10703 Deep Reinforcement Learning

Imitation Learning - 1

Tom Mitchell

November 4, 2018

Recommended readings:

Used Materials

- Much of the material and slides for this lecture were borrowed from Katerina Fragkiadaki, and Ruslan Salakhutdinov

So far in the course

Reinforcement Learning: Learning policies guided by sparse rewards, e.g., win the game.

- Good: simple, cheap form of supervision
- Bad: High sample complexity

Where is it successful so far?

- In simulation, where we can afford a lot of trials, easy to parallelize
- Not in robotic systems:
  - action execution takes long
  - we cannot afford to fail
  - safety concerns

Offroad navigation

Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain, Silver et al. 2010

Reward shaping

Ideally we want dense in time rewards to closely guide the agent closely along the way.

Who will supply those shaped rewards?

1. We will manually design them: *"cost function design by hand remains one of the 'black arts' of mobile robotics, and has been applied to untold numbers of robotic systems"*
2. We will learn them from demonstrations: *"rather than having a human expert tune a system to achieve desired behavior, the expert can demonstrate desired behavior and the robot can tune itself to match the demonstration"*

Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain, Silver et al. 2010

Reward shaping

Ideally we want dense in time rewards to closely guide the agent closely along the way.

Who will supply those shaped rewards?

1. We will manually design them: *"cost function design by hand remains one of the 'black arts' of mobile robotics, and has been applied to untold numbers of robotic systems"*
2. We will learn them from demonstrations: *"rather than having a human expert tune a system to achieve desired behavior, the expert can demonstrate desired behavior and the robot can tune itself to match the demonstration"*

Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain, Silver et al. 2010

Learning from Demonstrations

Learning from demonstrations a.k.a. Imitation Learning:

Supervision through an expert (teacher) that provides a set of demonstration trajectories: sequences of states and actions.

Imitation learning is useful when it is easier for the expert to demonstrate the desired behavior rather than:

a) coming up with a reward function that would generate such behavior,

b) coding up with the desired policy directly.

and the sample complexity is managable

Imitation Learning

Two broad approaches :

- Direct: Supervised training of policy (mapping states to actions) using the demonstration trajectories as ground-truth (a.k.a. behavior cloning)
- Indirect: Learn the unknown reward function/goal of the teacher, and derive the policy from these, a.k.a. Inverse Reinforcement Learning

Experts can be:

- Humans
- Optimal or near Optimal Planners/Controllers

Outline

Supervised training

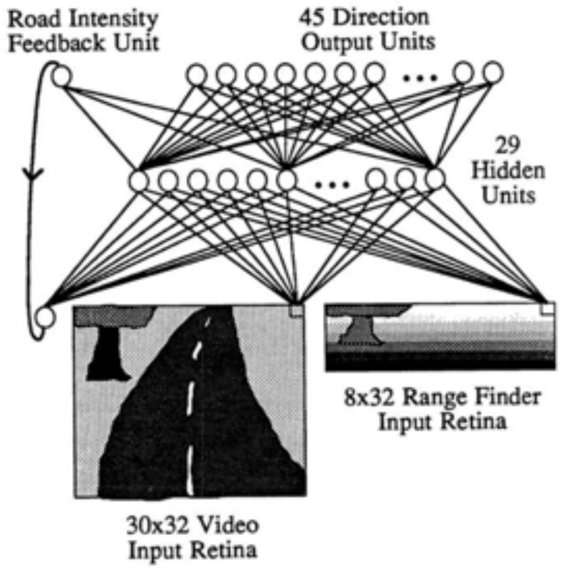
- Behavior Cloning: Imitation learning as supervised learning
- Compounding errors
- Demonstration augmentation techniques
- DAGGER

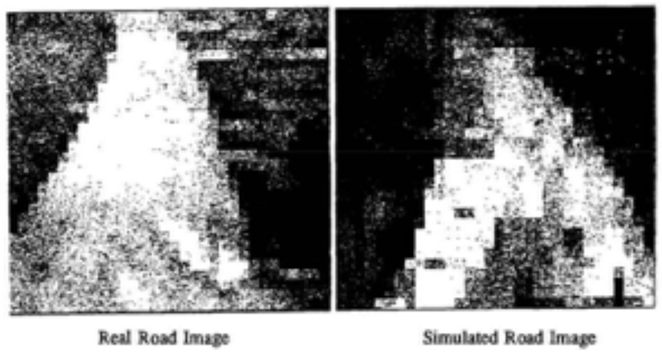
Inverse reinforcement learning

- Feature matching
- Max margin planning
- Maximum entropy IRL

Learning from Demonstration: ALVINN 1989

Road follower



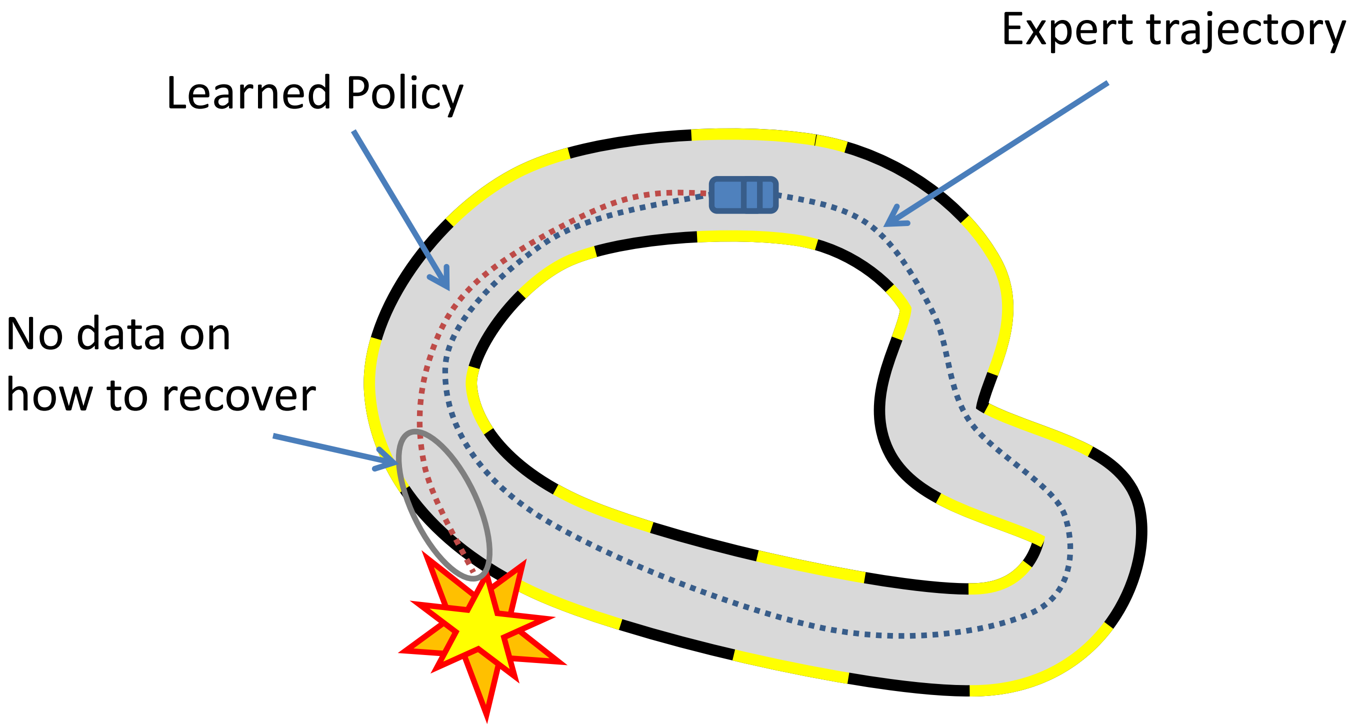


- Fully connected, single hidden layer, low resolution input from camera and lidar.
- Train to fit human-provided steering actions (i.e., supervised)
- First (?) use of data augmentation:

"In addition, the network must not solely be shown examples of accurate driving, but also how to recover (i.e. return to the road center) once a mistake has been made. Partial initial training on a variety of simulated road images should help eliminate these difficulties and facilitate better performance." ALVINN: An autonomous Land vehicle in a neural Network, [**Pomerleau 1989]**

Data Distribution Mismatch!

$$p_{\pi^*}(o_t) \neq p_{\pi_\theta}(o_t)$$



Data Distribution Mismatch!

	supervised learning	supervised learning + control (NAIVE)
train	$(x,y) \sim D$	$s \sim d_{\pi^*}$
test	$(x,y) \sim D$	$s \sim d_{\pi}$

Supervised Learning succeeds when training and test data distributions match.

But state distribution under learned π differs from those generated by π^*

Solution: Demonstration Augmentation

Change $p_{\pi^*}(\mathrm{o}_t)$ using demonstration augmentation!

