CSC2621 Topics in Robotics
Reinforcement Learning in Robotics

Week 2: Supervised & Imitation Learning

Instructor: Animesh Garg

TA: Dylan Turpin & Tingwu Wang

Agenda

- Invitation to Imitation
- DAGGER: Dataset Aggregation
- End-to-End learning for self-driving
- Behavioral Cloning from Observation

- Open-Problems and Project Ideas
- Logistics
- Presentation Sign-ups

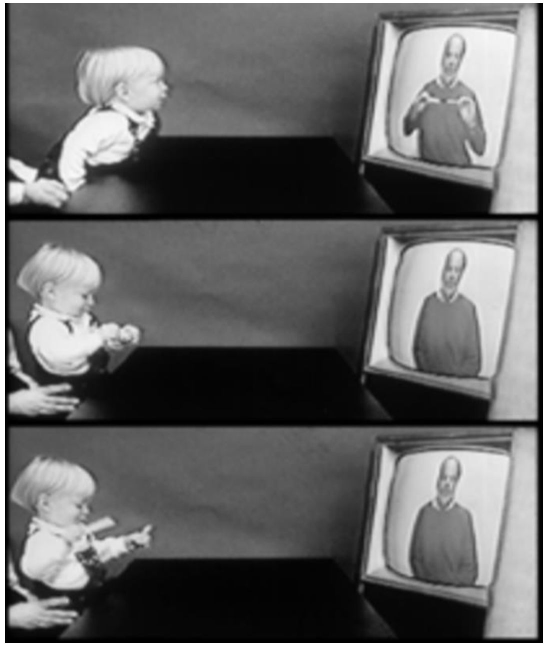
Invitation to Imitation

Drew Bagnell

Topic: Imitation Learning
Presenter: Animesh Garg

Why Imitation

How are people so good at learning quickly and generalizing?



Facial Gestures
Age: 19 hours to 20 days

Direct Imitation
Age: 18 months

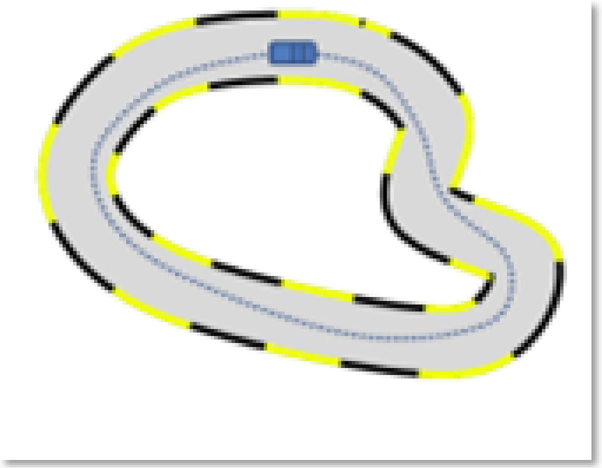
Assembly Tasks from TV
Age: 14-24 months

Meltzoff & Moore, Science 1977; Meltzoff & Moore, Dev Psych. 1989, Meltzoff 1988

Why Imitation

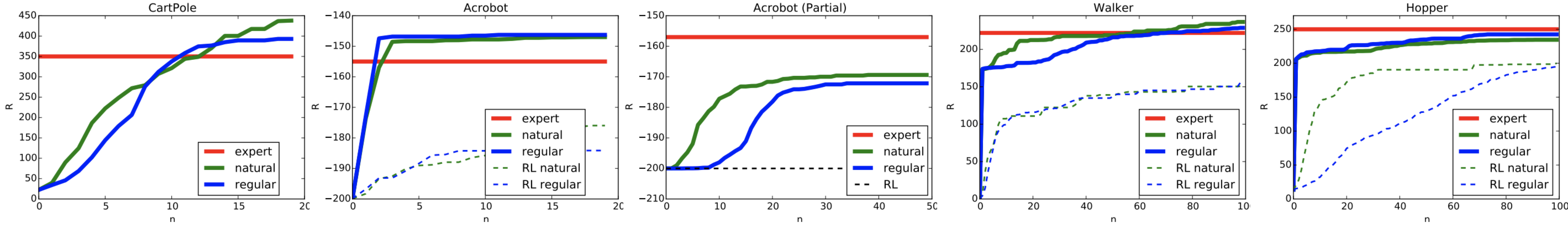
Consider Autonomous Driving:

- Input: Field of view
- Output: Steering Angle
- Manually programming this is difficult
- Having human expert demonstrate is easy



Learning from expert demonstrations = Imitation Learning!

Why Imitation? Why not RL?



(a) Cartpole (b) Acrobot (c) Acrobot (POMDP) (d) Walker (e) Hopper

Figure 2. Performance (cumulative reward R on y-axis) versus number of episodes (n on x-axis) of AggreVaTeD (blue and green), experts (red), and RL algorithms (dotted) on different robotics simulators.

Imitation learning is **exponentially lower sample complexity** than Reinforcement Learning for sequential predictions

RL: such as REINFORCE and Policy Gradient

Why Imitation? Is it just Supervised Learning?

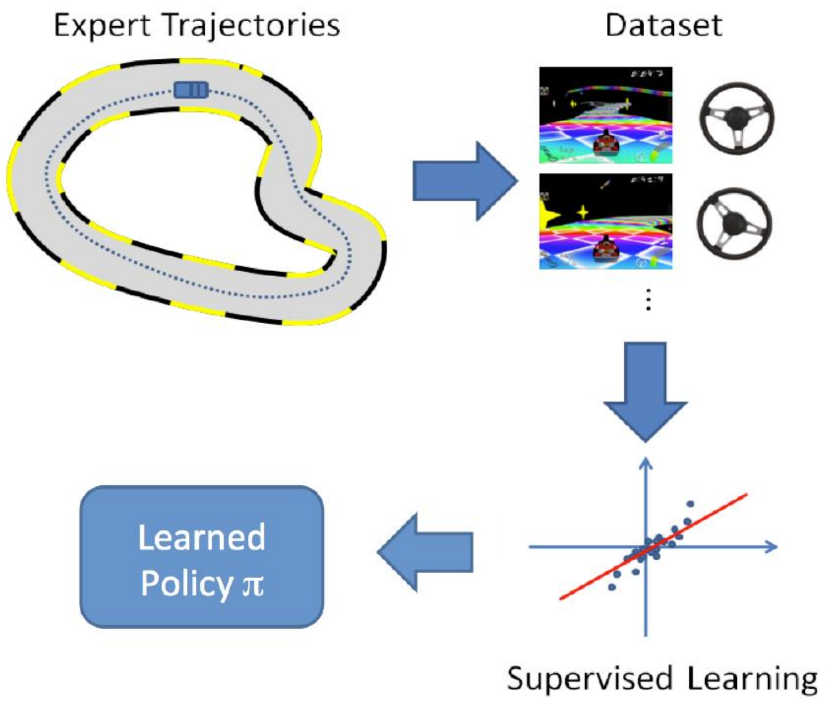
Supervised Learning:

