Learning to control

D.A.Forsyth, UIUC

Road layout maps

- Potential cues
  - streetview
  - openmaps
  -

Partially supervised cues

- Open Street Maps (OSM)

Map data: OpenStreetMap is an open-source mapping project covering over 21 million miles of road. Unlike proprietary maps, the underlying road coordinates and metadata are freely available for download. Accuracy and overlap with Google Maps is very high, though some inevitable noise is present as information is contributed by individual volunteers or automatically extracted from users' GPS trajectories. For example, roads in smaller cities may lack detailed annotations (e.g., the number of lanes may be unmarked). These inconsistencies result in varying-sized subsets of the data being applicable for different attributes.

Seff+Xiao

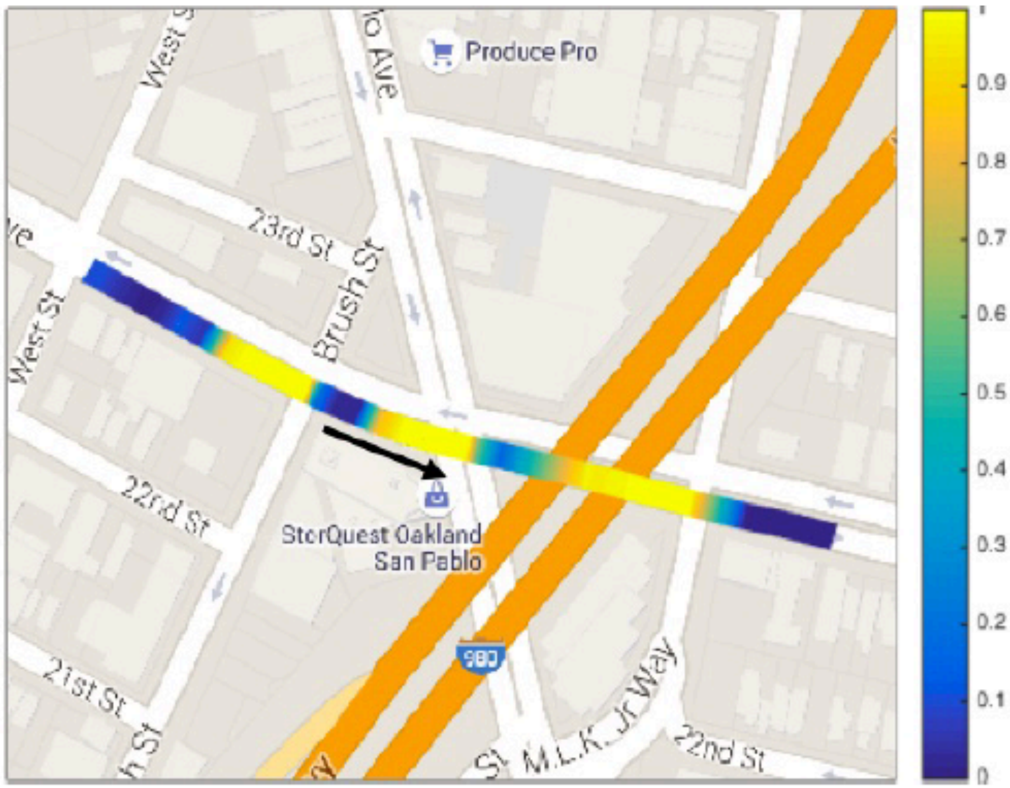


Fig. 3. Intersection detection heatmap. Images are cropped from test set GSV panoramas in the direction of travel indicated by the black arrow. The probabilities of "approaching" an intersection output by the trained ConvNet are overlaid on the road. (The images are from the ground level road, not the bridge.)

Seff+Xiao

Partially supervised cues

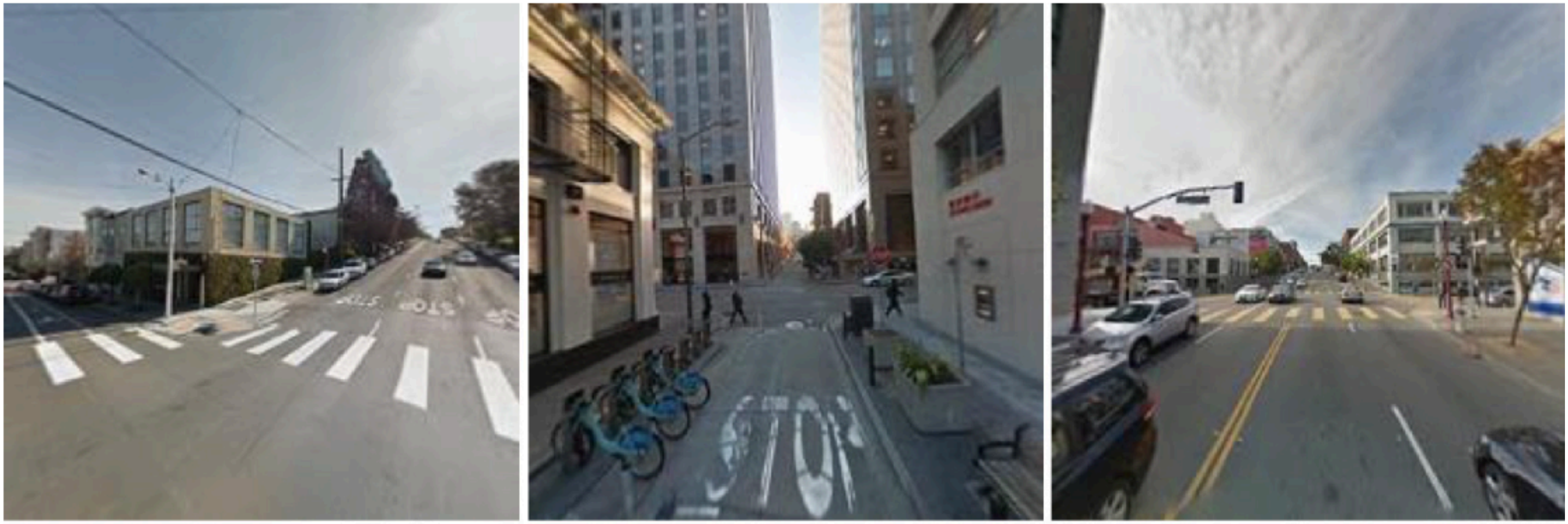
- Google street view

Image collection: Google Street View contains panoramic images of street scenes covering 5 million miles of road across 3,000 cities. Each panorama has a corresponding metadata file storing the panorama's unique "pano_id", geographic location, azimuth orientation, and the pano_ids of adjacent panoramas. Beginning from an initial seed panorama, we collect street view images by running a bread-first search, downloading each image and its associated metadata along the way. Thus far, our dataset contains one million GSV panoramas from the San Francisco Bay Area. GSV panoramas can be downloaded at several different resolutions (marked as "zoom levels"). Finding the higher zoom levels unnecessary for our purposes, we elected to download at a zoom level of 1, where each panorama has a size of 832×416 pixels.