Have expert label additional examples generated by the *learned* policy (e.g., drawn from $p_{\pi^{learned}}(o_t)$)

Solution: Demonstration Augmentation

Change $p_{\pi^*}(o_t)$ using demonstration augmentation!

Have expert label additional examples generated by the *learned* policy (e.g., drawn from $p_{\pi^{learned}}(o_t)$)

How?

1. use human expert
2. synthetically change observed o_t and corresponding u_t

Demonstration Augmentation: NVIDIA 2016

End to End Learning for Self-Driving Cars

Mariusz Bojarski NVIDIA Corporation Holmdel, NJ 07735
Davide Del Testa NVIDIA Corporation Holmdel, NJ 07735
Daniel Dworakowski NVIDIA Corporation Holmdel, NJ 07735
Bernhard Firner NVIDIA Corporation Holmdel, NJ 07735
Beat Flepp NVIDIA Corporation Holmdel, NJ 07735
Prasoon Goyal NVIDIA Corporation Holmdel, NJ 07735
Lawrence D. Jackel NVIDIA Corporation Holmdel, NJ 07735
Mathew Monfort NVIDIA Corporation Holmdel, NJ 07735
Urs Muller NVIDIA Corporation Holmdel, NJ 07735
Jiakai Zhang NVIDIA Corporation Holmdel, NJ 07735
Xin Zhang NVIDIA Corporation Holmdel, NJ 07735
Jake Zhao NVIDIA Corporation Holmdel, NJ 07735
Karol Zieba NVIDIA Corporation Holmdel, NJ 07735

Abstract

We trained a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands. This end-to-end approach proved surprisingly powerful. With minimum training data from humans the system learns to drive in traffic on local roads with or without lane markings and on highways. It also operates in areas with unclear visual guidance such as in parking lots and on unpaved roads.

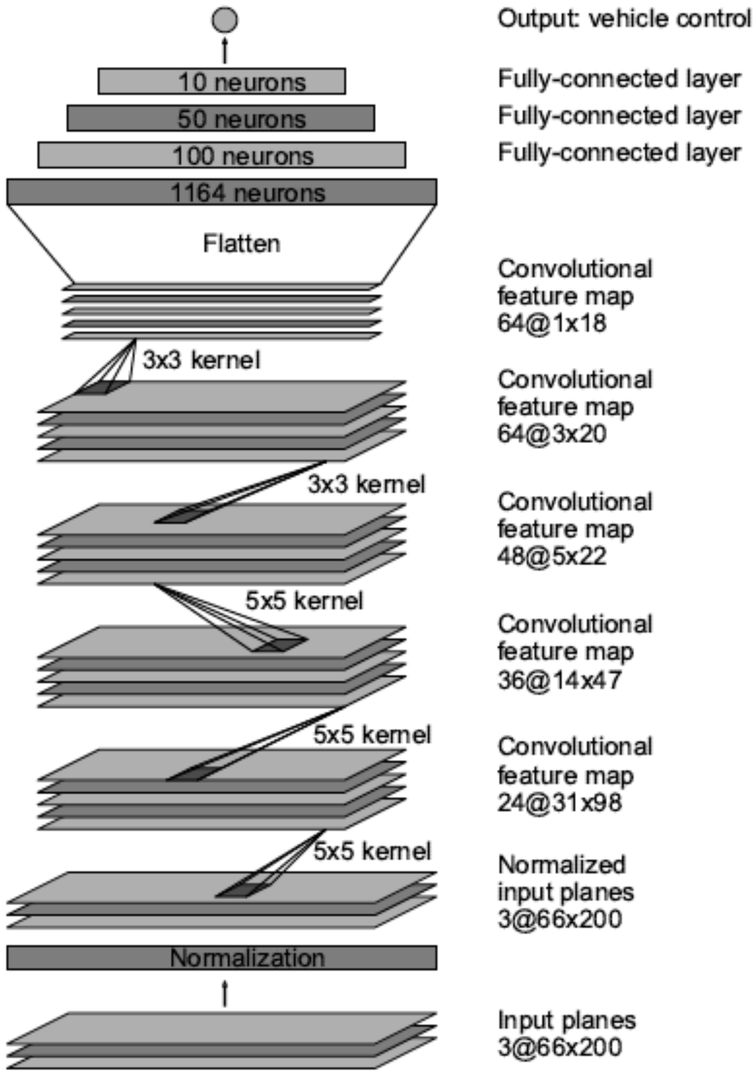
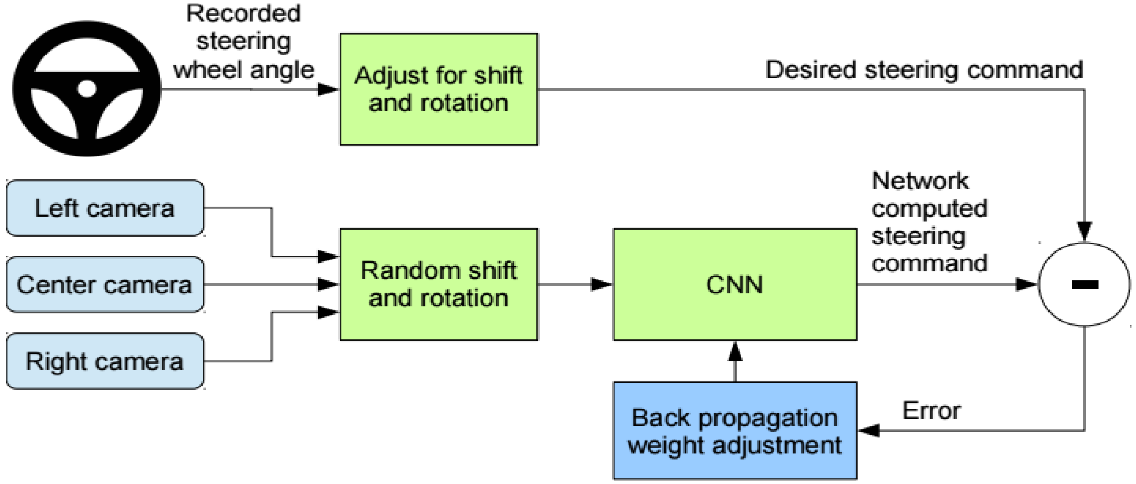
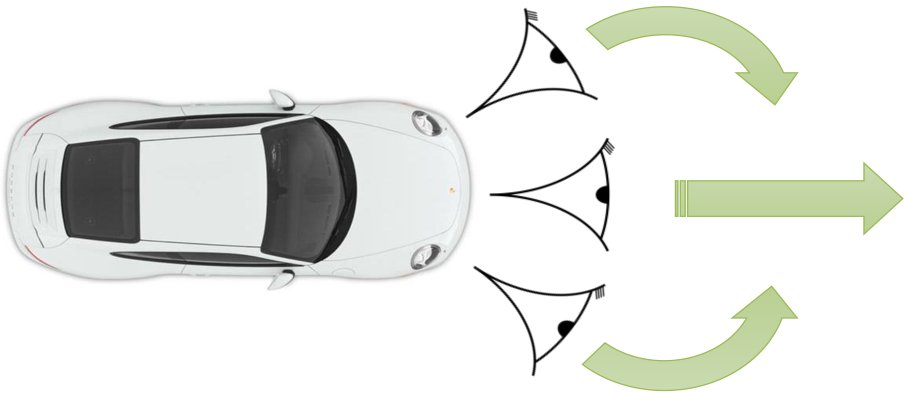


Figure 4: CNN architecture. The network has about 27 million connections and 250 thousand parameters.