- Prediction has no effect on world
  - Data is IID
- No sense of "future"

Imitation Learning:

- Predictions lead to actions that will change the world and affect future actions
  - Data is highly correlated
- Robotic Systems have sophisticated planning algorithms for reasoning into the future

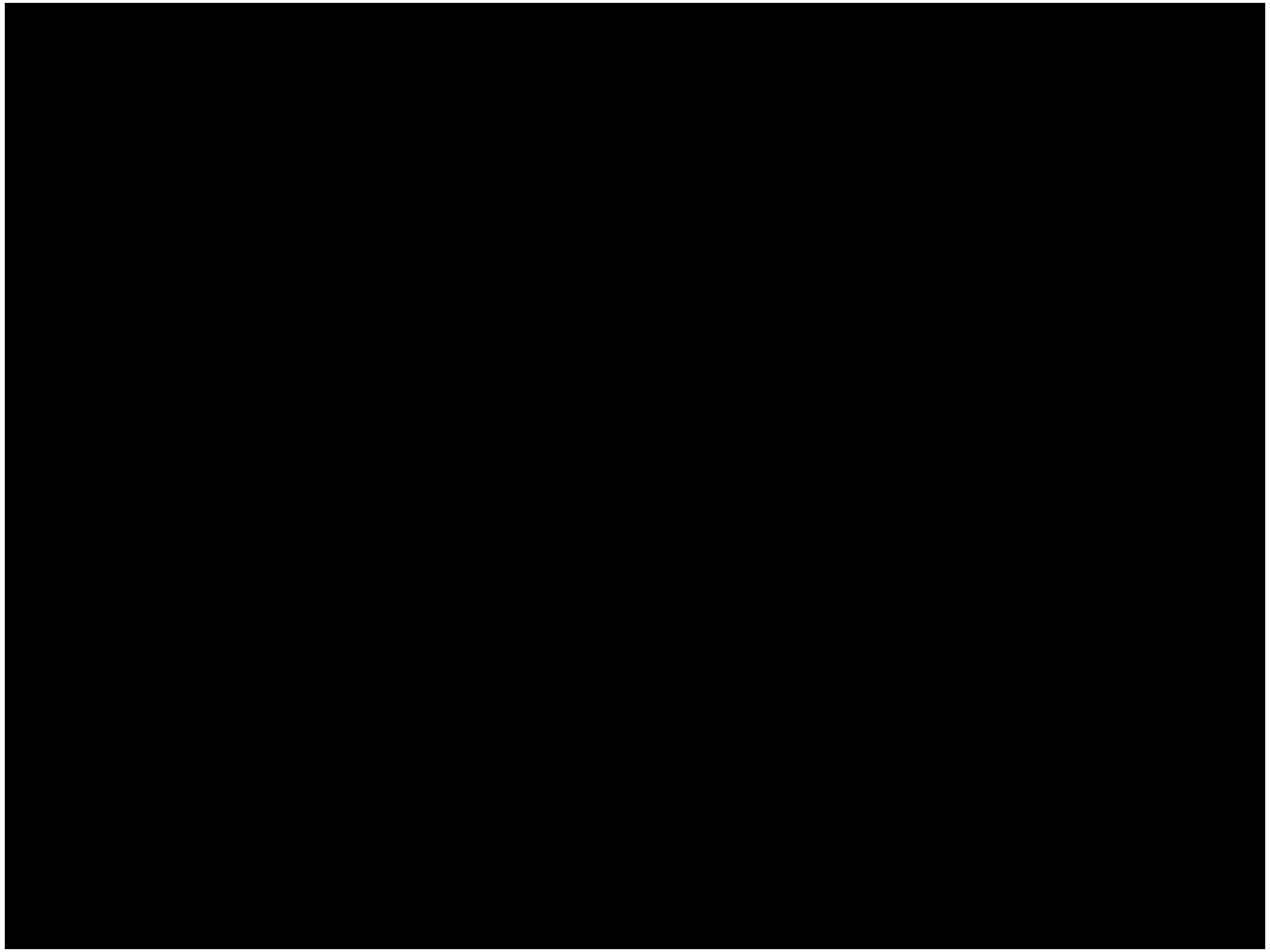
Autonomous Driving: Supervision



Supervised Learning Procedure:

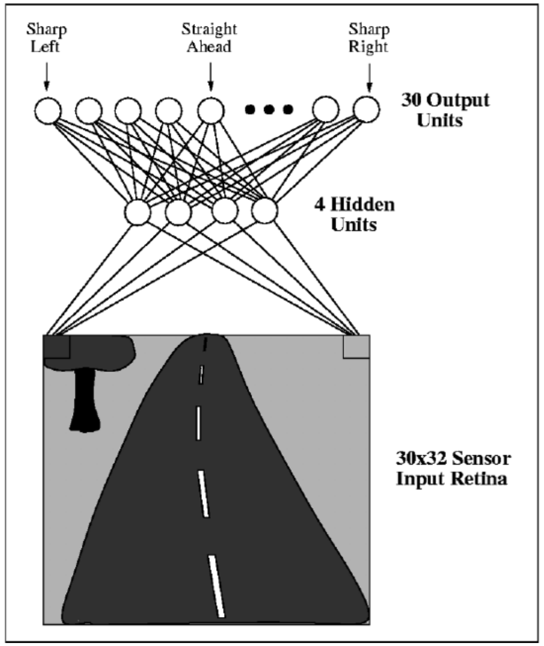
- Drive car
- Collect camera images and steering angles
- Linear Neural Net maps camera images to steering angles

Autonomous Driving: Supervision



ALVINN, Pomerleau, 1989

Autonomous Driving: Supervision



Supervised Learning Procedure:

- Drive car
- Collect camera images and steering angles
- Linear Neural Net maps camera images to steering angles

But this is insufficient.
Failure Rate is too high!

Autonomous Driving: Post-mortem

- Insufficient Model Capacity?

Linear predictor sufficient in imitation learning case

- Too small of a dataset?

