Decision Making in Robots and Autonomous Agents

Learning Decision Models from Data

Subramanian Ramamoorthy
School of Informatics

10 March, 2015
Where Do Our Models Come From?

... when we do not write them down explicitly?

- Consider the following, predicting/understanding behaviour:
  - Activity modelling (to find anomalies or regularities)
  - Plan and intention recognition
  - Teaching/instructing

- Learning of models is defined with respect to the underlying principle for decision making
  - E.g., Bellman optimality for RL & inverse RL by learning rewards
Making *Structured* Predictions

Learn a function mapping inputs to complex outputs:

\[ f : X \rightarrow Y \]

Input Space → Decoding → Output Space
What did We Mean by Structure?

• Correlations among outputs
  – Determiners often precede nouns
  – Sentences usually have verbs

• Global coherence
  – It doesn’t make sense to have three determiners next to each other

• Objective or loss functions enforce this:
  – Translations should have good sequences of words
  – Summaries should be coherent
A Robotics Example – Legged Motion
What is *Structure* Here?

- Physical characteristics of the task:
  - Need to stay balanced on foot, e.g., do not topple over
  - Dynamics of how the body responds to motor forces
  - Limited to only move legs as per kinematic structure

- Is there a compact encoding of all trajectories?
  - Manifold that can be learnt from data

- If I give you a trajectory, could you check if valid? (anomaly detection?)

- If I ask for a new or partial trajectory, could you generate it? (prediction)
Encoding Dynamical Tasks

Problem:
How to instruct a humanoid robot to safely navigate irregular terrain?
  – Task: Step on a sequence of footholds
  – Constraints: Don’t slip, don’t lose balance, bounded torques, etc
Motion in Phase Spaces: Caricature 1

Bug goes from A to B
- Picks up some velocity and slows down at goal
Motion in Phase Spaces: Caricature 2

Why does the bug move?
Dynamics – different “laws of motion”

\[ \dot{x} = f_1(x) \]
\[ \dot{x} = f_2(x) \]
Motion in Phase Spaces: Caricature 3

Constrained, high-dimensional nonlinear dynamics

\[ \dot{x} > g(x, \dot{x}) \]
Example: Pendulum Phase Space

- Phase space is organized into families (open sets) of trajectories
- Trajectories may be parameterized by a single variable: energy
Global Planning via *Natural Dynamics*

After large impact!

Increasing Energy

Suspended pendulum

Inverted pendulum

Stabilize at hyperbolic fixed point

$E = E^*$ (Stay on *Separatrix*)

$\dot{s} \geq 0$

$\dot{s} \leq 0$

$E < E^*$ (Libration - Pump)

Sliding Mode

$E > E^*$ (Rotation - Spin Down)
Control of Cart-Pole System [R. + Kuipers ‘03]
Template Model: Compass Gait Walking

(Kuo, Science, 2005)
Generating Trajectories [R. + Kuipers ‘08]

Low-dimensional Plan → Nonparametric approximation of Dynamics → High-dimensional Trajectory

Known Analytically

Observations: <State, Action, State>

Dynamically Equivalent

1. Random actions
2. Imperfect gait
3. Active learning
Approaching Unknown Manifold from Data

Organize data in a k-NN graph

Where is manifold in the graph?

- Manifold $\Leftrightarrow$ Set of geodesic trajectories restricted to it
- If the manifold encodes task – every geodesic must behave like template plan
- Diagram must commute!
  - Minimize commutativity error

$S_H \xrightarrow{\mathcal{M}_H} S_H$

$\downarrow \pi \quad \downarrow \pi$

$S_L \xrightarrow{\mathcal{M}_L} S_L$

$e_{comm} = f(\pi(x_j) - y_j)$, for sequences $\{x_j\}, \{y_j\}, x_j \in S_H, y_j \in S_L$

Find sequence $\{x_j\}$, given template plan sequence $\{y_j\}$:

$\{x_j\} = \text{argmin} \sum_{j=1}^{N} f(\pi(x_j) - y_j), x_j \in S_H, y_j \in S_L$
Manifold Learning – Structured Model