Seff+Xiao

Labelling - I

- Match panoramas to roads
  - panorama center location, orientation is known
  - (essentially) project to plane
  - thresholded nearest neighbor to road center polyline
    - thresholding removes panoramas inside buildings, etc.
  - some noise
    - under bridges, etc.
- Annotations
  - Intersections
  - Drivable heading
  - Heading angle
  - Bike lane
  - Speed limit, wrong way, etc.



Pred = 0.1 m
True = 1.9 m

Pred = 18.5 m
True = 19.2 m

Pred = 22.9 m
True = 22.4 m

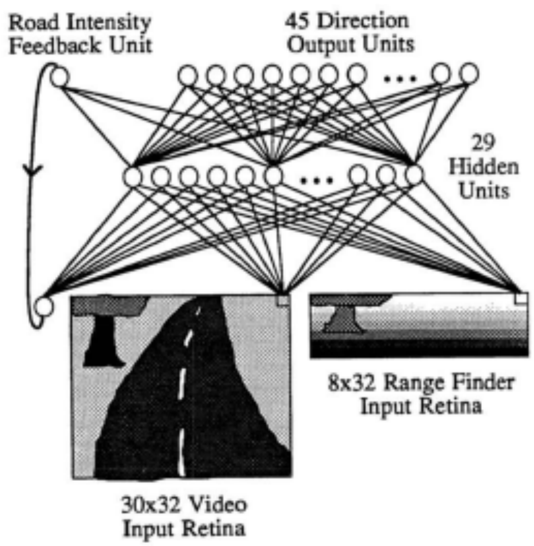
Fig. 4. Distance to intersection estimation. For images within 30 m of true intersections, our model is trained to estimate the distance from the host car to the center of the intersection across a variety of road types.

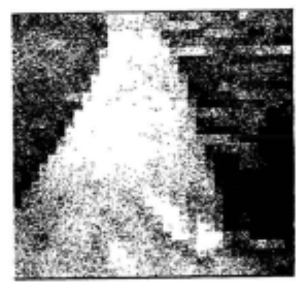
Seff+Xiao

BIG GOOD QUESTIONS

- Recall mashup of openmaps and street view
  - it could predict drivable directions, steering directions, lanes, signs, etc.
- Q: WHY IS THIS NOT DRIVING AROUND NOW?
  - A: (pretty obviously) because it doesn’t work
- Q: WHY NOT?
  - A: interesting

First learned steering controller





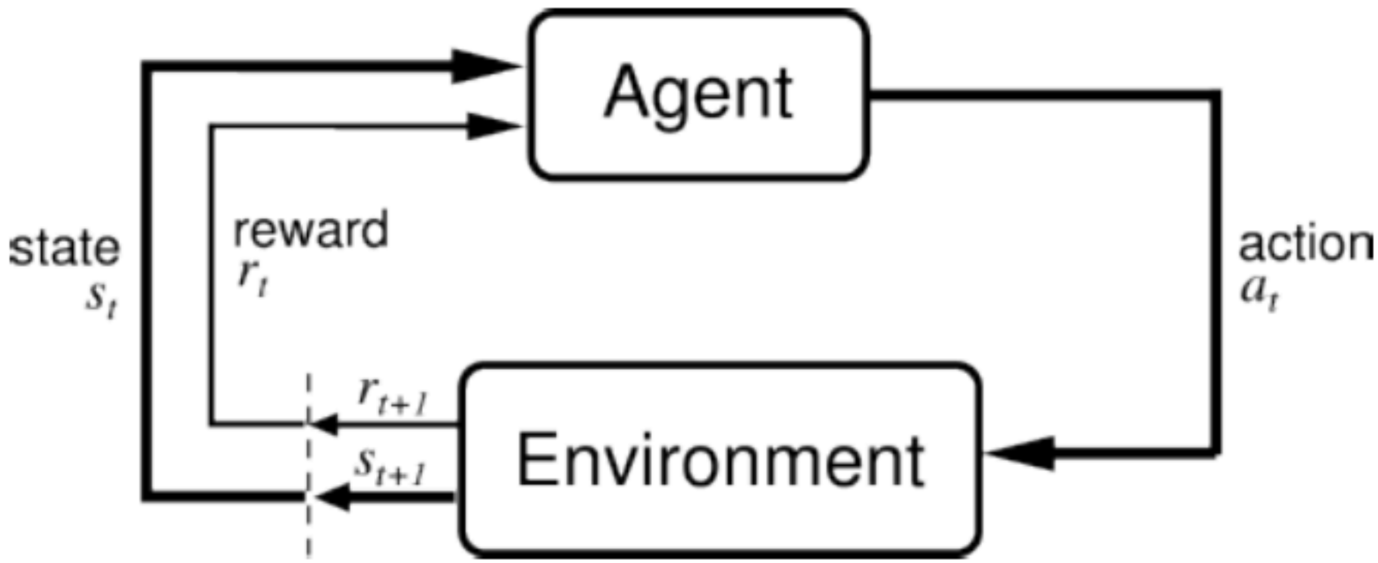
Real Road Image

" ALVINN: An autonomous Land vehicle in a neural Network, Pomerleau 1989

Topics

- Vocabulary
- Simplest imitation learning and DAGGER
  - to set up possible projects, and answer Q1, Q2
- Simple reinforcement learning ideas
- More imitation learning; inverse reinforcement learning
  - and its variants and problems

Markov Decision Process



Assumption: agent gets to observe the state

[Drawing from Sutton and Barto, Reinforcement Learning: An Introduction, 1998]

Abbeel slides

Model

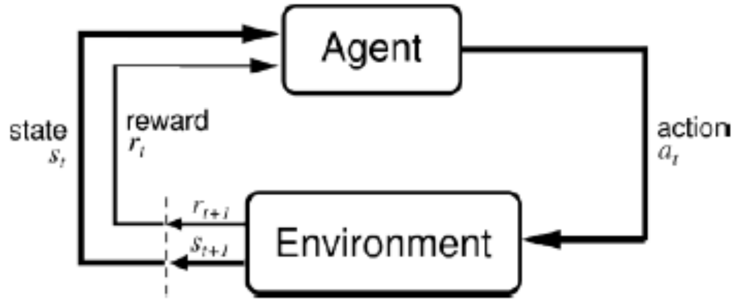
- At time 0, environment samples initial state
  - agent is in that state
- Then for t=0 till done
  - agent chooses action
  - environment samples new state conditioned on action, old state
  - environment samples reward conditioned on action, old state, new state
  - agent gets that reward and moves into new state
- Policy
  - what action to take in each state
    - this could be stochastic
- Maximise total discounted reward

Examples

- Cleaning robot
- Walking robot
- Pole balancing
- Games: tetris, backgammon
- Server management
- Shortest path problems
- Model for animals, people

Abbeel slides

Markov Decision Process (S, A, T, R, H)