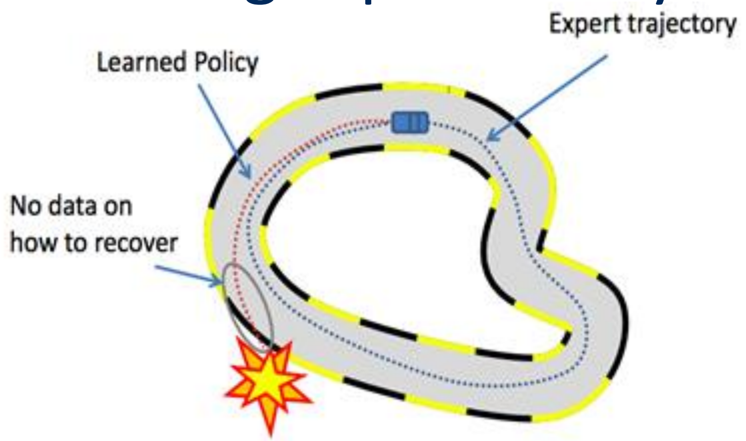
Larger training set data does not improve performance

Hold-out errors close to training errors

Autonomous Driving: Post-mortem

Real Problem: Errors Cascade:

- Algorithm makes small error with small probability ε
- Steer different than a human driver
- New unencountered images = unencountered states
- Further, larger errors with larger probability



Imitation Learning: Covariate Shift

E[Total errors] $\lesssim \varepsilon(T + (T-1) + (T-2) + \ldots + 1) \propto \varepsilon T^2$

Supervised Learning =
Independent data points

Error Bound: **Tε over T decisions**

Structured Prediction
→ Highly correlated data
→Cascading errors

Best expected error: **O($T^2\varepsilon$) over T decisions**

Imitation Learning: DAgger

DAgger (Dataset Aggregation) :

- Uses Interaction
- Have human expert to provide correct execution

Expected error:
O(Tε) over T decisions
instead of **O(T²ε)**

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

Algorithm 3.1: DAGGER Algorithm.

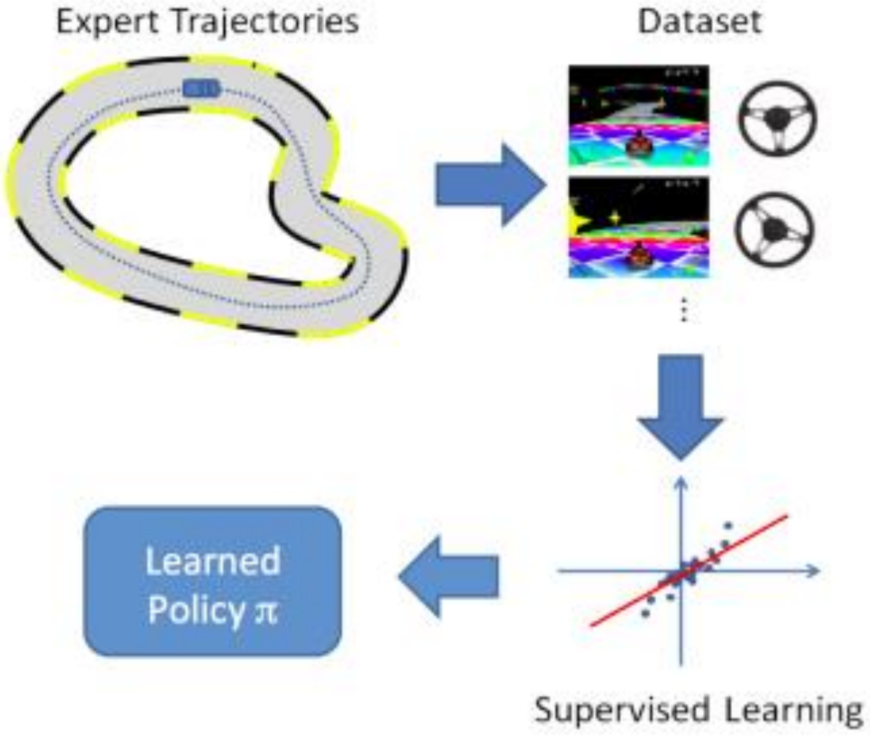
(DAgger) A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross et al. 2011

Imitation Learning: DAgger

Step 1: Start the same as the supervised learning attempt

- Collect data from experts driving (the human expert's policy is the optimal policy π^*) around a track

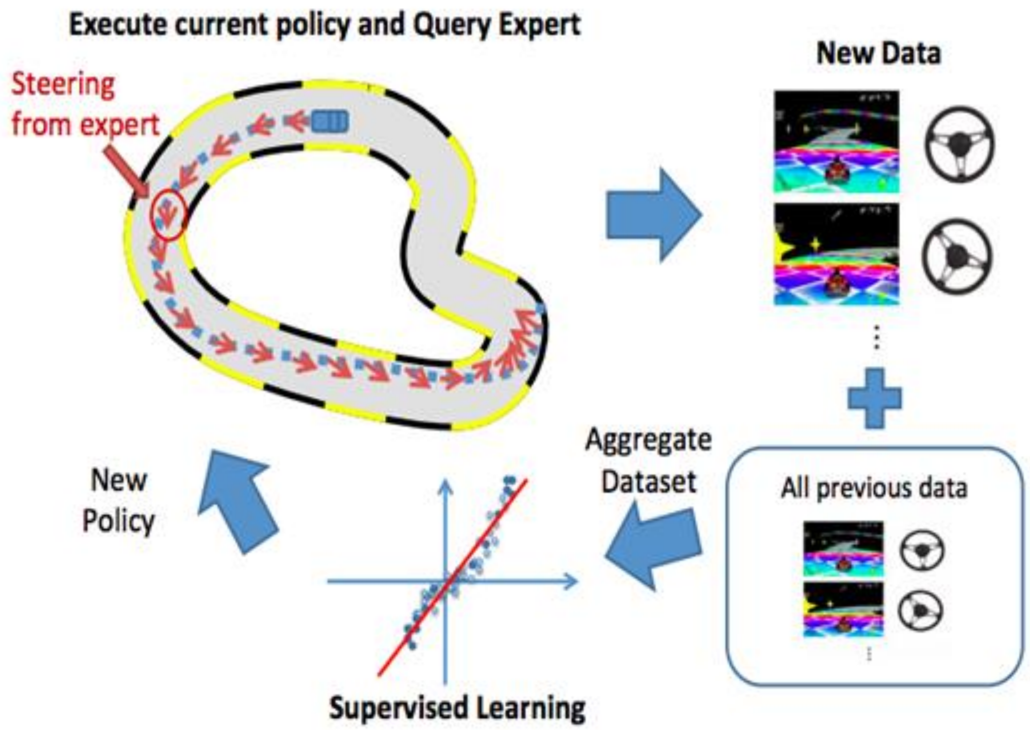
- Use expert trajectories with supervised learning techniques to obtain a policy (π_1)