• Learn a mapping from a point on the manifold to its tangent basis $H(x)$,

$$H : x \in \mathbb{R}^D \mapsto \left[ \frac{\partial}{\partial y_1} M(y) \cdots \frac{\partial}{\partial y_d} M(y) \right] \in \mathbb{R}^{D \times d}$$

$$H_\theta(x^{ij}) \epsilon^{ij} \approx \Delta^{i,j},$$

$$\text{err}(\theta) = \min_{\{\epsilon^{ij}\}} \sum_{i,j \in N_i} \left\| H_\theta(x^{ij}) \epsilon^{ij} - \Delta^{i,j} \right\|_2^2,$$

• Metric (via k-NN graph) accounts for:
  • Task space distance
  • Temporal order

[Dollar et al 2006].
The grey mesh is the Delaunay triangulation of the 100 data points - shown for comparison against the desired manifold (from which curves in fig. c are drawn)

Check: What is the “decision” being made here?
Constrained Trajectory Generation on Skill Manifolds [Havoutis + R.]
Constrained Walking: Variable Foot Placement

Following the unconstrained geodesics, oblivious to obstacles

Constrained geodesic trajectory – avoid obstacles, while staying within manifold
Markov Decision Process (MDP)

- Main elements of problem specification are the transition and reward probabilities
- Through Bellman optimality, we get at policy: $\pi(s)$ or $\pi(s,a)$.
- This notion of optimality imposes structure – recursive specification of what could possibly be optimal!
MDPs and Reinforcement Learning

- $S$: Finite set of $N$ states
- $A = \{a_1, \ldots, a_k\}$: set of $k$ actions
- $P_{sa}(\cdot)$: state transition probabilities
- $R : S \rightarrow \mathbb{R}$: reward function, with maximum value $R_{max}$

\begin{align*}
V^\pi(s_1) &= E \left[ R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \cdots | \pi \right] \\
Q^\pi(s, a) &= R(s) + \gamma E_{s' \sim P_{sa}(\cdot)} [V^\pi(s')] \\
\pi(s) &\in \arg \max_{a \in A} Q^\pi(s, a)
\end{align*}
**Inverse Reinforcement Learning**  
[Russell + Ng, ICML 2000]

- **Given:**
  - Measurements of an agent’s behaviour over time, in varied circumstances
  - Possibly, measurements of the sensory inputs to that agent
  - Sometimes, a model of the environment

- **Find:** the reward function $R(s,a)$

- **Long history:**
  - Kalman 1968, how to get cost function of LQR?
  - Boyd 1994, solution using semidefinite programming
Many Motivations for Reward Learning

• Computational models for animal learning
  – “In examining animal and human behavior we must consider the reward function as an unknown to be ascertained through empirical investigation”

• Agent design
  – “An agent designed ... may only have a very rough idea of the reward function whose optimization would generate ‘desirable’ behavior.”
  – What does it mean to drive “well”?

• Multi-agent systems and Mechanism design
  – Learning opponents’ reward functions that guide their actions to devise strategies against them
Starting Point: Characterising Solution Set

The policy \( \pi(s) := a_1 \) is optimal if and only if \( \forall a, \)

\[
(P_{a_1} - P_a)(I - \gamma P_{a_1})^{-1} R \succeq 0
\]

To derive this, start by observing that:

\[
V^\pi = R + \gamma P_{a_1} V^\pi
\]

Therefore, \( V^\pi = (I - \gamma P_{a_1})^{-1} R \)
Characterising Solution Set

For the optimal action $a_1$, we have that,

$$a_1 := \pi(s) \in \text{arg max}_a \sum_{s'} P_{sa}(s')V^\pi(s')$$

We can rewrite this as,

$$\sum_{s'} P_{sa_1}(s')V^\pi(s') \geq \sum_{s'} P_{sa}(s')V^\pi(s'), \forall s \in S, a \in A$$

$$P_{a_1}V^\pi \geq P_aV^\pi, \forall a \in A \setminus a_1$$
Characterising Solution Set

