Lecture 11 – Learning with Distal Teachers
(Forward and Inverse Models)

Contents:
• The Distal Teacher Problem
• Distal Supervised Learning
• Direct Inverse Modeling
• Forward Models
• Jacobian
• Combined Inverse & Forward Models
Trajectory Planning, Inv. Kinematics & Inv. Dynamics

\( x_s = (x, y, z) \)

\( x_e = (x, y, z) \)

\( \text{start} (x_s), \text{end} (x_e) \)

Trajectory Planner

\( \{x_s(0), \ldots, x_e(t)\} \)

Inverse Kinematics

\( \{\theta_s(0), \ldots, \theta_e(t)\} \)

Inverse Dynamics

\( \{\tau_s(0), \ldots, \tau_e(t)\} \)

\( \theta = f(x) \)

\( \tau = f(\theta) \)
The Distal Teacher Problem

For certain problems, i.e., learning to act, teaching signals are not easily obtained:

- Example 1: Distal Supervised Learning

- Example 2: Reinforcement Learning

How to learn to choose the correct action without getting supervised information about the quality of an action?
Distal Supervised Learning

- **Given:**
  - target \( t \) in distal space and the current state of the system \( x \)
  - an unknown system \( y = f(x, u) \)

- **Goal:**
  - find the action \( u \) to achieve \( t \)

Example:

\[ t \rightarrow \text{Learner} \rightarrow x \rightarrow \text{Environment} \rightarrow y \rightarrow \text{Learner} \]

\( x \) and \( y \) are not the same: \( y \) is a certain observable as the outcome of action \( u \) in state \( x \)
Direct Inverse Models

- **Static Models:**
  - Forward Model: \( y = f(x) \)
  - Inverse Model: \( x = f^{-1}(y) \)

- **Dynamic Models:**
  - Forward Model: \( y = f(x,u) \)
  - Inverse Model: \( u = f^{-1}(y,x) \)

**Example:**

![Diagram showing training and testing processes for modeling.]
Problems of Direct Inverse Learning: Training Data

How to generate useful training data?

- exhaustive data generation
  - only possible for low dimensional problems (the curse ...)
  - for real physical systems, it is not easy to drive the system to generate exhaustive data
- use incompletely trained model to generate “explorative” data
  - predict $u$ for targets $t$
  - then apply those predicted $u$, observe $y$, update the learning system
  - but there is a big danger of getting stuck in irrelevant training data!
Problems of Direct Inverse Learning: Non-Uniqueness

What if the Inverse Function is not unique?

- Example 1: \( y = au_1 + bu_2 + cx \)
- Example 2: how to determine \( u \) from just two equations, but 3 unknowns?

**Important:** For non-unique inverse models, the learning system has to pick ONE out of many solutions, but it also has to ensure that this solution is REALLY a valid solution!
Can we learn Non-Unique mappings?

Analogy: Learning networks are like lookup-tables
- they average over all outputs that belong to the same input in order to eliminate noise!
Convex and Non-convex Mappings

**When can we learn a non-unique inverse model?**

- Only when the mapping is convex in output space!
Dealing with Non-Convex Inverse Models

Only Use Training Data from a Convex Region
- this is usually very hard

Change of Representation in Conjunction with Spatially Localized Learning
- this is REALLY cool if applicable!

\[
x = f(\theta) \implies \dot{x} = J(\theta) \dot{\theta}
\]
\[
\dot{x} = J(\theta) \dot{\theta}_1; \quad \dot{x} = J(\theta) \dot{\theta}_2 \implies \dot{x} = J(\theta)(\dot{\theta}_1 + \dot{\theta}_2)
\]
\[
\implies (\dot{x}, \theta) \rightarrow (\dot{\theta}) \text{ is a convex mapping}
\]
Dealing with Non-Convex Inverse Models

Learning the Joint Density

- There are various Bayesian and nonparametric methods to do this
Dealing with Non-Convex Inverse Models