Imitation Learning: DAgger

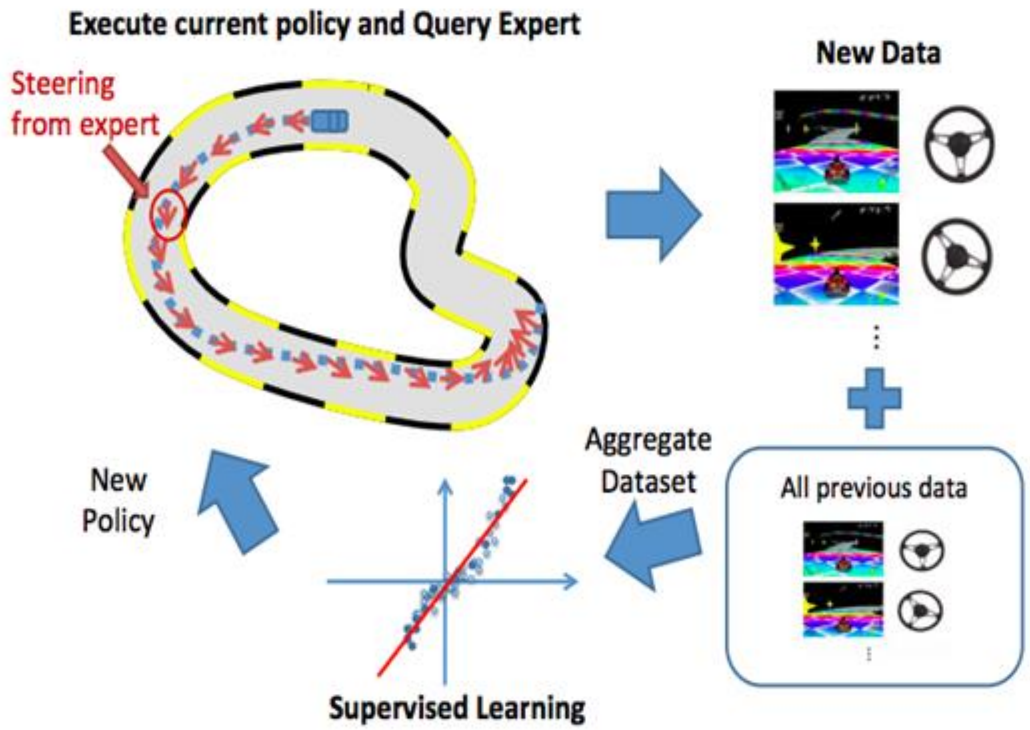
Step 2: Collect more data

- Set parameter $\boldsymbol{\beta}_1 \in [0, 1]$
- At each timestep collect data:
  - With probability $\boldsymbol{\beta}_1$, let the expert take actions
  - With probability (1- $\boldsymbol{\beta}_1$), take actions from the current policy (π_1), but record the expert's actions
- Combine the newly collected data with all the existing data to create an aggregated dataset
- Use supervised learning on the aggregated dataset to obtain a new policy (π_2)

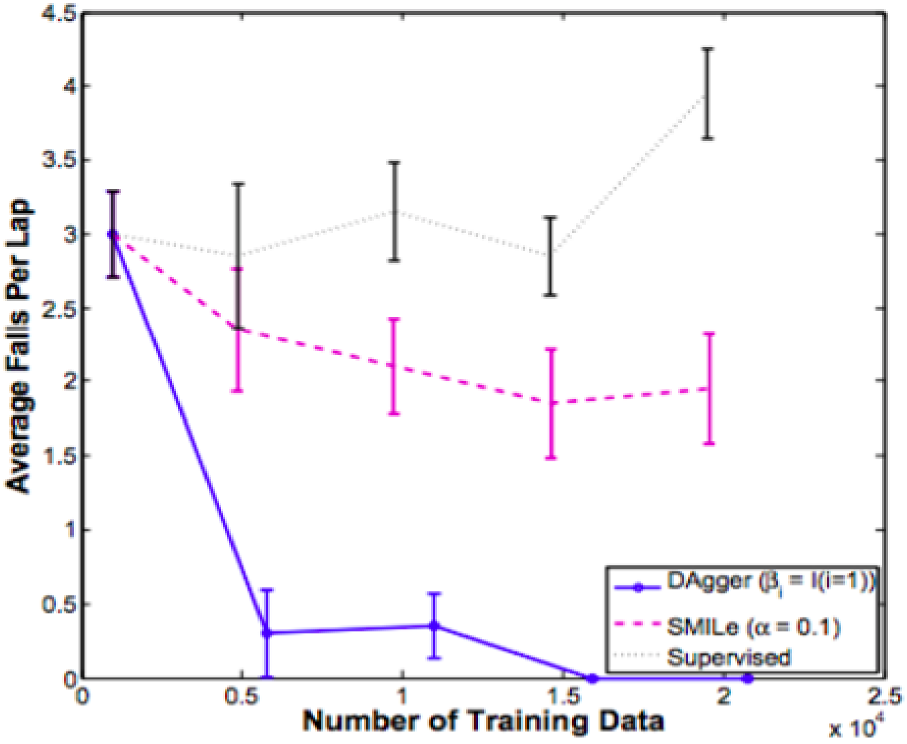


Imitation Learning: DAgger

Step 3: Iterate step 2, decaying $\boldsymbol{\beta}_i$ at every iteration, until the policy is converged