Knowing the following,

\[ V^\pi = (I - \gamma P_{a1})^{-1} R \]

\[ P_{a1} V^\pi \geq P_a V^\pi, \forall a \in A \setminus a_1 \]

We get,

\[ P_{a1} (I - \gamma P_{a1})^{-1} R \geq P_a (I - \gamma P_{a1})^{-1} R, \forall a \in A \setminus a_1 \]

If all inequalities were strict, this is a necessary and sufficient condition for \( a_1 \) to be unique optimal policy.
Towards Learning Reward Function

• We have a set of constraints: \[(P_{a_1} - P_a)(I - \gamma P_{a_1})^{-1} R \succeq 0\]

For Inverse Reinforcement Learning, we ask that:
• R should make \(\pi\) optimal
• Any single step deviation from \(\pi\) should be as costly as possible

• We could choose a function \(R\) so as to maximise,

\[
\sum_{s \in S} \left( Q^\pi(s, a_1) - \max_{a \in A \setminus a_1} Q^\pi(s, a) \right)
\]
Linear Programming for Learning $R$

$$\sum_{s \in S} \left( Q^\pi(s, a_1) - \max_{a \in A \setminus a_1} Q^\pi(s, a) \right)$$

- We are trying to **maximise** sum of differences between,
  - Quality of optimal action
  - Quality of the next-best action

- One might also want to apply Ockham’s razor – prefer reward functions that have small values (this is a version of “simple”)
- One way to achieve this is to penalise the norm of reward using an $\ell_1$ penalty term.
LP Formulation of IRL

$$\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{N} \min_{a \in \{a_2, \ldots, a_k\}} \left\{ \left( P_{a_1}(i) - P_a(i) \right) \right. \\
& \left. \quad \left( I - \gamma P_{a_1} \right)^{-1} R \right\} - \lambda \| R \|_1 \\
\text{s.t.} & \quad (P_{a_1} - P_a) \left( I - \gamma P_{a_1} \right)^{-1} R \succeq 0 \\
& \quad \forall a \in A \setminus a_1 \\
& \quad |R_i| \leq R_{\text{max}}, \quad i = 1, \ldots, N
\end{align*}$$
Using Function Approximation

- Often, when learning in large state spaces, we want to see if the reward function can be made simpler – limit search
- One simple (albeit restrictive) assumption is that of linear functions – which does allow for function approximation

- Using a set of basis functions, write,

\[ R(s) = \alpha_1 \phi_1(s) + \alpha_2 \phi_2(s) + \cdots + \alpha_d \phi_d(s) \]

- Due to linearity of expectation, we will then have

\[ V^\pi = \alpha_1 V_1^\pi + \cdots + \alpha_d V_d^\pi. \]
Linear Function Approximation for IRL

• The condition for optimal action becomes,

\[ E_{s' \sim P_{s a_1}} [V^\pi(s')] \geq E_{s' \sim P_{sa}} [V^\pi(s')] \]

• So, the modified LP formulation is:

\[
\begin{align*}
\text{maximize} & \quad \sum_{s \in S_0} \min_{a \in \{a_2, \ldots, a_k\}} \left\{ p(E_{s' \sim P_{s a_1}} [V^\pi(s')] - E_{s' \sim P_{sa}} [V^\pi(s')]) \right\} \\
\text{s.t.} & \quad |\alpha_i| \leq 1, \quad i = 1, \ldots, d
\end{align*}
\]
Grid World
– Trajectories and Reward Function
IRL in Grid World (for $\lambda = 0, 1.05$)
What Might Data Look Like in Reality?

- Length
- Speed
- Road Type
- Lanes
- Accidents
- Construction
- Congestion
- Time of day

25 Taxi Drivers

[Source: Zebart et al., AAA108]
Activity Forecasting [Kitani et al., ECCV ‘12]
Acknowledgements

Many slides are adapted from the following sources:

• Tutorial on SPIRL at AAAI 2011 by Hal Daume III
• Russell and Ng’s original paper in ICML 2000