Decision Making
in Robots and Autonomous Agents

Special Topics: Safety and Security

Subramanian Ramamoorthy
School of Informatics

29 March 2019

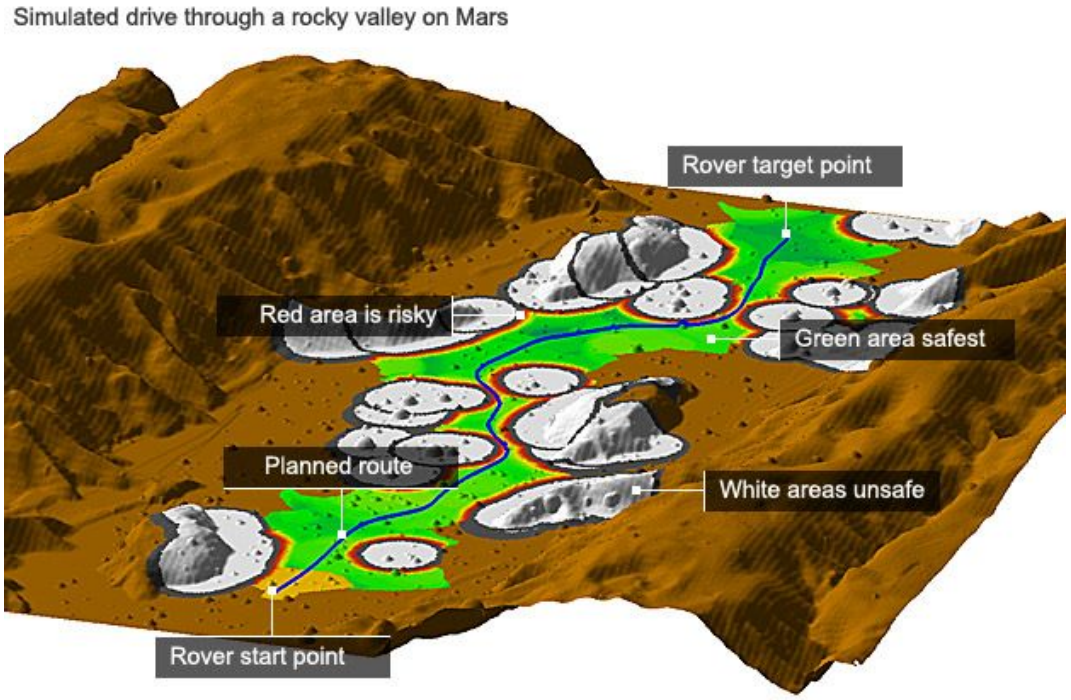
Plan

1. Ensuring safety in policies
   - concept of adversarial attacks

2. Differential privacy and relation to estimation

SAFE POLICIES

Recap from DP: How to go from "A to B"



$$x_n^* = \arg\min f_n(s, x_n)$$

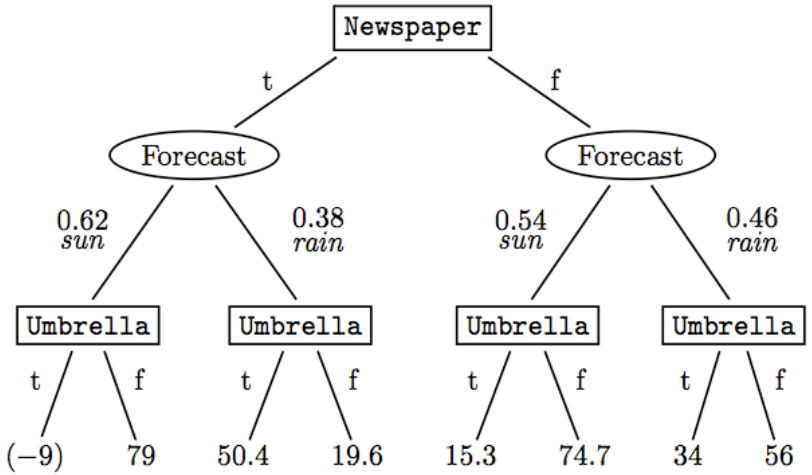
$$f_n^*(s) = \text{minimum value of } f_n(s, x_n)$$

$$f_n^*(s) = f_n(s, x_n^*)$$

Recap from Decision Theoretic and Bayesian Methods

- Action selection can be based on maximizing expected value in a sequential (and possibly interactive) tree structured choice problem

- In worked example, you saw different variations of this approach

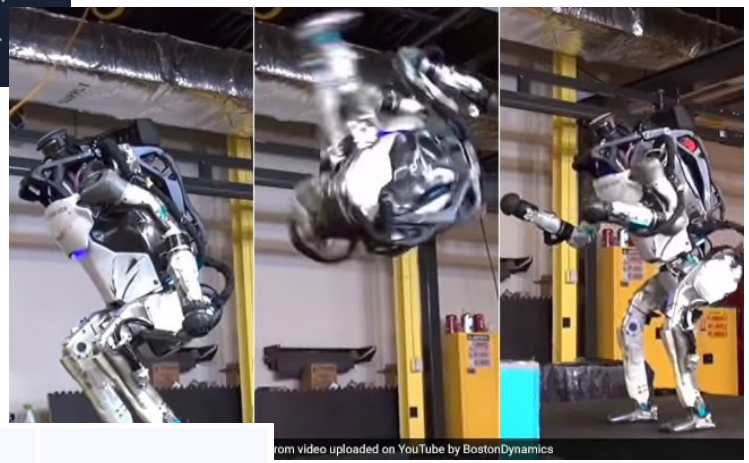


Other Modern Methods

- The basic dynamic programming recursions and the concept of the value function is developed into a variety of approximation methods within reinforcement learning
- Modern Deep Reinforcement Learning methods represent significant computational advances (e.g., use a neural network to efficiently approximate the f for very large state spaces in a Deep Q network, DQN)
- Likewise Monte Carlo Tree Search approaches represent sampling based extensions to the basic decision tree, which can scale up to very large search problems