Given

- S: set of states
- A: set of actions
- T: S x A x S x {0,1,…,H} → [0,1], $T_t(s,a,s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$
- R: S x A x S x {0, 1, …, H} → $\Re$ $R_t(s,a,s')$ = reward for ($s_{t+1} = s'$, $s_t = s$, $a_t = a$)
- H: horizon over which the agent will act

Goal:

- Find π : S x {0, 1, …, H} → A that maximizes expected sum of rewards, i.e.,

$$\pi^* = \arg\max_{\pi} \mathrm{E}[\sum_{t=0}^{H} R_t(S_t, A_t, S_{t+1}) | \pi]$$

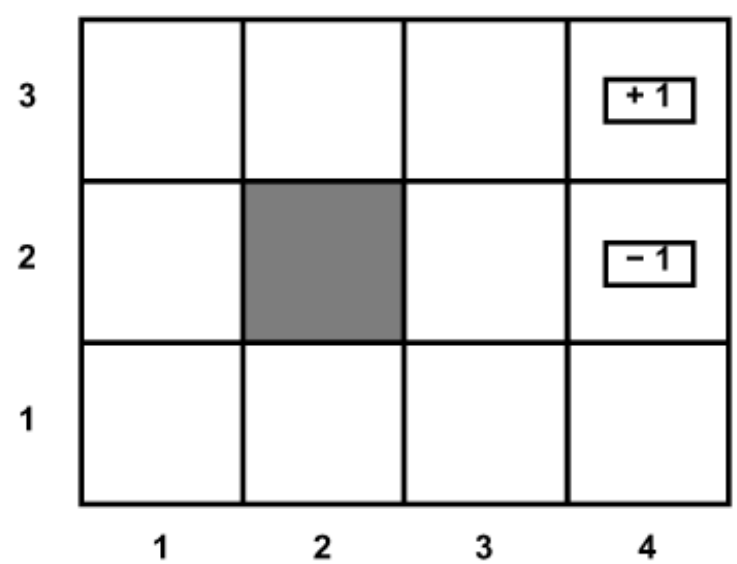
This is usually discounted by gamma

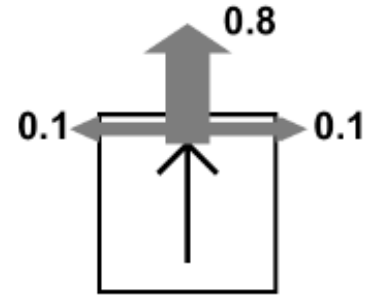
Abbeel slides

Canonical Example: Grid World

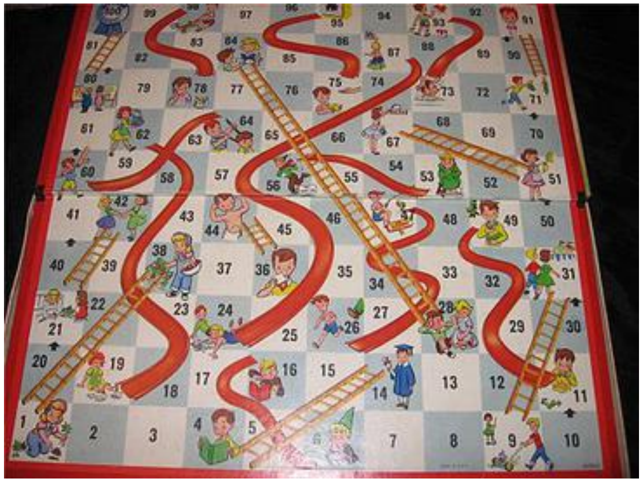
- The agent lives in a grid
- Walls block the agent's path
- The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- Big rewards come at the end

And this is true for the other three; 80% of the time you go where you intended, 10% at right angles one way 10% the other