Demonstration Augmentation: NVIDIA 2016



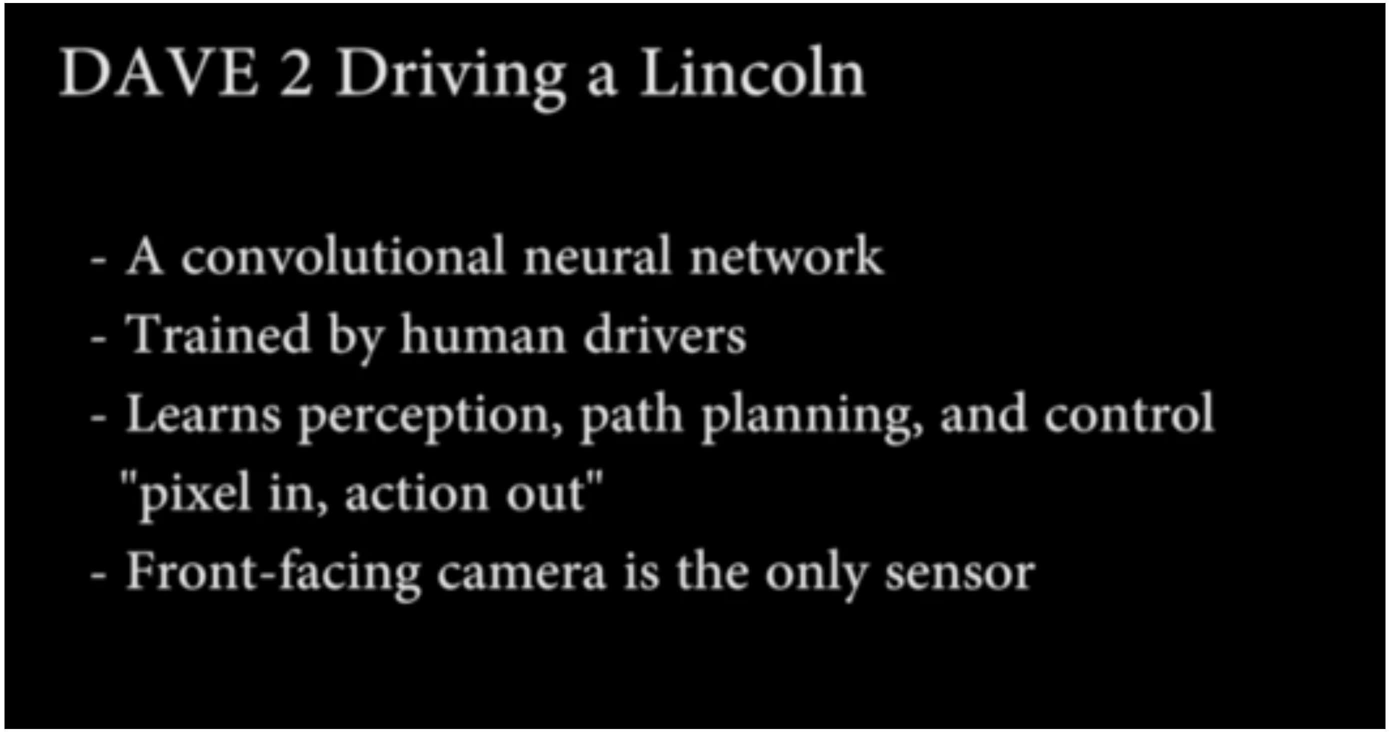


Additional, left and right cameras with automatic ground-truth labels to recover from mistakes

"*DAVE-2 was inspired by the pioneering work of Pomerleau [6] who in 1989 built the Autonomous Land Vehicle in a Neural Network (ALVINN) system*. Training with data from only the human driver is not sufficient. The network must learn how to recover from mistakes. …",

End to End Learning for Self-Driving Cars , Bojarski et al. 2016

Data Augmentation (2): NVIDIA 2016



Synthesizes new state-action pairs by rotating and translating input image, and calculating compensating steering command

[VIDEO]

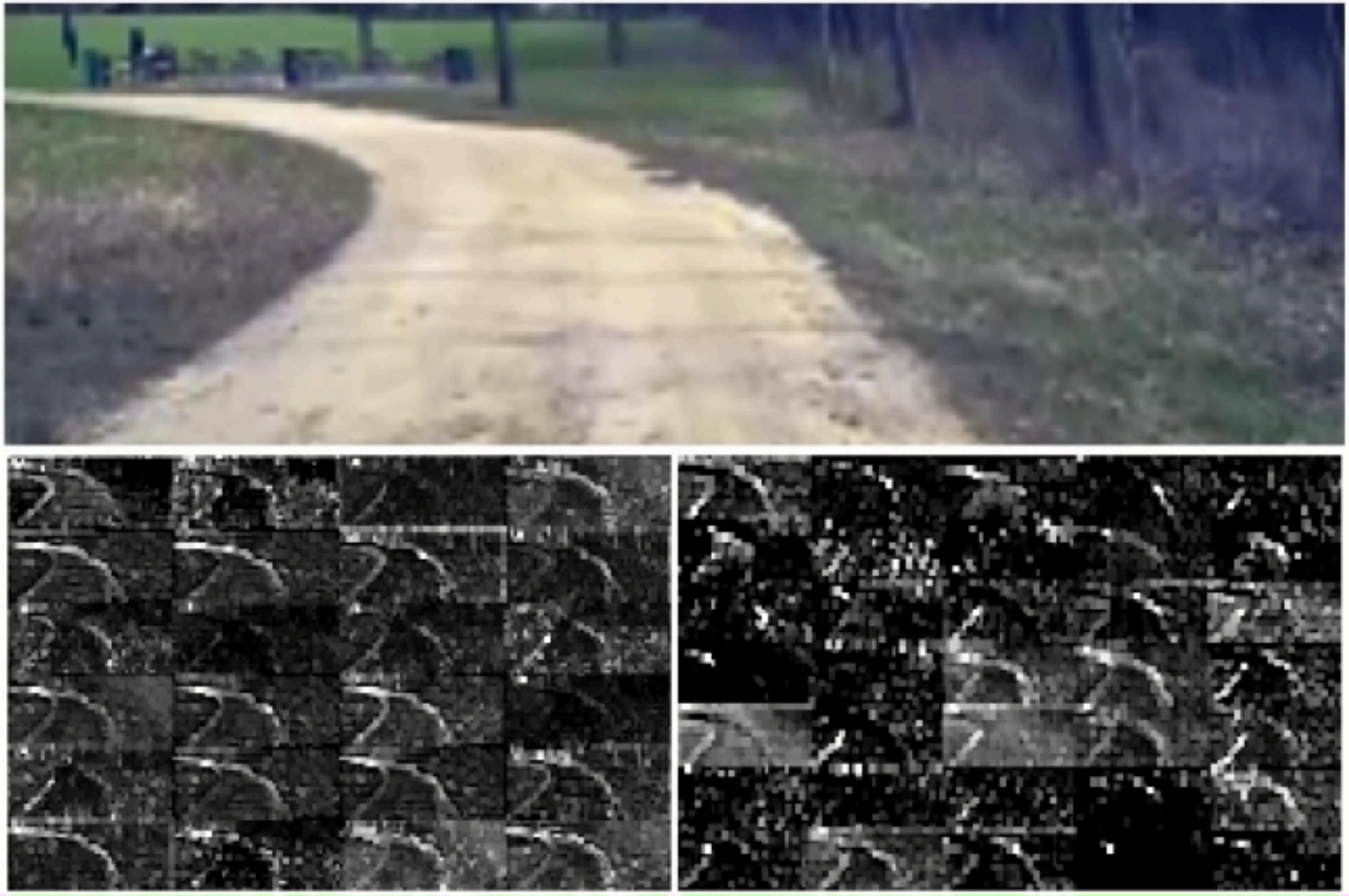
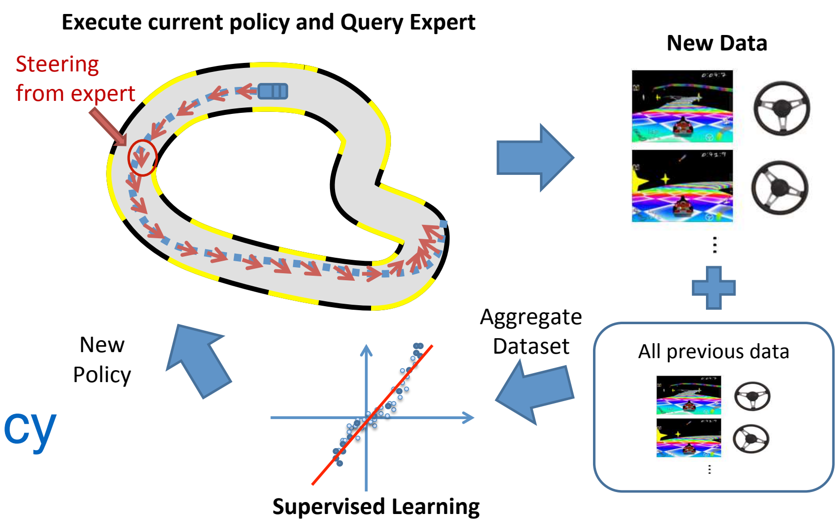


Figure 7: How the CNN “sees” an unpaved road. Top: subset of the camera image sent to the CNN. Bottom left: Activation of the first layer feature maps. Bottom right: Activation of the second layer feature maps. This demonstrates that the CNN learned to detect useful road features on its own, i. e., with only the human steering angle as training signal. We never explicitly trained it to detect the outlines of roads.

DAGGER

Dataset AGGregation: bring learner's and expert's trajectory distributions closer by iteratively labelling expert action for states generated by the current policy

1. train $\pi_\theta(u_t|o_t)$ from human data $\mathcal{D}_{\pi^*} = \{o_1, u_1, ..., o_N, u_N\}$
2. run $\pi_\theta(u_t|o_t)$ to get dataset $\mathcal{D}_\pi = \{o_1, ..., o_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions u_t
4. Aggregate: $\mathcal{D}_{\pi^*} \leftarrow \mathcal{D}_{\pi^*} \cup \mathcal{D}_\pi$
5. GOTO step 1.