**Search in Forward Model**

- Learn (well-defined) forward model
- Search for appropriate action by gradient descent (a more AI-ish technique)
  - this will generate a new action that should give interesting new training data that helps to achieve the target

\[ u^{n+1} = u^n - \alpha \frac{\partial J}{\partial u} \]

where \( J = (t - y)^T(t - y) \)

\[ y = f(x, u) \]

Creating appropriate training data by learning the easier (well-conditioned) forward model
The Jacobian

Taking derivatives through learning methods

- From least squares learning, we already know the derivative:
  \[
  \frac{\partial J}{\partial \mathbf{w}} = - \frac{\partial y}{\partial \mathbf{w}}^T (\mathbf{t} - \mathbf{y}) \quad \text{where} \quad J = (\mathbf{t} - \mathbf{y})^T (\mathbf{t} - \mathbf{y})
  \]

- But we can also calculate the derivative with respect to any other quantity of the learner, e.g., with respect to the input
  \[
  \frac{\partial J}{\partial \mathbf{x}} = - \frac{\partial y}{\partial \mathbf{x}}^T (\mathbf{t} - \mathbf{y})
  \]

The derivative of the outputs with respect to the inputs is called the **Jacobian Matrix**. Note that the coefficients of the matrix change nonlinearly as function of the $\mathbf{x}$ and the $\mathbf{y}$ data. The derivation of this derivative is just like backprop or gradient descent in RBFs. (Bishop, Ch.4)
Inverse-Forward Model Combination

Instead of using search techniques, learning machines can learn the entire distal teacher problem directly

\[
\frac{\partial J}{\partial \mathbf{w}_{\text{inv}}} = - \frac{\partial \mathbf{y}}{\partial \mathbf{w}_{\text{inv}}}^T (\mathbf{t} - \mathbf{y}) = - \frac{\partial \mathbf{u}}{\partial \mathbf{w}_{\text{inv}}}^T \frac{\partial \mathbf{y}}{\partial \mathbf{u}}^T (\mathbf{t} - \mathbf{y})
\]

\[
y = f(x, u)
\]

\[
u = f^{-1}(y, x)
\]

Jacobian of forward model

Gradient descent in Inverse Model
Inverse-Forward Model Combination

What is the correct error signal to use for learning:

- **the forward model**
  \[ (t - y) \text{ or } (t - \hat{y}) \text{ or } (y - \hat{y}) \]

- **the inverse model**
  \[ (t - y) \text{ or } (t - \hat{y}) \text{ or } (y - \hat{y}) \]

- How accurate does the Forward Model have to be to learn an accurate Inverse Model?

- Which solution to the inverse problem does the Learning box acquire?

We have as error criteria:
the performance error, the predicted performance error or the prediction error

**Learning the forward model**: use the prediction error, this is clear!
**Learning the Inverse model**: the performance error or the predicted performance error can be used.

Using the real performance error is advantageous since it will drive the inverse model to become more accurate, even in the case of an inaccurate forward model. Otherwise we would learn the inverse of the inaccurate forward model, but this is not useful for fulfilling the real task.
Inverse-Forward Model Combination

Additional Optimization Criteria to Select a Particular Inverse Solution

\[ e.g.: J = \frac{1}{2} (t - y)^T (t - y) + \gamma \frac{1}{2} u^T u \]

\[ \frac{\partial J}{\partial w_{inv}} = - \frac{\partial y}{\partial w_{inv}}^T (t - y) + \gamma \frac{\partial u}{\partial w_{inv}}^T u \]

\[ = - \frac{\partial u}{\partial w_{inv}}^T \frac{\partial y}{\partial u}^T (t - y) + \gamma \frac{\partial u}{\partial w_{inv}}^T u \]

\[ = - \frac{\partial u}{\partial w_{inv}}^T \left( \frac{\partial y}{\partial u}^T (t - y) - \gamma u \right) \]

\[ y = f(x,u) \]

\[ u = f^{-1}(y,x) \]
Summary

- Certain problems cannot just be learned—they are theoretically and practically impossible to learn (e.g., non-convex inverses)
- Clever representations and machine learning architectures allow to overcome such problems (after the essence of the problem has been understood)
- Distal supervised learning is a powerful method to approach inverse problems and simple reinforcement learning problems