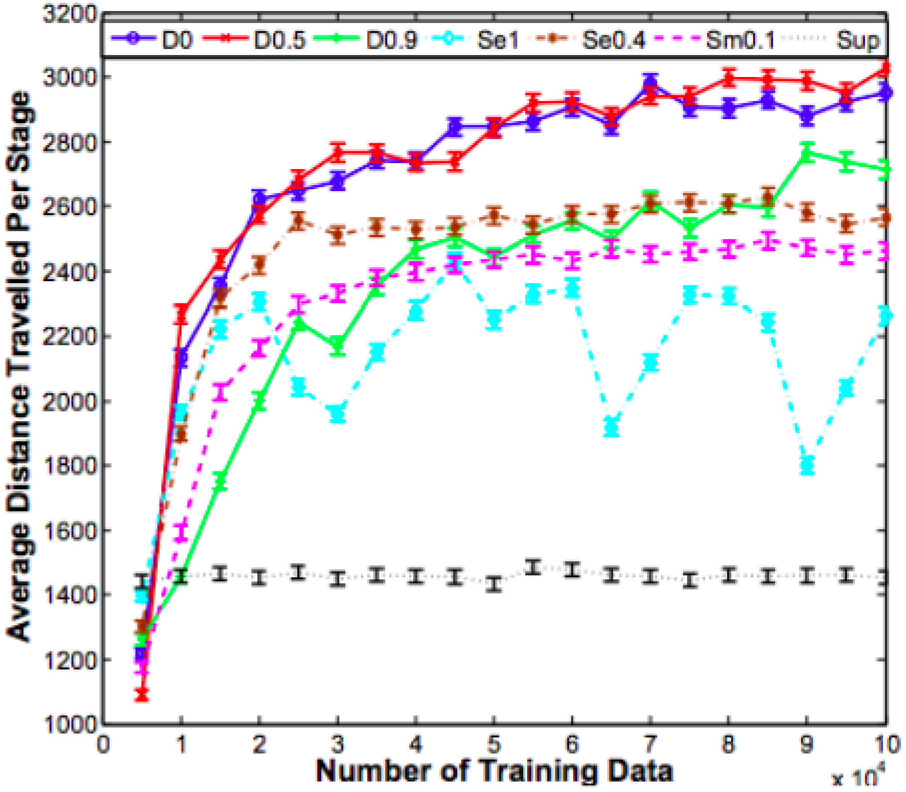
Imitation Learning: DAgger



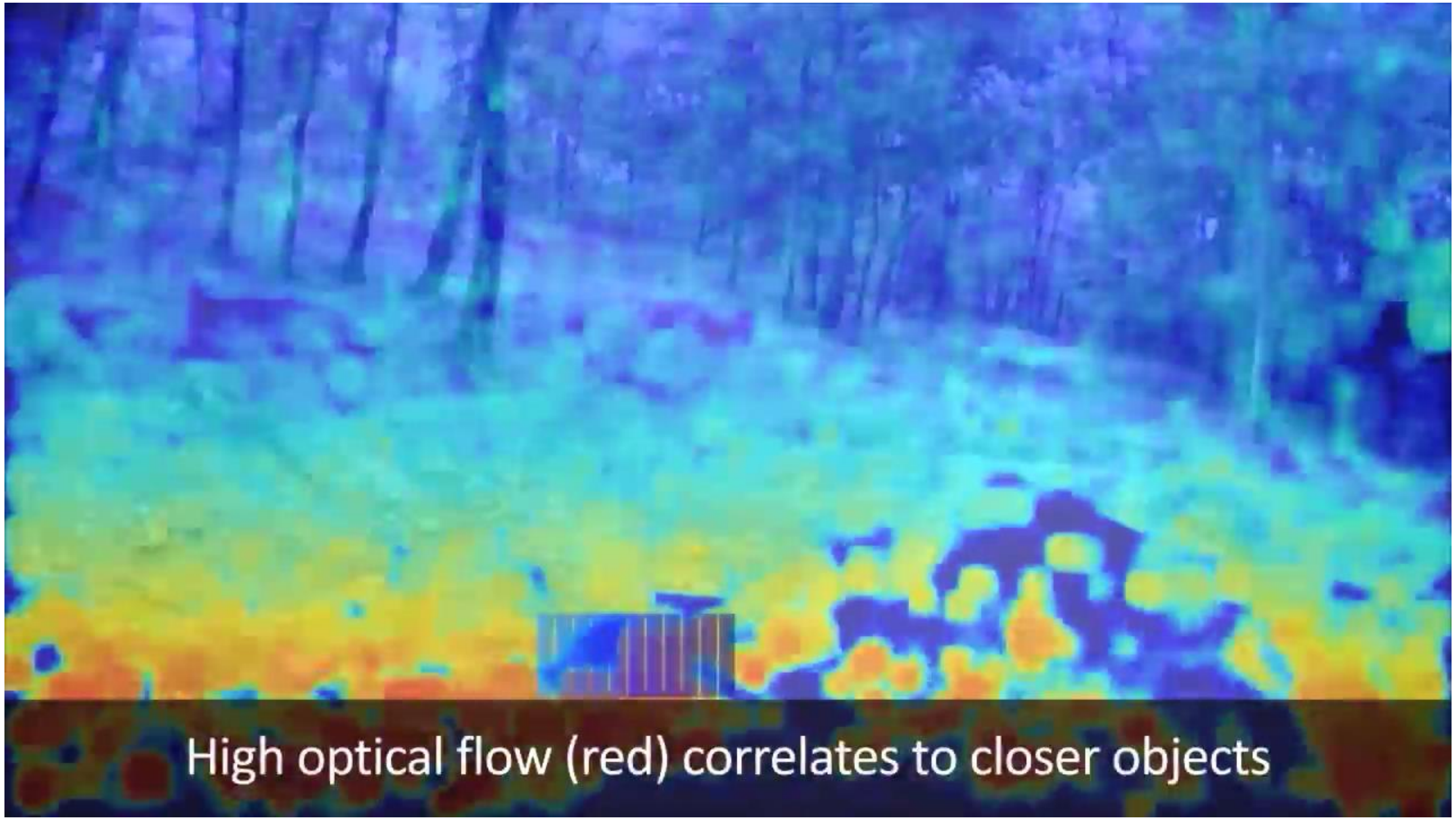
- Correct own mistakes
- Aggregation prevents forgetting previously learned situations

Imitation Learning: DAgger

Super Mario Bros



Imitation Learning: DAgger

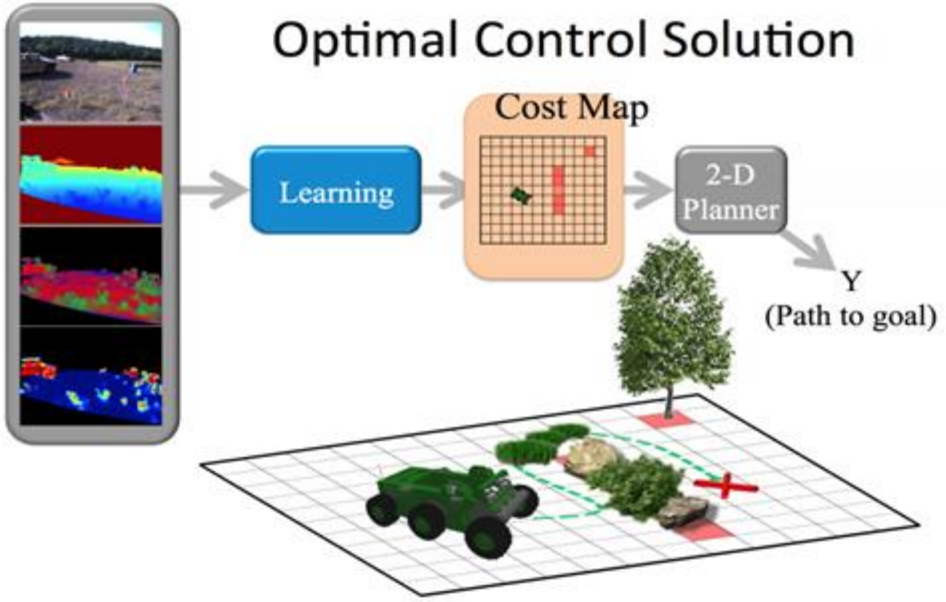


Anatomy of a Robotic System Architecture

- Sensors (laser RADAR, cameras) feed a perception system that computes a rich set of features

    color and texture, estimated depth, and shape descriptors of a LADAR point cloud.

- These features are then massaged into an estimate of "traversability" – a scalar value that indicates how difficult it is for the robot to travel across the location on the map
- "Cost map" is updated as robot moves and perceives



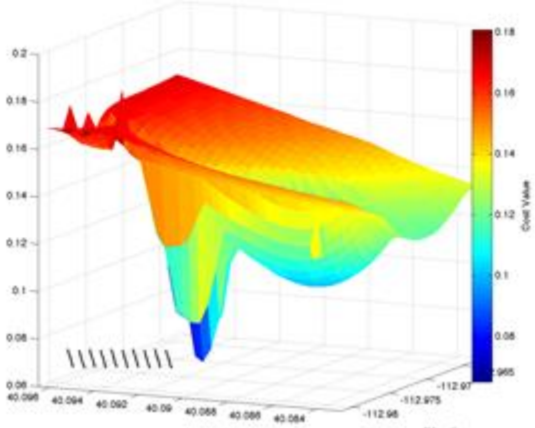
A Closer Look: Role of imitation learning

- Perception computes features that describe the environment
- **We need to connect perception and planning**
  - Task needs a long, coherent sequence of decisions to achieve the goal.
  - Requires planning and re-planning upon new information acquisition

- Manual engineering? → difficult
- Supervised learning method? → not interactive, unlikely to work
- Imitation learning techniques make it possible to automate the process.
- The imitation learning algorithm then must transform the feature vector of each state into a scalar cost value that the robot's planner uses to compute optimal trajectories

Cost Function Modelling

- Costing is one of the most difficult tasks in autonomous navigation.