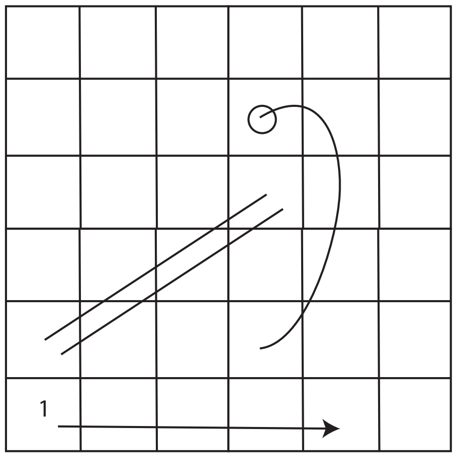


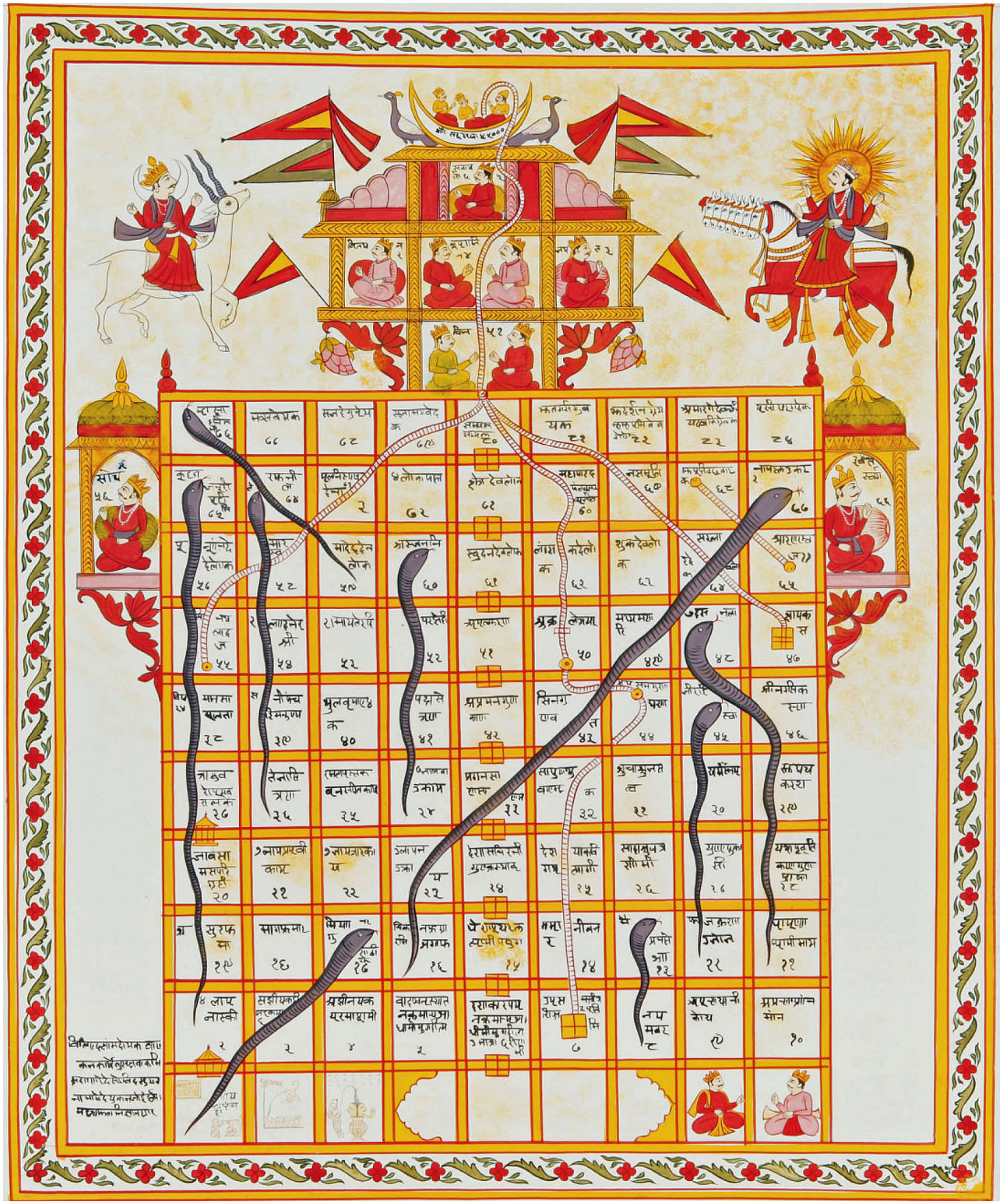
Snakes + Ladders



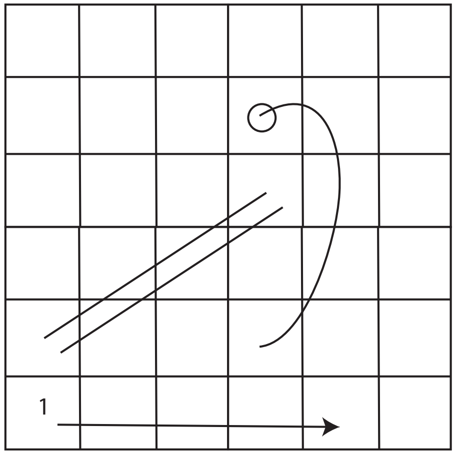
Lift from Wikipedia entry

- Sometimes, chutes and ladders
  - There is a board (typically, 10x10 grid) with numbered cells
  - Two players (for us - more OK)
    - both start at 1
  - In turn, each
    - throws a die
    - moves forward the given number of cells
    - if final cell is base of ladder, goes up that ladder
    - if final cell is head of snake, goes down that snake
  - winner is first to leave board
- This is an MDP
  - but there's no choice of action, so it's really a Markov Chain





Lift from Wikipedia entry



S+L as MDP

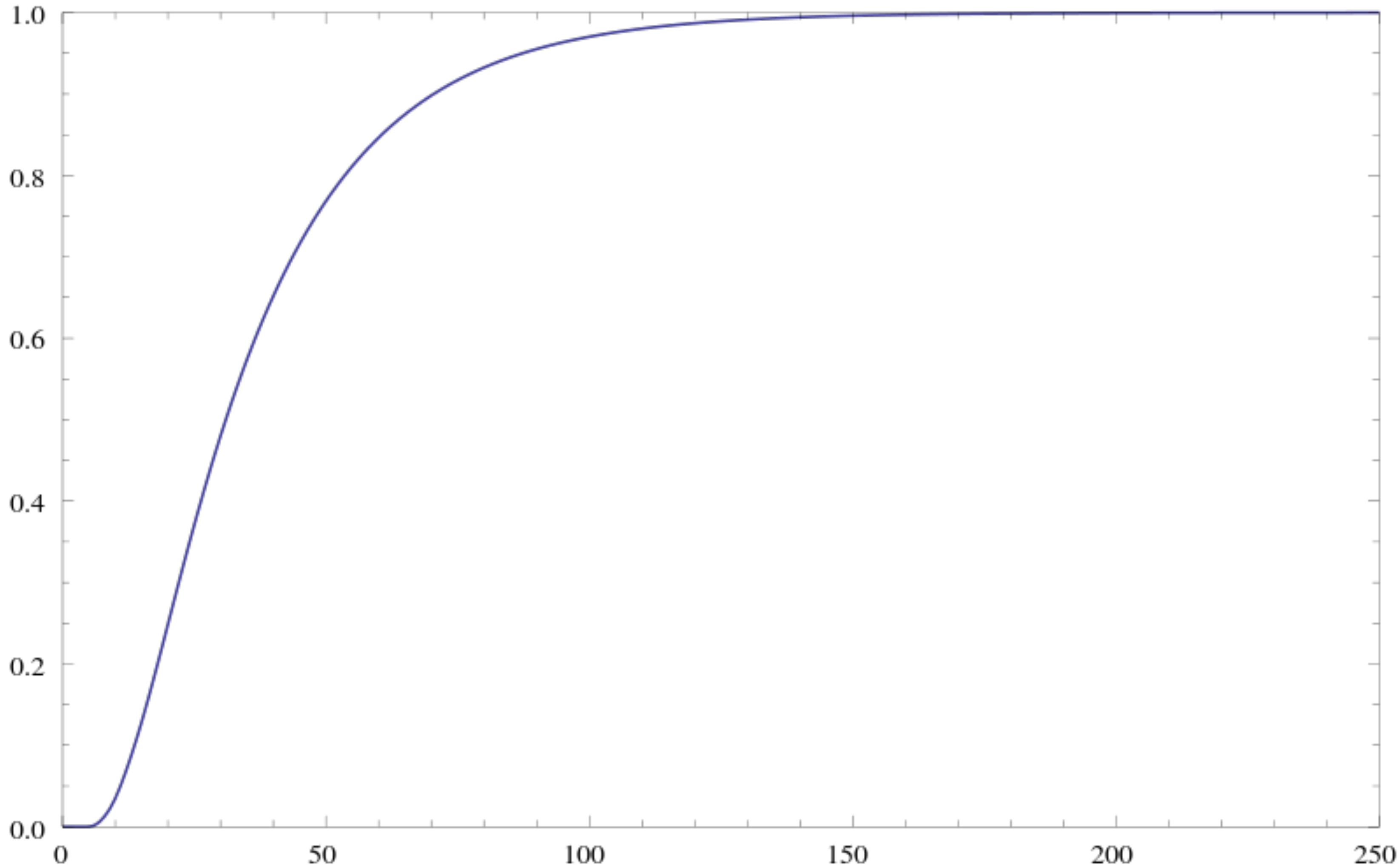
- States:
  - 10 x 10 x 2 = (position of p1, position of p2, who moves next)
  - transitions
    - for each state, six possible new states
      - big table will do it
      - P(new|old)=1/6
- Q:
  - who wins?
  - how long does game go on?
  - what is the value of a particular position

Math

Any version of snakes and ladders can be represented exactly as an absorbing Markov chain, since from any square the odds of moving to any other square are fixed and independent of any previous game history.[24] The Milton Bradley version of *Chutes and Ladders* has 100 squares, with 19 chutes and ladders. A player will need an average of 39.2 spins to move from the starting point, which is off the board, to square 100. A two-player game is expected to end in 47.76 moves with a 50.9% chance of winning for the first player.[25] These calculations are based on a variant where throwing a six does not lead to an additional roll; and where the player must roll the exact number to reach square 100 and if they overshoot it their counter does not move.

