous Robot

Headings

Neural Network Training Signal

Sensor Inputs
Desired Heading

→ Neuro Controller

Steering

→ Autonomous Robot

Headings
Robot Control: Learn with a Teacher

Robot Control: Learn without a Teacher

Sensor Inputs
Desired Heading

Steering Error

Steering

Autonomous Robot

Heading

Neuro Controller

Robot Performance

Desired Heading
Neural Networks for Nonlinear Control

- Motivation:
  - Control a system with nonlinear dynamics
    - Robot
    - Satellite
    - Air vehicle

- Do we know what the good control strategies are?
  - Yes: “teach” neural network those strategies
    - Drive a car and record good driver actions for each state
    - Fly a helicopter and record good pilot actions for each state
  - No: have a neural network discover those strategies
    - Let car drive around and provide feedback on performance

Neuro-Control with a teacher

1. Initialize a neural network
2. Let neural network pick heading
3. Compute heading error using teacher
4. Use error to update neural network weights
   \( \Delta w = ... \)
5. Go to step 2
Unsupervised Learning

- What if we don’t have a teacher?
- Unsupervised learning: learning without a set of labeled examples
- Each input results in an outcome (measured by a reward)
- Training:

Neuro-Control without a teacher

1. At t=0 initialize N neural networks
2. Pick a network using $\epsilon$-greedy alg ($\epsilon=.1$)
3. Randomly modify network parameters
4. Use network on this agent for T steps
5. Evaluate network performance
6. Re-insert network into pool
7. Remove worst network from pool
8. Go to step 2
Example: Quadrotor Control

• Benefits of quadrotors:
  − Operate in dangerous & challenging environments
  − Overcome resource limitations
  − Maneuverability - over airplanes
  − Mechanical simplicity - over helicopters

• Drawbacks:
  − Highly non-linear dynamics
  − Unintuitive control (difficult for a human)
  − Stability problems – as opposed to airplanes
  − Control problems – as opposed to helicopters

Background – Quadrotor Control
Background — Quadrotor Control
Background — Quadrotor Control
Background — Quadrotor Control
Background — Quadrotor Control
Background — Quadrotor Control
Controller Formulation

- Break problem down into solvable units
  - Top Level: Position controller - for waypoint navigation
  - Middle Level: Attitude controller - for maintaining stability
  - Low level: Pitch/Yaw controller – for specific rotor movement

- All following results from:

Position Controller Formulation

- \( -X_d - X \)
- \( -\dot{X} \)
- \( -Y_d - Y \)
- \( -\dot{Y} \)
- \( -Z_d - Z \)
- \( -\dot{Z} \)

Position Controller

\[ \phi_d \]
\[ \theta_d \]
\[ \dot{Z}_d \]

Position Controller Training

- Fitness based on distance traveled and distance to goal:
  \[ F = 10000 - d_g + (100 - d_t)^2 \]

- Stability criteria
  - Erratic motion resulted in a fitness set to 0

- 15000 iterations to train:
  - Select - Mutate - Evaluate
  - Evaluate position controller multiple grid points
  - Run with neuro-controller for 10 seconds
Attitude Controller Formulation

\[ \phi_d - \phi \rightarrow \Omega_1 \]
\[ \theta_d - \theta \rightarrow \Omega_2 \]
\[ \hat{Z}_d - \hat{Z} \rightarrow \Omega_3 \]
\[ \psi \rightarrow \Omega_4 \]

Attitude Controller

Attitude Controller Formulation

\[ \phi_d - \phi \rightarrow E_\phi \rightarrow \Omega_1 \]
\[ \theta_d - \theta \rightarrow E_\theta \rightarrow \Omega_2 \]
\[ \hat{Z}_d - \hat{Z} \rightarrow E_v \rightarrow \Omega_3 \]
\[ \psi \rightarrow E_\psi \rightarrow \Omega_4 \]

Attitude Controller

Roll

Pitch

Vertical

Yaw
Attitude Controller Formulation

- **Step 1:** Supervised
  - Population of 100 controllers
  - Control quadrotor for 10s
  - Fitness based on matching PID results
    \[ F = e^{-\sum |A_{nn} - A_{pid}|} \]
  - Run for 3000 iterations

- **Step 2:** Unsupervised
  - Create new population from best controllers from step 1
  - Add 20% weight mutation
  - Fitness based directly on achieved angle
  - Run for another 1000 iterations
Experiments

1. **Navigation**
   - Set of waypoints given, controller performance measured

2. **Robustness to disturbances**
   - Modeled as a discrete change in the craft's current orientation
     - Sudden at 30° pitch
     - Time to recover, and range of ability to recover compared

3. **Robustness to noise**
   - Random sensor and actuator noise added.
   - Average distance craft traveled around the desired hold point compared

4. **Robustness to design parameters**
   - Craft physical parameters (size, weight, thrust & drag coefficients)
   - Time to perform move compared (mostly binary result).

---

**Navigation Results**

[Graphs showing navigation results comparing Neuro Controller and PID Controller]
2. Robustness to Disturbances

3. Robustness to Sensor/Actuator Noise
4. Robustness to Design Parameters

![Graphs showing time to complete vs. % of design mass and thrust coefficient with comparison of Neuro Controller and PID Controller.]

Conclusions — Controller Development/Performance

- Developed a hierarchy of adaptive controllers
  - Divided problem
  - Applied relevant model information

- Produced Quadrotor control that Overcame Disturbances
  - Up to 180° flip

- Handled sensor and actuator Noise
  - 4 times better than PID for sensor noise
  - 8 times better than PID for actuator noise

- Provided insensitivity to Design Parameter
  - Mass & Thrust changes of ±30%
Quadrotor Control: Demo

Quadrotor Recovery: Demo
Learning in the Real World

Human in the Loop: Suggestion Agents