- **Inverse Optimal Control**: Cost functions generalize more broadly than policies or value functions, so learn and plan with cost functions when possible, and revert to directly learning values or policies only when it is too computationally difficult infer cost functions

Inverse Optimal Control for Imitation Learning

- IOC attempts to find a cost function that maps perception features to a scalar cost signal
  - A teacher (human expert driver) drives the robot through a representative stretch of complex terrain.
  - The robot can use imitation to learn this cost-function mapping.
- Limitations
  - Assumes teacher’s driving pattern is near optimal.
  - Potentially substantially more computationally complex and sample inefficient than DAgger

Inverse Optimal Control for Imitation Learning

- Also called inverse reinforcement learning (Ng & Russell, 2000)
- Distinction between imitation learning and IOC
  - Imitation learning is the task of learning by mimicking expert demonstrations.
  - IOC is the problem of deriving a reward/cost function from observed behavior.
  - IOC is one approach to imitation learning, policy search approaches like DAgger are another

- Long history
  - Linear-Quadratic-Regulator [Kalman, 1964]
  - Convex programming formulation for the multi-input, multi-output linear-quadratic problem [Boyd et al., 1994]

Inverse Optimal Control for Imitation Learning

- Enabling a cost function to be derived for essentially arbitrary stochastic control problems using convex optimization techniques – any problem that can be formulated as a Markov Decision Problem.
- Requiring a weak notion of access to the purported optimal controller e.g. access to example demonstrations.
- Statistical guarantees on the number of samples required to achieve good predictive performance and even stronger results in the online or no-regret setting that requires no probabilistic assumptions at all.
- Robustness to imperfect or near-optimal behavior and generalizations to probabilistically predict the behavior of such approximately optimal agents.
- Some algorithms further require only access to an oracle that can solve the optimal control problem with a proposed cost function a modest number of times to address the inverse problem

LEARCH: Learning to Search

- Best of both worlds
- Pure imitation +
  Inverse Optimal Control

```
# Take a sequence of MDPS and demonstrations [ℳ_i,ξ_i)]_{i=1}^N where MDP ℳ is a stochastic planning problems
      consisting of states, actions, and a transition function used for planning,
#  (optional) loss functions l_i: state, action->ℝ that measures deviations from the demonstrated plan,
# feature function f: state, action -> R^d  that describes states in terms of features meaningful for
      cost

def LEARCH({(ℳ_i,ξ_i)}_{i=1}^N, f, {l_i}_{i=1}^N = 0):
  s_0 = 0 #Initialize (log)-cost function, s_0 : ℝ^d → ℝ to zero
  for t in range(T): # run for T iterations
        D = [] # Initialize the data set to empty
        for i in range(N): # for each example in the data set
              c_i^l = e^{s_t(F_i)} − l_i^T # Compute costmap with optional loss augmentation
              μ_i^* = Plan(ℳ_i,c_i) # find the resulting optimal plan μ_i^* = argmin_μ c_i^l μ, μ consistent with ℳ_i
              # μ^*'s counts the time spent in state/actions pairs under the plan—
              # for deterministic MDPS this is simply an indicator of whether the optimal plan
              # visits that edge in the planning graph
              μ_i = [ ξ_i.count((s,a)) for (s,a) in ℳ_i ] #compute states-actions in demonstration
              # Generate positive and negative training examples:
        D_i = [ (f_i(s,a),  sign(μ_i^{*sa} − μ_i^{sa}), |μ_i^{*sa} − μ_i^{sa}|)  for (s,a) in ℳ_i ]
        # if |μ_i^{*sa} − μ_i^{sa}| = 0 for a state-action we can simply not generate that point
        D.append(D_i)
        h_t = Learn(D) #Train a regressor (or classifier) h_t: R^d->R  on the resulting weighted data set
        s_{t+1} = s_t + α_t h_t # Update the (log) hypothesis cost function
    return exp(s_T)
```

LEARCH: Learning to Search

- Consider a discretized grid of states that the robot can occupy.
- Teacher provides path from a start point to a goal point.
- Choose an initial cost function

For every iteration of the algorithm:

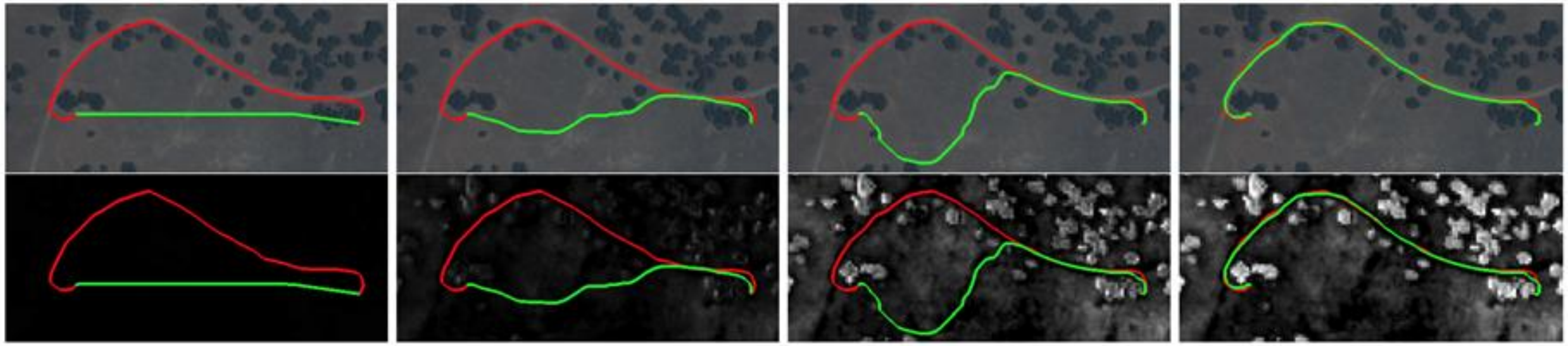
1. Compute the current best optimal plan/policy
2. Identify where the plan and teacher disagree and create a data set consisting of features and the direction in which we should modify the costs
3. Use a supervised learning algorithm to turn that data set into a simple predictor for updating costs
4. Compute a cost function as a (weighted) sum of the learned predictors.