Lift from Wikipedia entry

Cumulative probability of finishing by (ie. at or before) N'th round of S+L



Lift from Wikipedia entry

Questions

- Known environment, rewards
  - Assume
    - we know T(s, a, s'), R(s, a, s')
  - What should our policy be?
    - do math

Solving MDPs

- Unknown environment, rewards
  - What should our policy be?
    - act and adjust policy to improve rewards

Reinforcement learning

- Unknown environment, rewards, but access to expert
  - What should our policy be?
    - (a1) do what the expert does
    - (a2) figure out the experts reward function, and maximize that

Imitation learning

Inverse reinforcement learning

Reinforcement Learning: Learning policies guided by sparse rewards, e.g., win or not the game.

- Good: simplest, cheapest 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:
    1. action execution takes long
    2. we cannot afford to fail
    3. safety concerns

Crusher robot

Fragkiadaki, ND

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"*

Fragkiadaki, ND

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 is easier for the expert to demonstrate the desired behavior rather than:

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

b) coding up the desired policy directly.

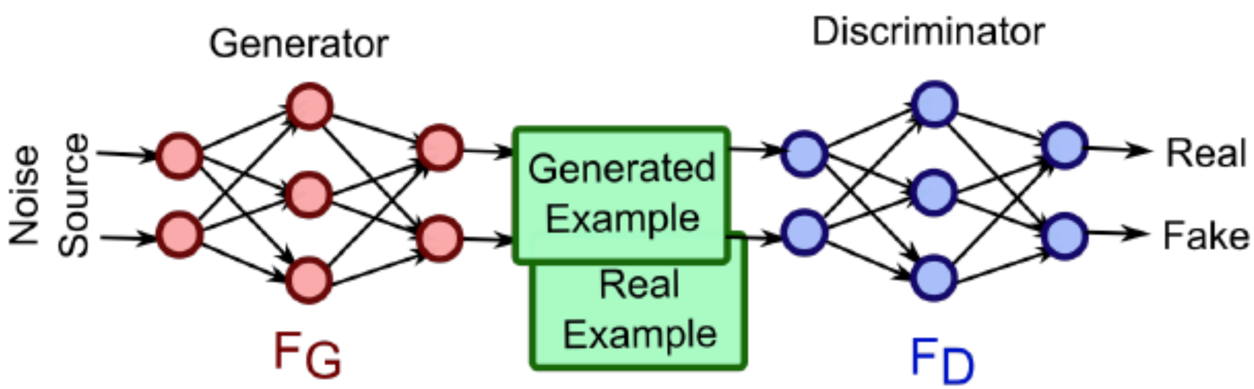
Fragkiadaki, ND

The Imitation Learning problem

The agent (learner) needs to come up with a policy whose resulting state, action trajectory distribution matches the expert trajectory distribution.

Does this remind us of something…?

GANs! Generative Adversarial Networks (on state-action trajectories)



Fragkiadaki, ND

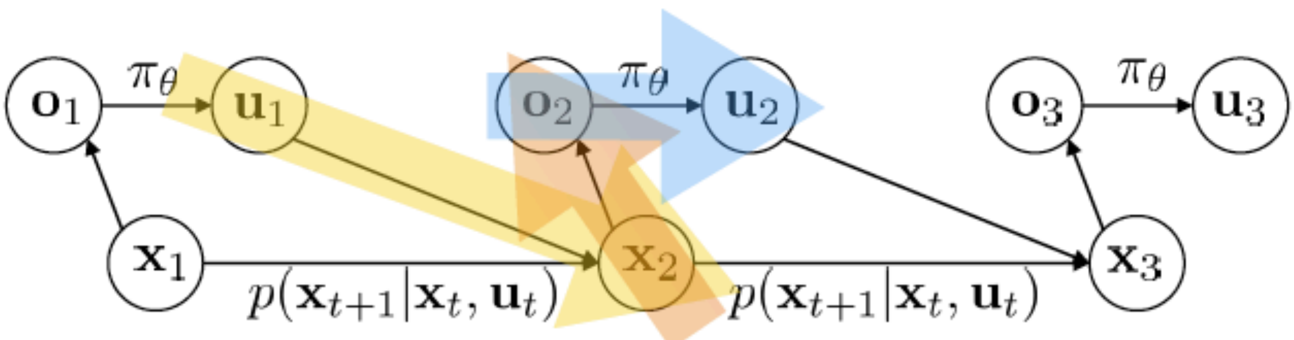
Generative Adversarial Networks, Goodfellow et al. 2014

The Imitation Learning problem: Challenge

Actions along the trajectories are interdependent, as actions determine state transitions and thus states and actions down the road.

interdependent labels -> structure prediction

Action interdependence in time:



Algorithms developed in Robotics for imitation learning found applications in structured predictions problems, such as, sequence generation/labelling e.g. parsing.

Imitation Learning

For taking this structure into account, numerous formulations have been proposed:

- Direct: Supervised learning for policy (mapping states to actions) using the demonstration trajectories as ground-truth(a.k.a. behavior cloning) + ways to handle the neglect of action interdependence.

- Indirect: Learning the latent rewards/goals of the teacher and planning under those rewards to get the policy, a.k.a. Inverse Reinforcement Learning (next lecture)

Experts can be:

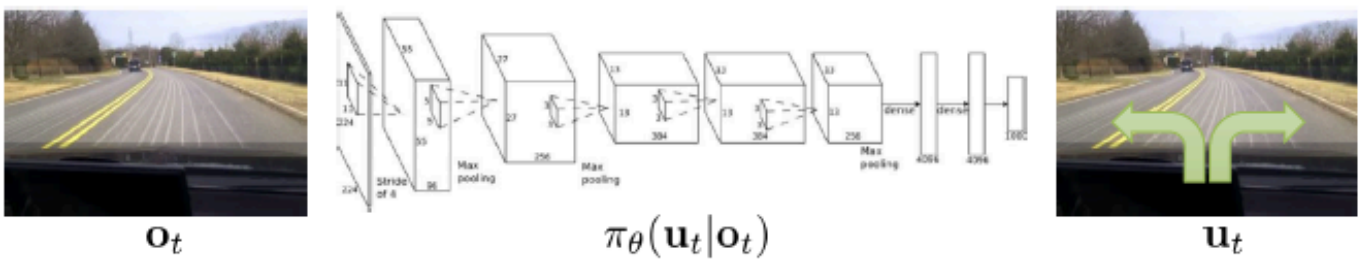
- Humans

- Optimal or near Optimal Planners/Controllers

Fragkiadaki, ND

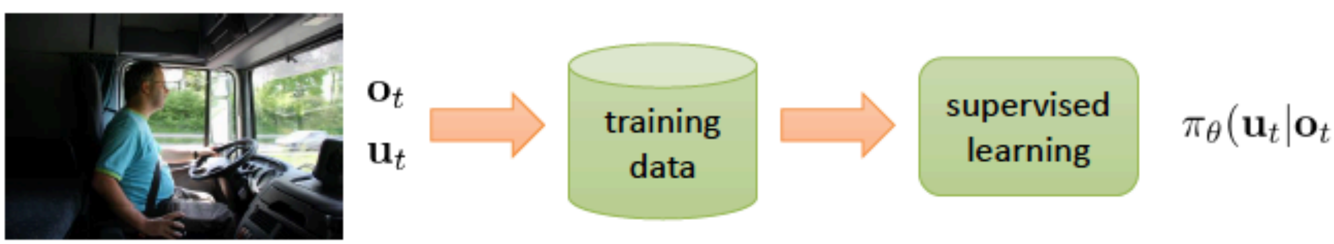
Imitation Learning as Supervised Learning

Driving policy: a mapping from (history of) observations to steering wheel angles



Behavior Cloning=Imitation Learning as Supervised learning

- Assume actions in the expert trajectories are i.i.d.
- Train a classifier or regressor to map observations to actions at each time step of the trajectory.



Fragkiadaki, ND

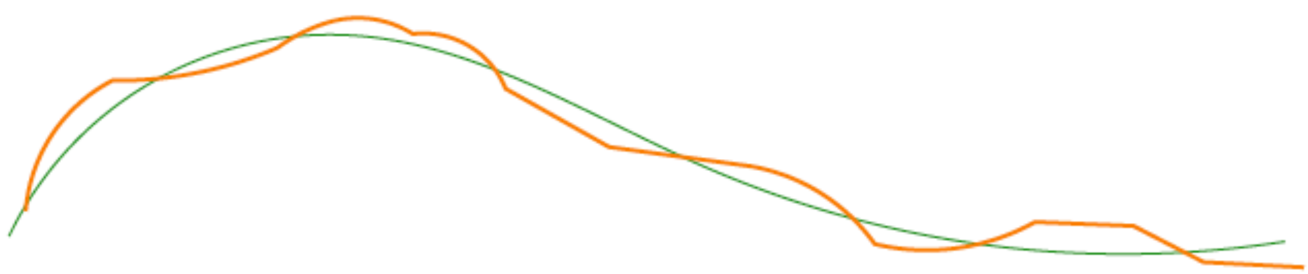
End to End Learning for Self-Driving Cars, Bojarski et al. 2016

Classifier or regressor?

Because multiple actions u may be plausible at any given observation o, policy network $p_{\pi_\theta}(u_t|o_t)$ usually is not a regressor but rather:

- A classifier (e.g., softmax output and cross-entropy loss, after discretizing the action space)
- $J(\theta) = -\sum_{i=1}^{m}\sum_{k=1}^{K} 1_{y(i)=k} \log[P(y_{(i)} = k|x_{(i)};\theta)]$
- A GMM (mixture components weights, means and variances are parametrized at the output of a neural net, minimize GMM loss, (e.g., Hand writing generation Graves 2013)
- A stochastic network (previous lecture)

Independent in time errors

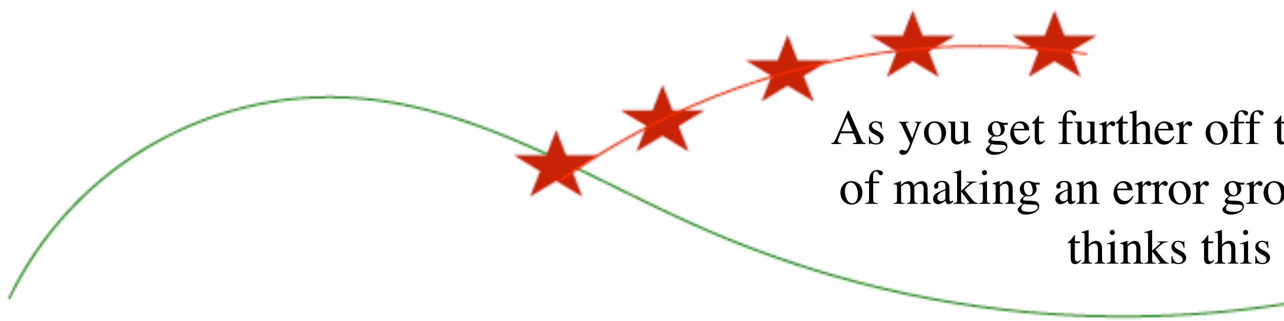


error at time t with probability ε

E[Total errors] ≲ εT

Fragkiadaki, ND

Compounding Errors

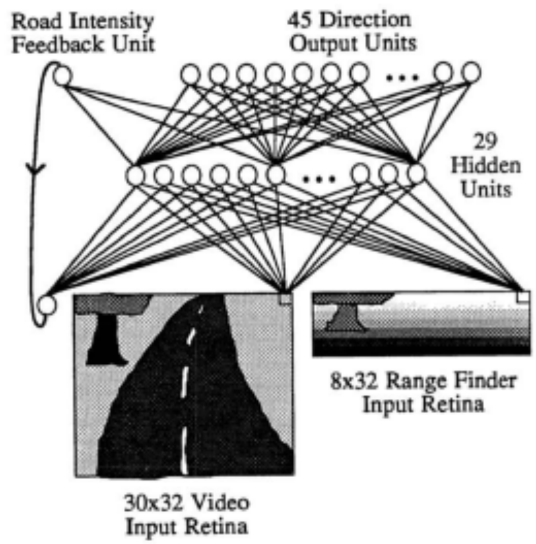


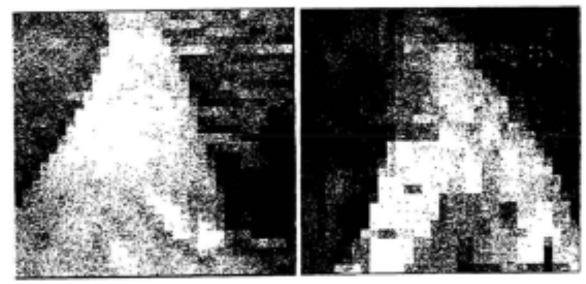
As you get further off the path, the probability of making an error grows, cause the classifier thinks this state is rare