Problems:

- execute an unsafe/partially trained policy
- repeatedly query the expert

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

DAGGER (in a real platform)

Application on drones: given RGB from the drone camera predict steering angles

http://robotwhisperer.org/bird-muri/ VIDEO

Learning monocular reactive UAV control in cluttered natural environments, Ross et al. 2013

DAGGER (in a real platform)

Caveats:

1. Is hard for the expert to provide the right magnitude for the turn without feedback of his own actions!
   Solution: provide visual feedback to expert

2. The expert's reaction time to the drone's behavior is large, this causes imperfect actions to be commanded.
   Solution: play-back in slow motion offline and record their actions.

3. Executing an imperfect policy causes accidents, crashes into obstacles.
   Solution: safety measures which again make the data distribution matching imperfect between train and test, but good enough.

Learning monocular reactive uav control in cluttered natural environments, Ross et al. 2013

Imitation Learning

Two broad approaches :

- Direct: Supervised training of policy (mapping states to actions) using the demonstration trajectories as ground-truth (a.k.a. behavior cloning)
- **Indirect**: Learn the unknown reward function/goal of the teacher, and derive the policy from these, a.k.a. **Inverse Reinforcement Learning**

Inverse Reinforcement Learning

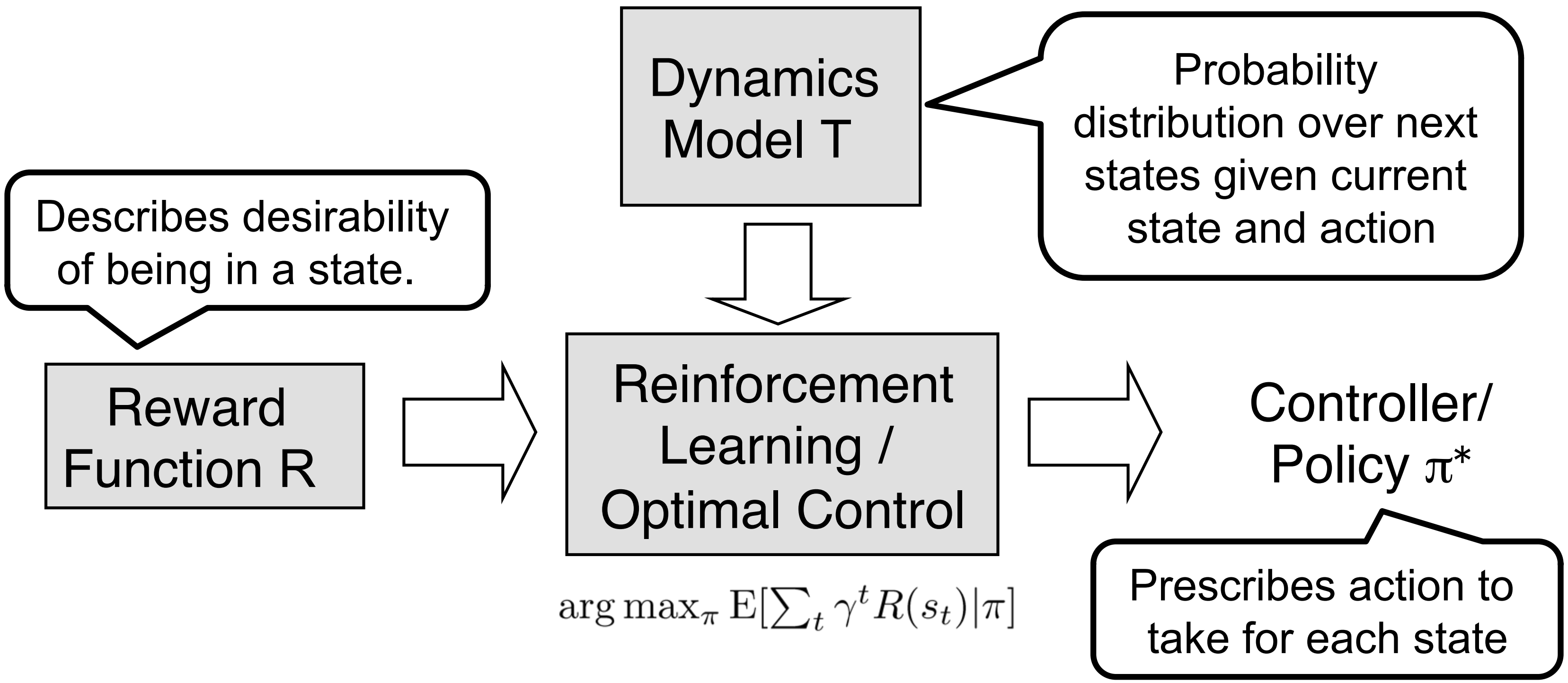


Diagram: Pieter Abbeel

Given π, let's recover R!

Problem Setup

- **Given**:
  - State space, action space
  - *No* reward function
  - Dynamics (sometimes) $T_{s,a}[s_{t+1}|s_t, a_t]$
  - Teacher's demonstration:

    $s_0, a_0, s_1, a_1, s_2, a_2, \ldots$
    $(= \text{trace of the teacher's policy } \pi^*)$

- **Inverse RL**
  - Can we recover R?
- **Apprenticeship learning via inverse RL**
  - Can we then use this R to find a good policy?
- **Behavioral cloning (*previous*)**
  - Can we directly learn the teacher's policy using supervised learning?

Assumptions (for now)

- Known Dynamics (transition model) T
- Reward is a linear function over fixed state features ϕ

Inverse RL with linear reward/cost function

Demonstration

$$y^* = (x_1, a_1) \to (x_2, a_2) \to (x_3, a_3) \to \cdots \to (x_n, a_n)$$

$$w^T f(y^*) = w^T \quad + w^T \quad + w^T \quad + \cdots\cdots + w^T$$

Expert trajectory reward/cost

Jain, Hu

Principle: Expert is optimal

- Find a reward function R^* which explains the expert behavior
- i.e., assume expert follows optimal policy, given her R^*

- Find R^* such that

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi^*] \geq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi] \quad \forall \pi$$

Feature Based Reward Function

(We assume reward is linear over features)

Let $R(s) = w^T \phi(s)$ where $w \in \mathbb{R}^n$, and $\phi : S \rightarrow \mathbb{R}^n$

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t w^T \phi(s_t)|\pi]$$

$$= w^T \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi]$$

$$= w^T \mu(\pi)$$

Feature Based Reward Function

(We assume reward is linear over features)

Let $R(s) = w^T \phi(s)$ where $w \in \mathbb{R}^n$, and $\phi : S \to \mathbb{R}^n$

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t w^T \phi(s_t)|\pi]$$

$$= w^T \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi]$$

$$= w^T \boxed{\mu(\pi)}$$

expected discounted sum of feature values or feature expectations—dependent on state visitation distributions

Sub/ting into $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi^*] \geq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi] \quad \forall \pi$

gives us: Find w^* such that ${w^*}^T \mu(\pi^*) \geq {w^*}^T \mu(\pi) \quad \forall \pi$

Idea

1. Guess an initial reward function $R(s)$
2. Learn policy $\pi(s)$ that optimizes $R(s)$
3. Whenever $\pi(s)$ chooses action different from expert $\pi^*(s)$
   - Update estimate of $R(s)$ to assure value of $\pi^*(s)$ > value of $\pi(s)$
4. Go to 2

Feature Matching

- Inverse RL starting point: find a reward function such that the expert outperforms other policies

$$\text{Let } R(s) = w^T \phi(s), \text{ where } w \in \mathbb{R}^n, \text{ and } \phi : S \rightarrow \mathbb{R}^n$$

$$\text{Find } w^* \text{ such that } {w^*}^T \mu(\pi^*) \geq {w^*}^T \mu(\pi) \quad \forall \pi$$

Here we define $\mu(\pi^*)$ as the expected discounted sum of feature values obtained by following this policy.