LEARCH: Learning to Search

Initialize with constant cost → straight line path between start and end

Places where teacher visits but current plan does not → lower cost

Places where current plan visits but teacher does not → raise cost



A demonstration of the Learning to Search (LEARCH) algorithm applied to provide automated interpretation in traversability cost (Bottom) of satellite imagery (Top) for use in outdoor navigation. Brighter pixels indicate a higher traversability cost on a logarithmic scale. From left to right illustrates progression of the algorithm, where we see the current optimal plan (green) progressively captures more of the demonstration (red) correctly.

Imitation Learning: Challenges

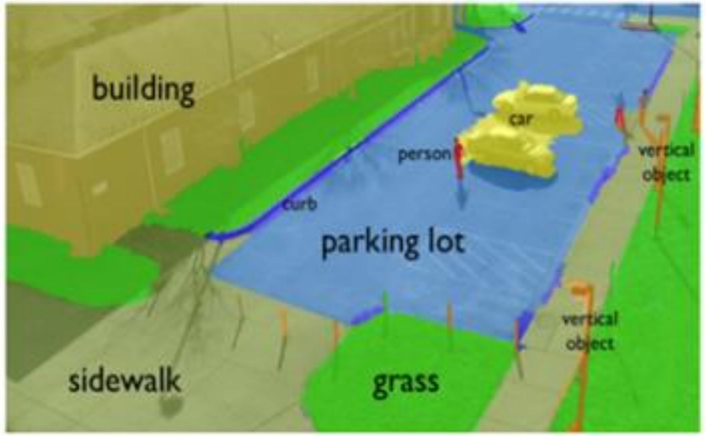
Problems:

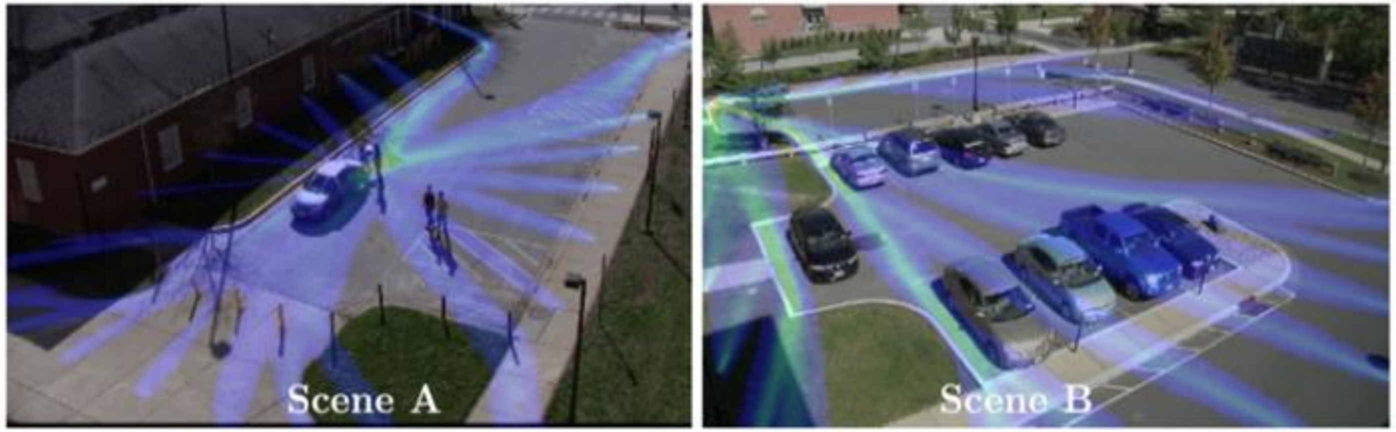
- Teacher is not truly an optimal controller
- World does not operate as simple Markov Decision Process
- Given a single behavior, there are many cost functions that lead to the same behavior (indeterminate)

Two commonly used notions of successful IOC used in machine learning:

1. Consider a class of reward functions that are linear in a set of features that describe states. Approach guarantees that the policy found will have performance comparable to or better than that of the expert even when the reward function itself cannot be identified. [Abbeel, 2004]
2. Ignore whether the teacher is actually an optimal controller or even whether there is a reward function. Quantifies a notion of successful imitation, e.g. agreement with teacher's trajectory, then attempt to optimize that notion of agreement with the teacher. [Ratliff et al., 2006b, 2009b]

Uncertainty with Probabilistic Approaches





- Many recent IOC learning techniques manage uncertainty
- Make probabilistic predictions of what people (non-optimal agents) are likely to do in the real world (non-MDP environment). [Kitani et al., 2012, Ziebart et al., 2008a, Ziebart et al., 2008b, Ziebart et al., 2010, 2013, Baker et al., 2009]

Since the paper came out…

- AggraVaTe: Reinforcement and Imitation Learning via Interactive No-Regret Learning
- Deeply AggreVaTeD: Differentiable Imitation Learning for Sequential Prediction

- Learning from Demonstrations for Real World RL
- Guided Policy Search
- Guided Cost Learning / Generative Adv. Imitation Learning
- One Shot Imitation Learning
- Third-Person Imitation Learning

And many more!

References

All images taken from one of the following sources:

1. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross, Gordon & Bagnell (2010). (DAgger algorithm)
2. An Invitation to Imitations, Ross (2015)

3. John Schulman's 2015 lecture at UC Berekley on DAgger: http://rll.berkeley.edu/deeprlcourse-fa15/docs/2015.10.5.dagger.pdf

Additional sources are:

4. Efficient Reductions for Imitation Learning, Ross, Bagnell (2010) (SMILe algorithm)
5. Efficient Reductions for Imitation Learning Supplementary Material, Ross, Bagnell (2010)

Agenda

- Invitation to Imitation
- DAGGER: Dataset Aggregation
- End-to-End learning for self-driving
- Behavioral Cloning from Observation

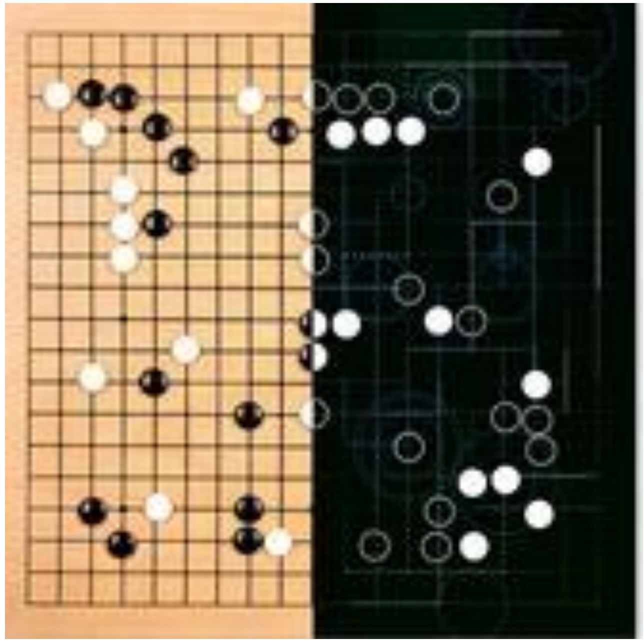
- Open-Problems and Project Ideas
- Logistics
- Presentation Sign-ups