error at time t with probability ε

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

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



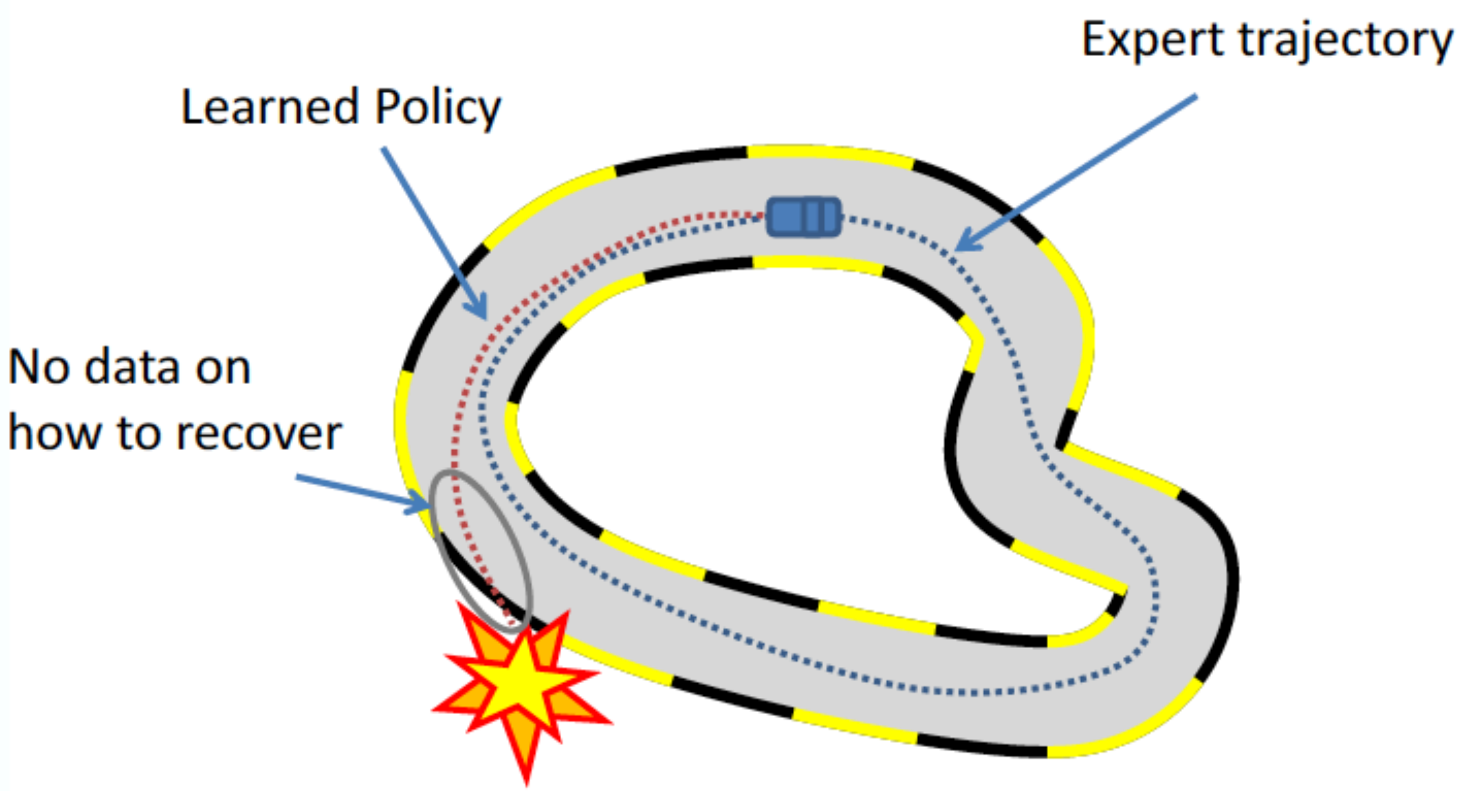


Real Road Image   Simulated Road Image

"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!

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



Fragkiadaki, ND

Data Distribution Mismatch!

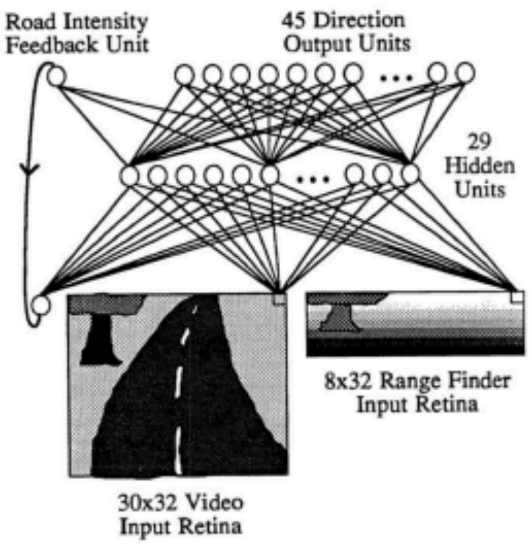
	supervised learning	supervised learning + control (NAIVE)
train	(x,y) ~ D	$s \sim d_{\pi^*}$
test	(x,y) ~ D	$s \sim d_{\pi}$

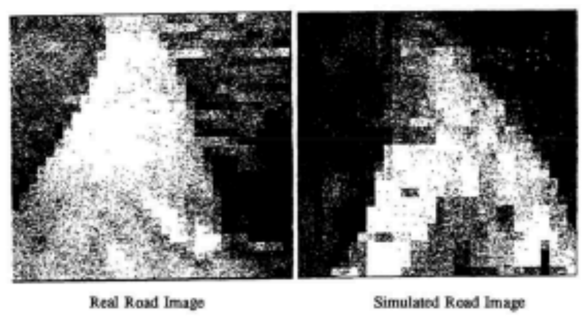
SL succeeds when training and test data distributions match, that is a fundamental assumption.