Given m trajectories generated by following the policy, we estimate it as

$$\hat{\mu}_E = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{\infty} \gamma^t \phi(s_t^{(i)})$$

Abbeel and Ng 2004

Feature Matching

- Inverse RL starting point: find a reward function such that the expert outperforms other policies

$$\text{Let } R(s) = w^T \phi(s), \text{ where } w \in \mathbb{R}^n, \text{ and } \phi : S \to \mathbb{R}^n$$

$$\text{Find } w^* \text{ such that } {w^*}^T \mu(\pi^*) \geq {w^*}^T \mu(\pi) \quad \forall \pi$$

- Observation in Abbeel and Ng, 2004: for a policy π to be guaranteed to perform as well as the expert policy $\mu*$, it suffices that the feature expectations match:

$$\|\mu(\pi) - \mu(\pi^*)\|_1 \leq \epsilon$$

Implies that for all w with $\|w\|_\infty \leq 1$:

$$|{w^*}^T \mu(\pi) - {w^*}^T \mu(\pi^*)| \leq \epsilon$$

Abbeel and Ng 2004

Why we wish to find a

$\tilde{\pi}$ such that $\|\mu(\tilde{\pi}) - \mu_E\|_2 \leq \epsilon$. For such a $\tilde{\pi}$, we would have that for any $w \in \mathbb{R}^k$ ($\|w\|_1 \leq 1$),

$$\begin{aligned} &|E[\textstyle\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi_E] - E[\textstyle\sum_{t=0}^{\infty} \gamma^t R(s_t)|\tilde{\pi}]| && (6) \\ &= |w^T \mu(\tilde{\pi}) - w^T \mu_E| && (7) \\ &\leq \|w\|_2 \|\mu(\tilde{\pi}) - \mu_E\|_2 && (8) \\ &\leq 1 \cdot \epsilon = \epsilon && (9) \end{aligned}$$

The first inequality follows from the fact that $|x^T y| \leq \|x\|_2 \|y\|_2$, and the second from $\|w\|_2 \leq \|w\|_1 \leq 1$. So the problem is reduced to finding a policy $\tilde{\pi}$ that induces feature expectations $\mu(\tilde{\pi})$ close to μ_E. Our

Abbeel and Ng 2004

Apprenticeship Learning [Abbeel & Ng, 2004]

- Assume $R_w(s) = w^T\phi(s)$ for a feature map $\phi : S \to \mathbb{R}^n$
- Initialize: pick some policy π_0
- Iterate for $i = 1, 2, \ldots :$
  - **"Guess" the reward function:**
    Find a reward function such that the teacher maximally outperforms all previously found policies

$$\max_{\gamma, w : \|w\|_2 \leq 1} \gamma$$

$$\text{s.t.} \quad w^T\mu(\pi^*) \geq w^T\mu(\pi) + \gamma \quad \forall \pi \in \{\pi_0, \pi_1, \ldots, \pi_{i-1}\}$$

  - **Find optimal control policy** π for the current guess of the reward function R_w
  - $\gamma \leq \varepsilon/2$ exit the algorithm

IRL in Simple Grid World (top two curves), Versus Three Supervised Learning Approaches

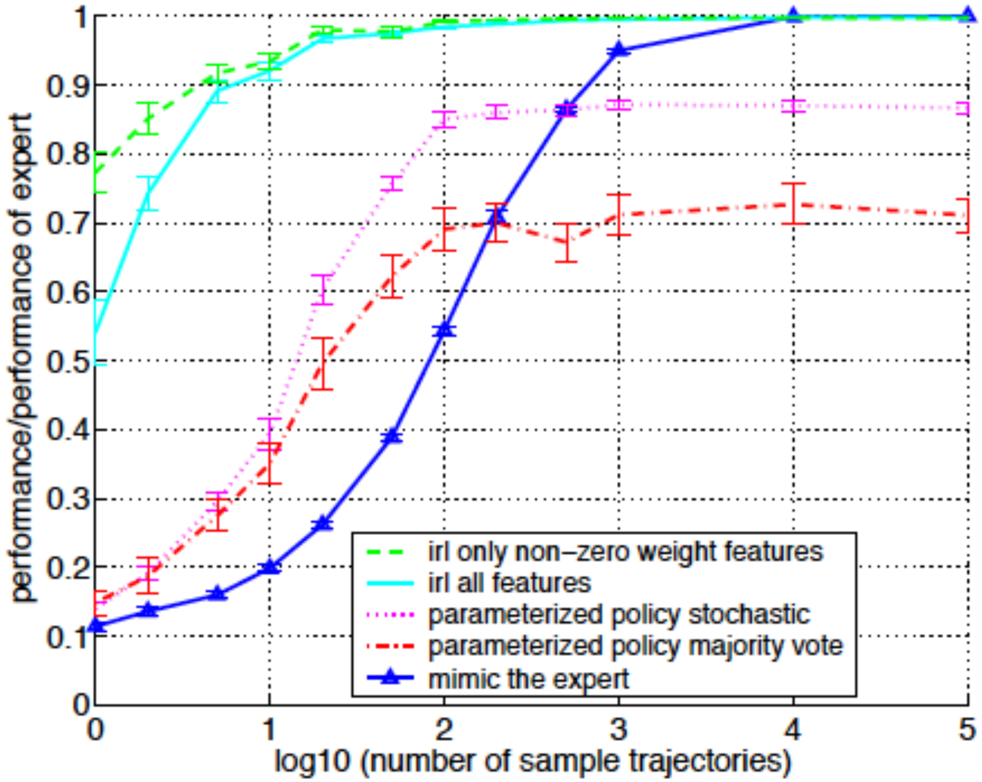


Figure 4. Plot of performance vs. number of sampled trajectories from the expert. (Shown in color, where available.) Averages over 20 instances are plotted, with 1 s.e. errorbars. Note the base-10 logarithm scale on the x-axis.

Abbeel and Ng 2004

Apprenticeship Learning [Abbeel & Ng, 2004]

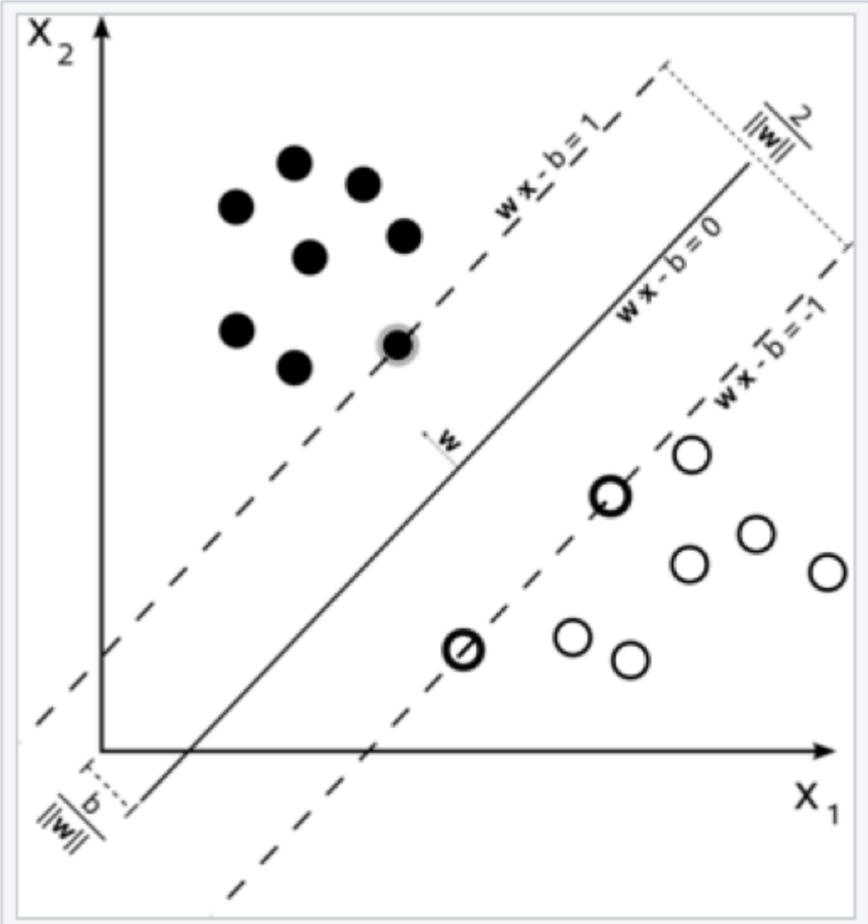
- Assume $R_w(s) = w^T \phi(s)$ for a feature map $\phi : S \to \mathbb{R}^n$
- Initialize: pick some policy π_0
- Iterate for $i = 1, 2, \ldots :$
  - **"Guess" the reward function:**
    Find a reward function such that the teacher maximally outperforms all previously found policies

$$\max_{\gamma, w: \|w\|_2 \leq 1} \gamma$$

$$\text{s.t.} \quad w^T \mu(\pi^*) \geq w^T \mu(\pi) + \gamma \quad \forall \pi \in \{\pi_0, \pi_1, \ldots, \pi_{i-1}\}$$

  - **Find optimal control policy** π for the current guess of the reward function R_w
  - $\gamma \leq \varepsilon/2$ exit the algorithm

Max-margin Classifiers



Here each point represents the feature expectations for one policy.

We can label them as the expert policy or not

And use SVM maximum margin algorithms to derive weights for the inferred reward function R

"Minimize $\|\vec{w}\|$ subject to $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$, for $i = 1, \ldots, n$"

Max-margin Classifiers

- We are given a training dataset of n points of the form

$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$$

- Where the y_i are either 1 or -1, each indicating the class to which the point $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a p-dimensional real vector.