RL in Recent Memory

Atari

DQN (Mnih et al. 2013)
DAGGER (Guo et al, 2014)
Policy Gradients (Schulman et al 2015)
DDPG (Lillicrap et al. 2015)
A3C (Mnih et al. 2016)

Go

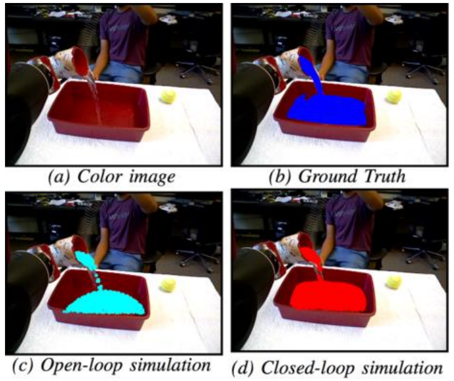


Policy Gradients
+
Monte Carlo Tree Search
(Silver et al. 2016)
…

Robotics

Levine et al. (2015)
Krishnan, G. et al (2016)
Rusu et al (2016)
Bojarski et al. (2016) nVidia
…

Success Stories for Learning in Robotics



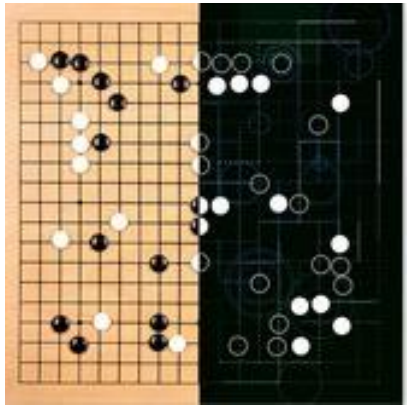
Mason & Salisbury 1985
Srinivasa et al 2010
Berenson 2013
Odhner1 et al 2014
Chavan-Dafle et al 2014
Yamaguchi, et. al, 2015
…

Li , Allen et al. 2015
Yahya et al, 2016
Schenck et al. 2017
Mar et al. 2017
Laskey et al 2017
Quispe et al 2018
…

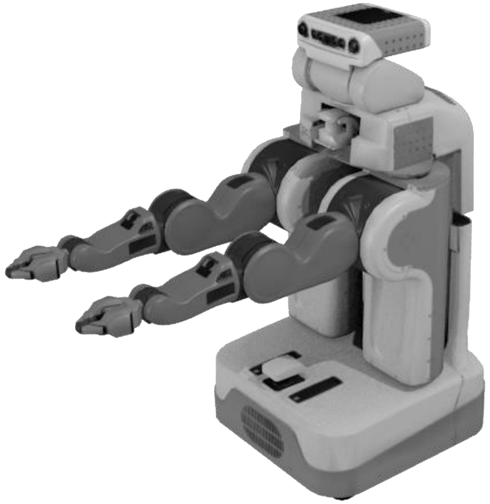
Mishra et al 1987
Ferrari & Canny, 1992
Ciocarlie & Allen, 2009
Dogar & Srinivasa, 2011
Rodriguez et al. 2012
Bohg et al 2014

Pinto & Gupta, 2016
Levine et al 2016
Mahler et al 2017
Jang et al 2017
Viereck et al 2017
...

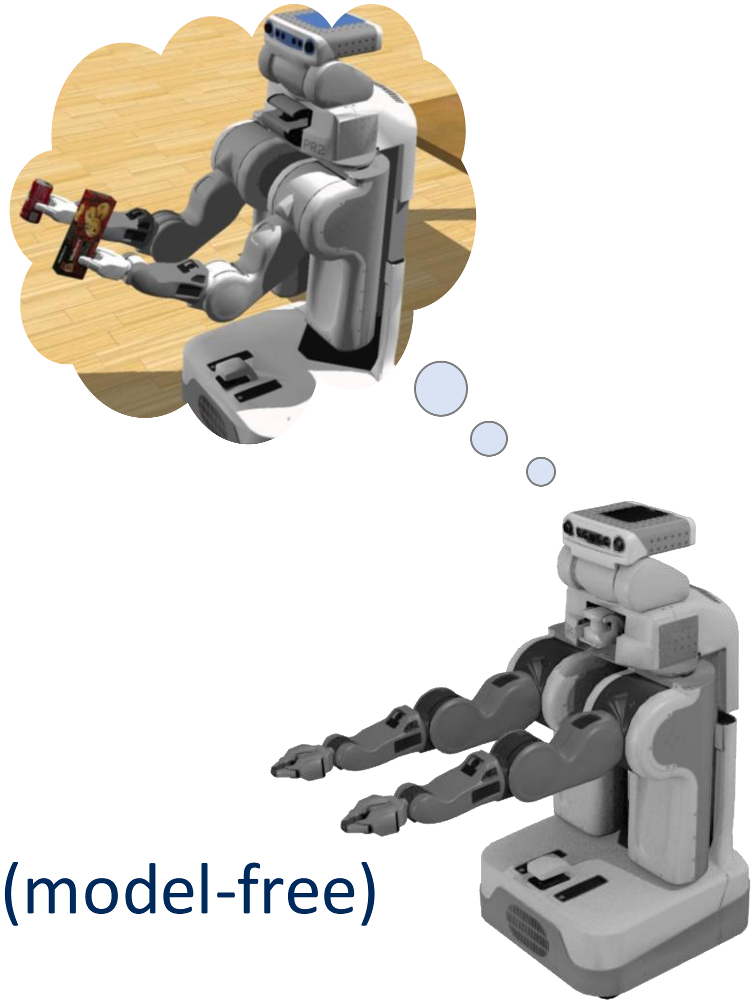
Going from Go to Robot/Control



- Known Environment vs Unstructured/Open World
- Need for Behavior Transfer
- Discrete vs Continuous States-Actions
- Single vs Variable Goals
- Reward Oracle vs Reward Inference



Other Open Problems



- Single algorithm for multiple tasks
- Learn new tasks very quickly
- Reuse past information about related problems
- Reward modelling in open environment
- How and what to build a model of?
- How much to rely on the model vs direct reflex (model-free)
- Learn without interaction if seen a lot of data

What this course plans to cover

- Imitation Learning: Supervised
- Policy Gradient Algorithms
- Actor-Critic Methods
- Value Based Methods
- Distributional RL
- Model-Based Methods
- Imitation Learning: Inverse RL
- Exploration Methods
- Bayesian RL
- Hierarchical RL

Agenda

- Invitation to Imitation
- DAGGER: Dataset Aggregation
- End-to-End learning for self-driving
- Behavioral Cloning from Observation

- Open-Problems and Project Ideas
- Logistics
- Presentation Sign-ups

Presentations

Jan 21

- Need 8 students – 4 teams of 2.
- Presentation Review Friday and/or Sat (video call) – (exception)

Jan 28

- Need 8 students – 4 teams of 2.
- Presentation Review Tues Jan 21 and Wed Jan 22 (week in advance)