Fragkiadaki, ND

Demonstration Augmentation: ALVINN 1989

Road follower

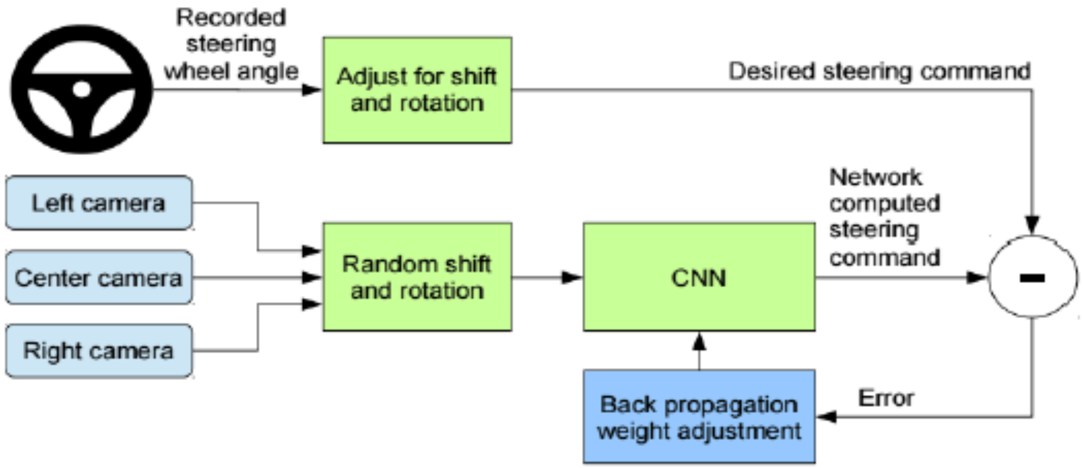


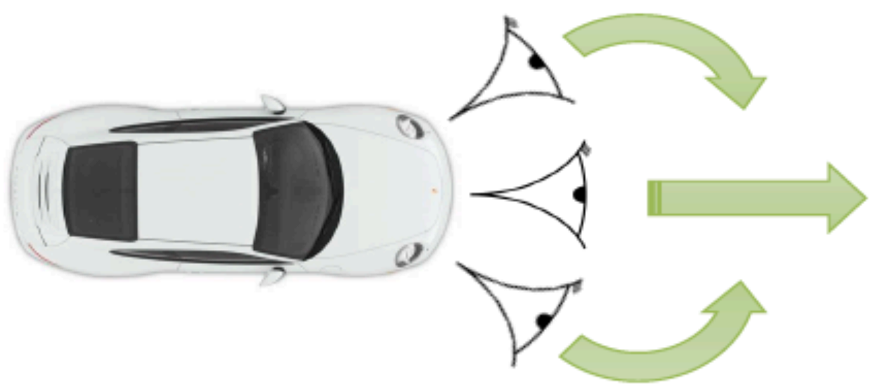


- Using graphics simulator for road images and corresponding steering angle ground-truth
- Online adaptation to human driver steering angle control
- 3 layers, fully connected layers, very low resolution input from camera and lidar..

"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

Demonstration Augmentation: NVIDIA 2016



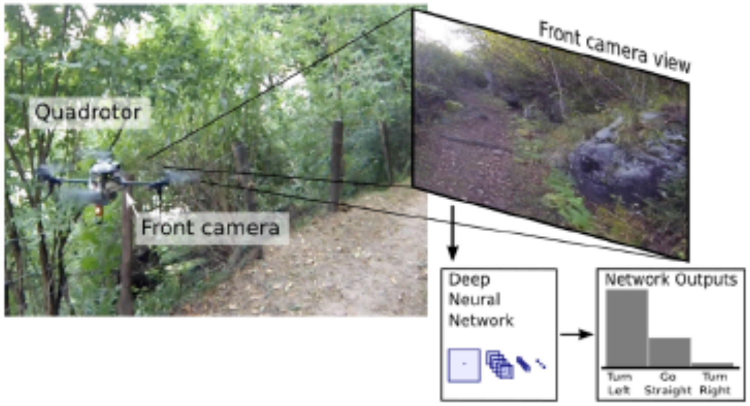


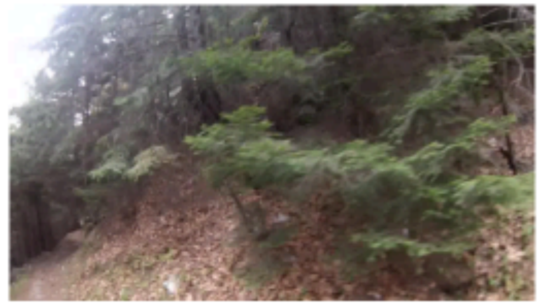
Additional, left and right cameras with automatic grant-truth labels to recover from mistakes

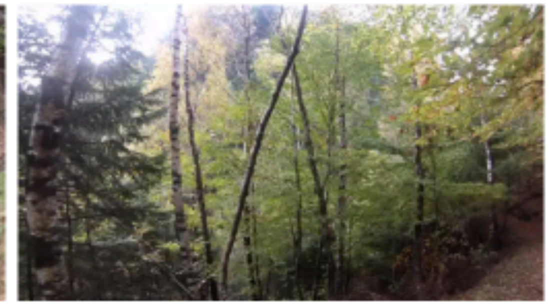
"*DAVE-2 was inspired by the pioneering work of Pomerleau [6] who in 1989 built the Autonomous Land Vehicle in a Neural Network (ALVINN) system*. Training with data from only the human driver is not sufficient. The network must learn how to recover from mistakes. …",

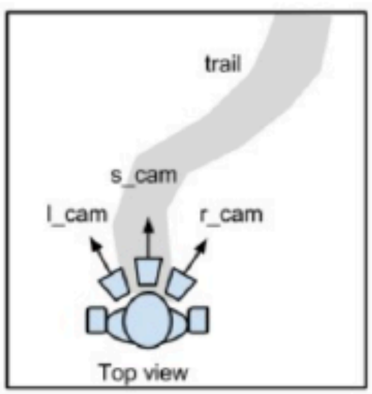
End to End Learning for Self-Driving Cars , Bojarski et al. 2016

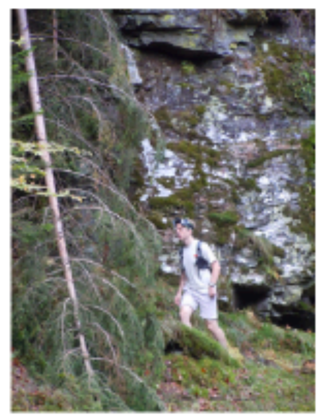
Data Augmentation (3): Trails 2015









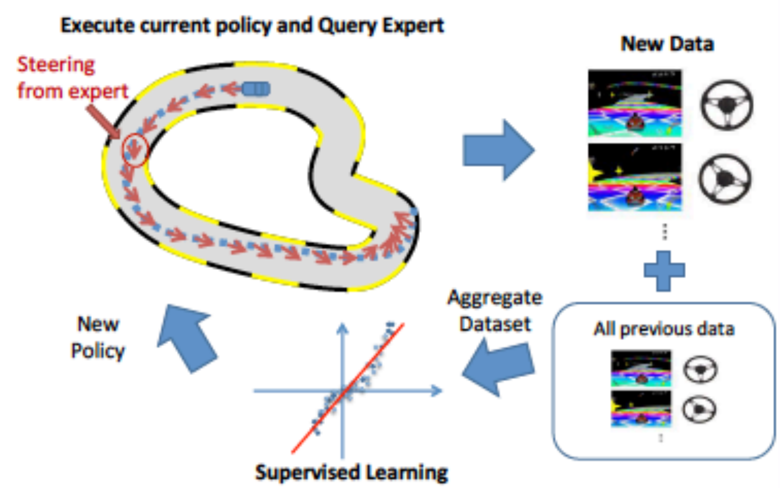


A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots Giusti et al.

DAGGER (in simulation)

Dataset AGGregation: bring learner's and expert's trajectory distributions closer by labelling additional data points resulting from applying 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. $\boxed{\text{run } \pi_\theta(u_t|o_t) \text{ to get dataset } \mathcal{D}_\pi = \{o_1, ..., o_M\}}$
3. $\boxed{\text{Ask human to label } \mathcal{D}_\pi \text{ with actions } u_t}$
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

Fragkiadaki, ND

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

DAGGER (in simulation)

Dataset AGGregation: bring learner's and expert's trajectory distributions closer by labelling additional data points resulting from applying 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

Fragkiadaki, ND

A Reduction of Imitation Learning and Structured Predictio