- We want to find the "maximum-margin hyperplane" that divides the group of points $\vec{x}_i$, for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized.

- Any hyperplane can be written as the set of points $\vec{x}$ satisfying

$$\vec{w} \cdot \vec{x} - b = 0$$

where $\vec{w}$ is the normal vector the the hyperplane

Max Margin Planning

- Standard max margin:

$$\min_{w} \|w\|_2^2$$
$$\text{s.t.} \quad w^T \mu(\pi^*) \geq w^T \mu(\pi) + 1 \quad \forall \pi$$

Maximum Margin Planning, Ratliff et al. 2006

Max Margin Planning

- Standard max margin:

$$\min_w \|w\|_2^2$$
$$\text{s.t.} \quad w^T\mu(\pi^*) \geq w^T\mu(\pi) + 1 \quad \forall \pi$$

- "Structured prediction" max margin:

$$\min_w \|w\|_2^2$$
$$\text{s.t.} \quad w^T\mu(\pi^*) \geq w^T\mu(\pi) + m(\pi^*, \pi) \quad \forall \pi$$

- Justification: margin should be larger for policies that are very different from π^*

- Example: $m(\pi^*, \pi) =$ number of states in which π^* and π disagree

Maximum Margin Planning, Ratliff et al. 2006

Expert Suboptimality

- Structured prediction max margin with slack variables:

$$\min_{w,\xi} \|w\|_2^2 + C\xi$$

$$\text{s.t.} \quad w^T \mu(\pi^*) \geq w^T \mu(\pi) + m(\pi^*, \pi) - \xi \quad \forall \pi$$

- Can be generalized to multiple MDPs (could also be same MDP with different initial state)

$$\min_{w,\xi^{(i)}} \|w\|_2^2 + C \sum_i \xi^{(i)}$$

$$\text{s.t.} \quad w^T \mu(\pi^{(i)*}) \geq w^T \mu(\pi^{(i)}) + m(\pi^{(i)*}, \pi^{(i)}) - \xi^{(i)} \quad \forall i, \pi^{(i)}$$

Complete Max-margin Formulation

$$\min_{w} \|w\|_2^2 + C \sum_i \xi^{(i)}$$

$$\text{s.t.} \quad w^T \mu(\pi^{(i)*}) \geq w^T \mu(\pi^{(i)}) + m(\pi^{(i)*}, \pi^{(i)}) - \xi^{(i)} \quad \forall i, \pi^{(i)}$$

- Challenge: very large number of constraints.
- Solution: iterative constraint generation

Maximum Margin Planning, Ratliff et al. 2006

Example: Learn Cost Function of Expert Driver

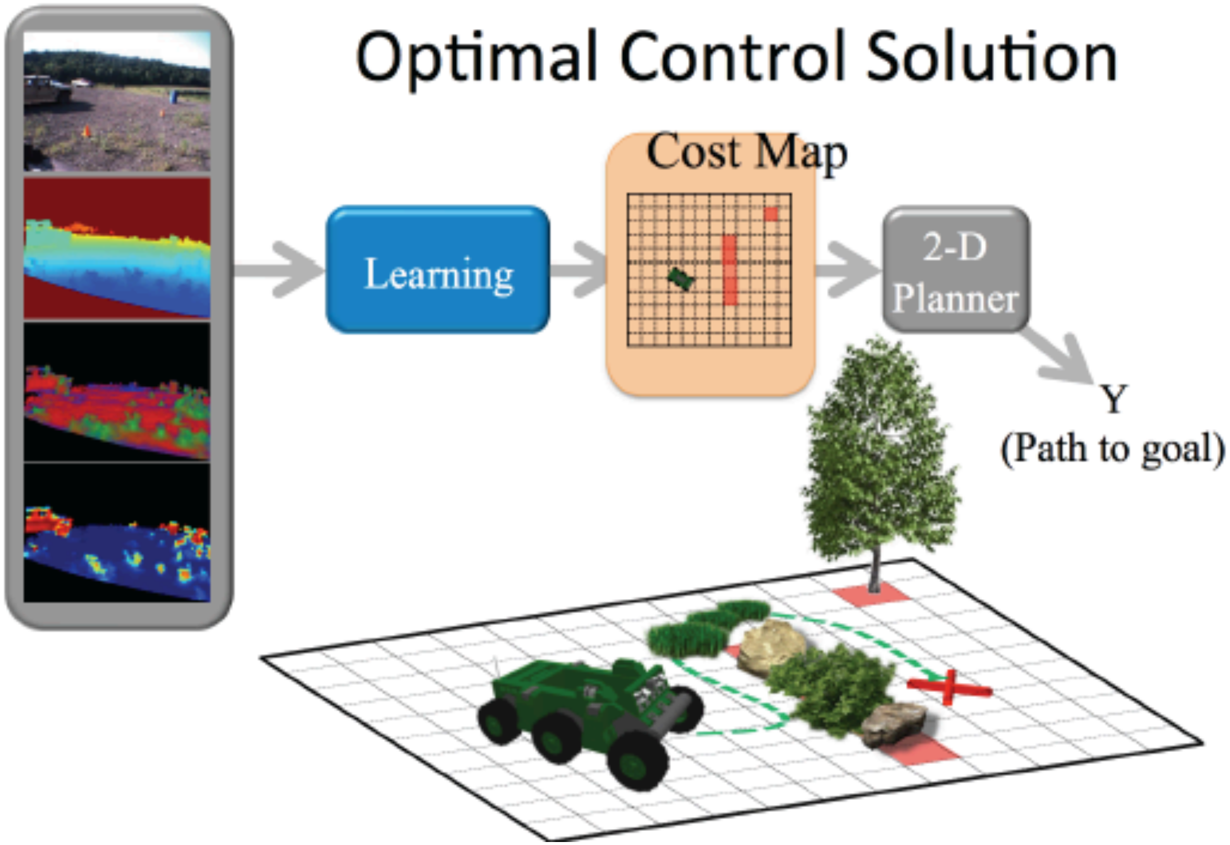
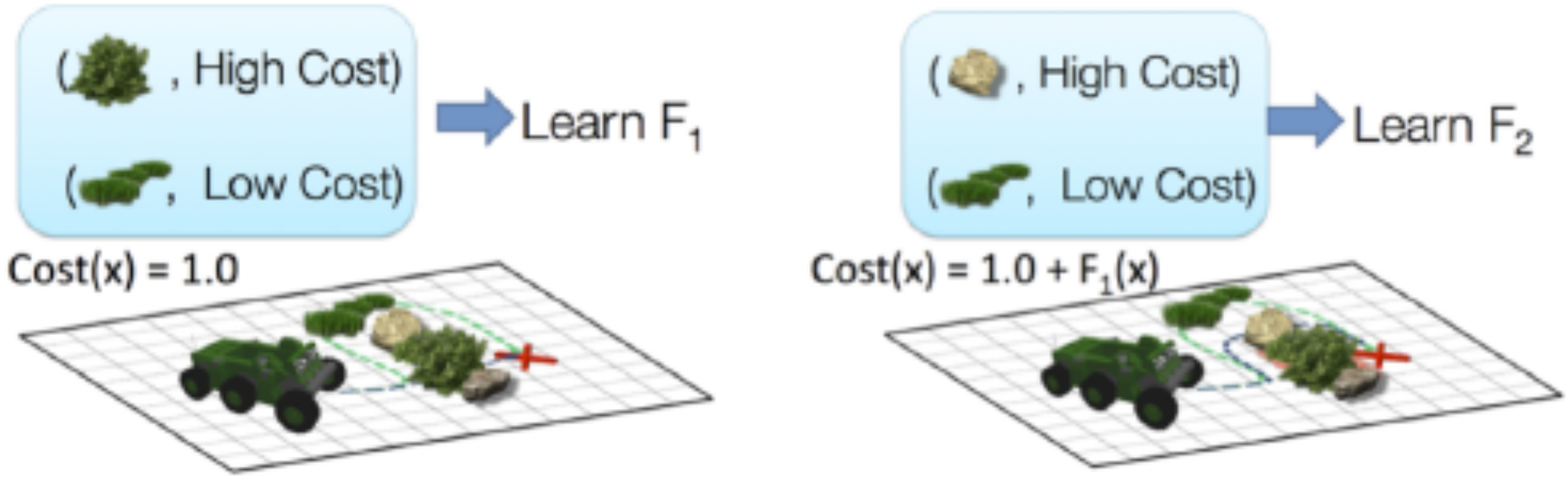


Figure 7: Components of a robot architecture: Sensors (LADAR, cameras) feed a perception system that computes a rich set of features (left side) developed in the computer vision and robotics fields. Depicted features include estimates of color and texture, estimated depth, and shape descriptors of a LADAR point cloud. Features that are not depicted here include estimates of terrain slope, semantic labels ("rock"), and other feature descriptors that can be assigned a location in a 2D grid map. 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.

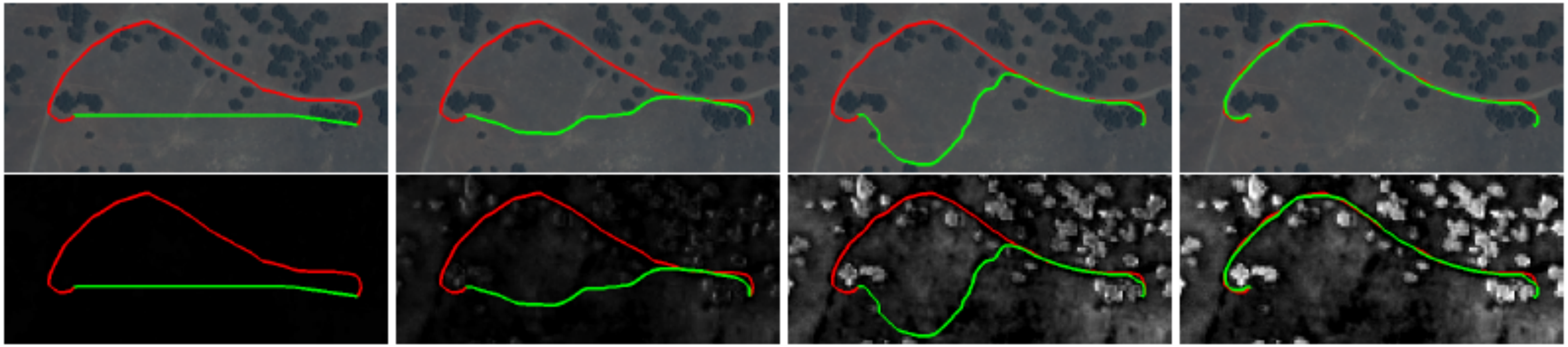
Nathan D Ratliff, David Silver, and J Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. Autonomous Robots, 27(1):25–53, 2009b.

Example: Learn Cost Function of Expert Driver

LEARCH Algorithm: Iteratively learn/refine a cost/reward function that makes expert driver appear optimal.



Example: Learn Cost Function of Expert Driver



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.

Something Different

- Learning from Demonstration

  →

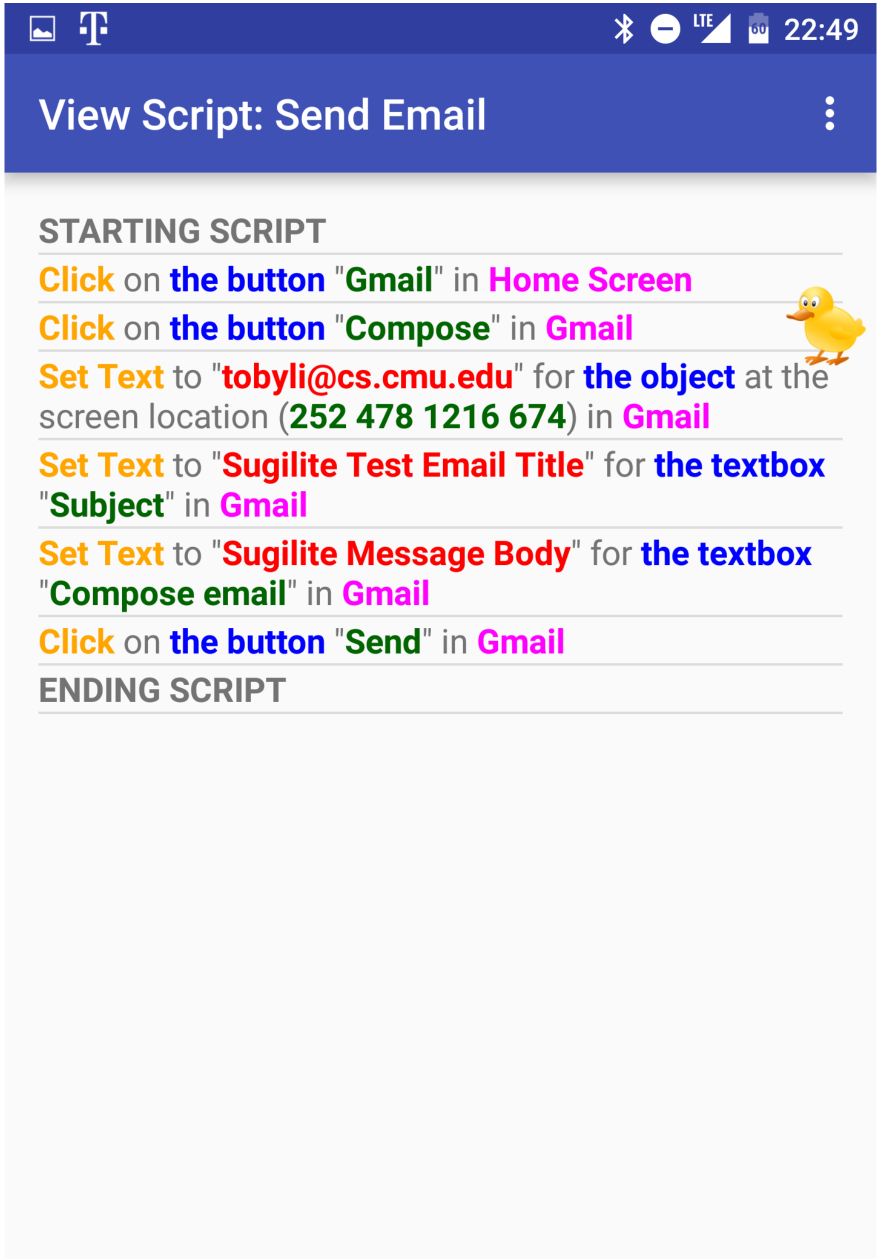
- Learning from Instruction more generally?

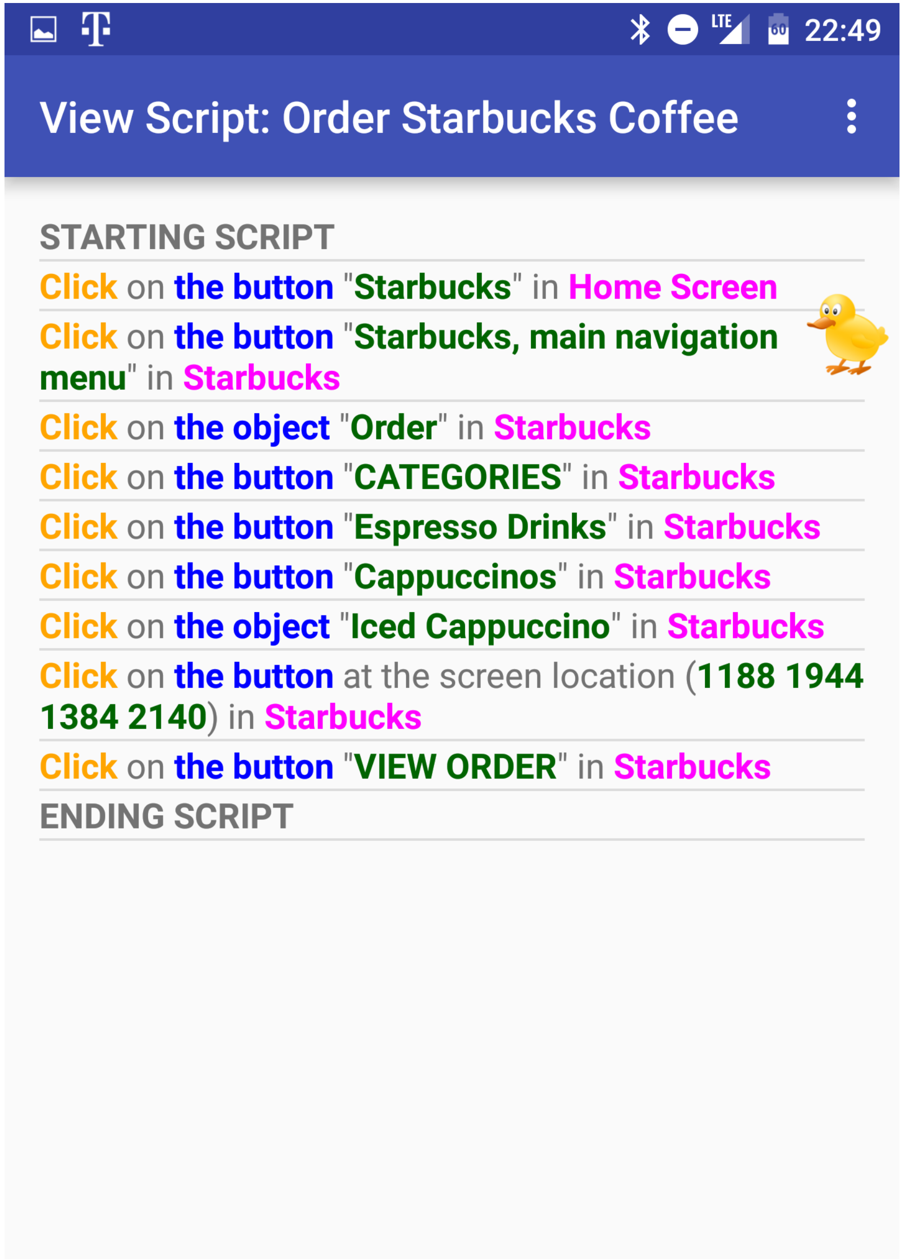
SUGILITE: Creating Multimodal Smartphone Automation by Demonstration

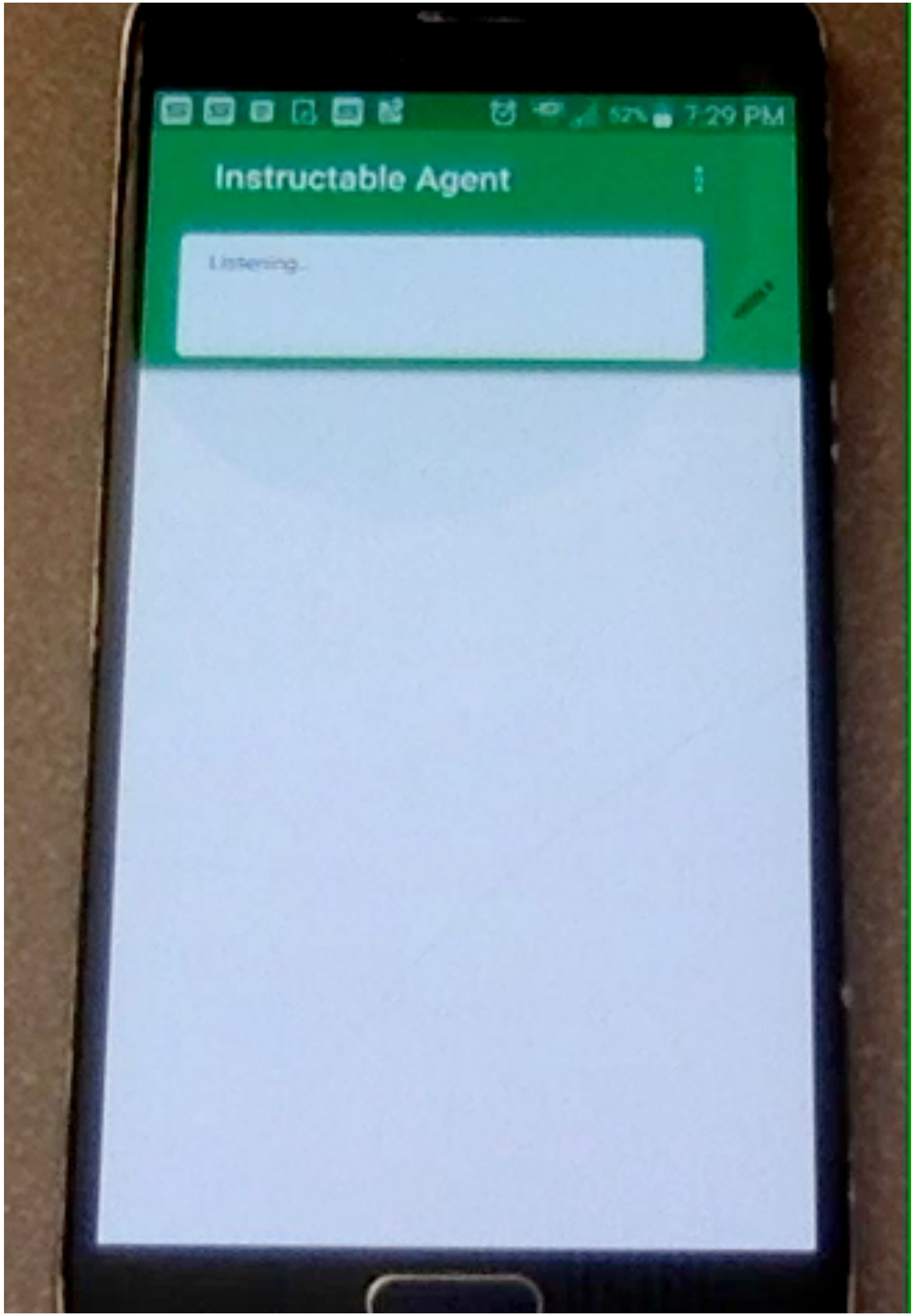
[1]Toby Jia-Jun Li, [2]Amos Azaria, [1]Brad A. Myers
[1]Human-Computer Interaction Institute, Carnegie Mellon University
[2]Computer Science Department, Ariel University

{tobyli, bam}@cs.cmu.edu, amos.azaria@ariel.ac.il

Learning by Demonstration (B. Meyers, T. Li)

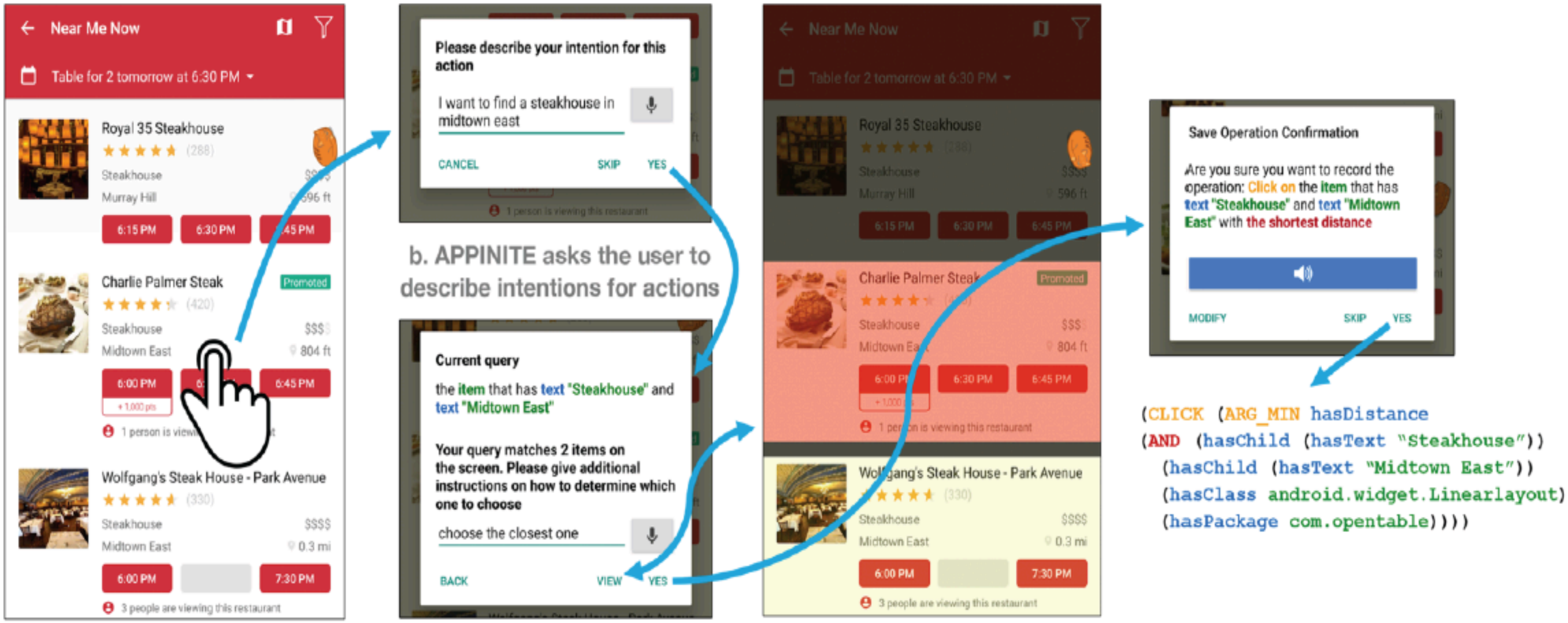






file:///Users/mitchell/Documents/My%20Documents/ppt/LIA_tellKatie_3min.mp4

Learning From Showing and Telling



a. User demonstrates the action directly on unmodified GUIs of third party apps

c. Multi-turn conversations help users refine ambiguous descriptions

d. User can view the result for the current query and the originally clicked UI object

e. APPINITE generates formal executable data descrption queries to be used in automation scripts

Fig. 1. Specifying data description in programming by demonstration using APPINITE: (a, b) enables users to naturally express their intentions for demonstrated actions verbally; (c) guides users to formulate data descriptions to uniquely identify target GUI objects; (d) shows users real-time updated results of current queries on an interaction overlay; and (e) formulates executable queries from natural language instructions.