Tension Between Two World Views: Methods like DQN/MCTS have only been shown to work in some applications

[Source: Wired]

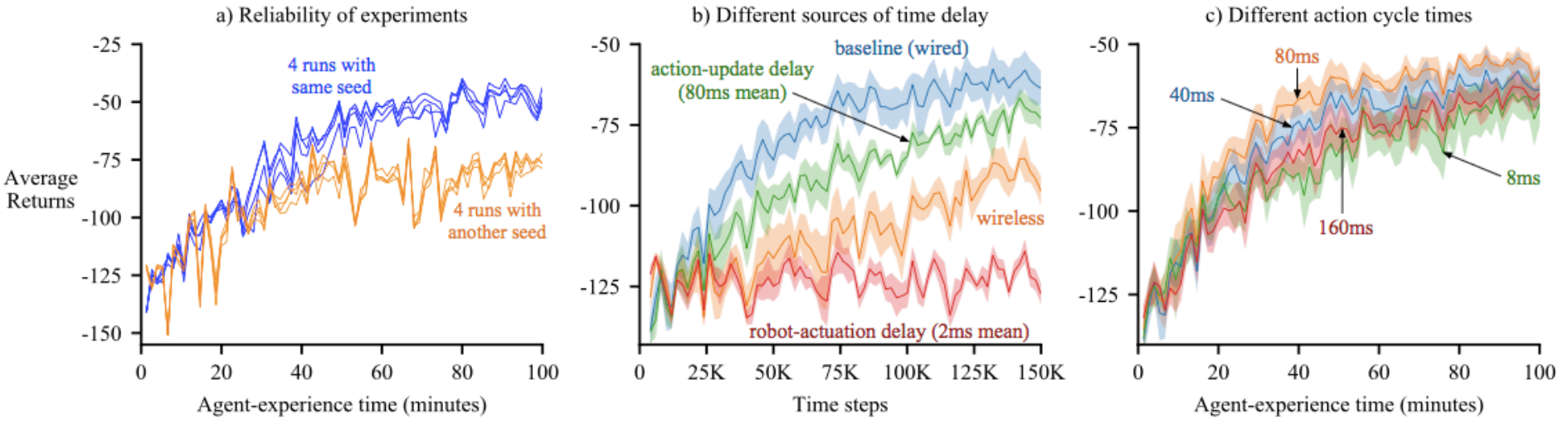


[Source: NDTV]

Complex robot behaviours have been hard to encode, need control theory

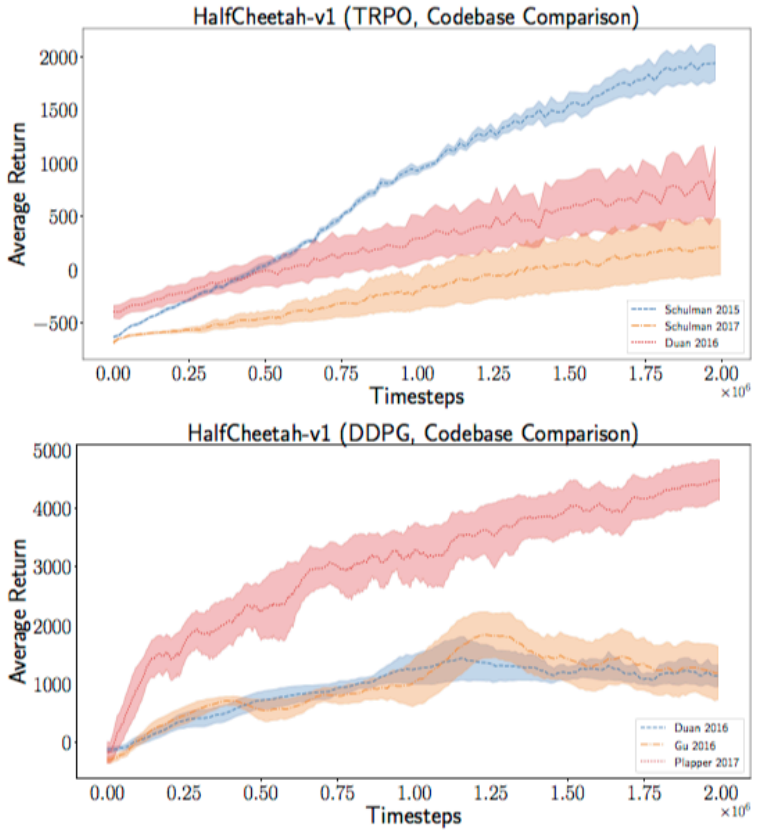
[Source: DeepMimic, https://arxiv.org/pdf/1804.02717.pdf]

How Robust is the Robot Behaviour? Performance of RL on Robot is VERY Sensitive to Numerous Parameter Choices



[Mahmood et al. IROS 2018, Setting up a Reinforcement Learning Task with a Real world Robot https://arxiv.org/pdf/1803.07067.pdf]

How Robust is Policy Learning?
Not Very: Details in Code can have Big Impact



[Henderson et al. 2017, Deep Reinforcement Learning that Matters, https://arxiv.org/abs/1709.06560]

Can we deploy these policies in safety-critical applications?

[Source: Justin Kovalsky, Johns Hopkins Clinical Connections]

[Source: Dan Boman, Scania CV AB]

Accidents with ML Systems

- Accident: situation where a human designer had in mind a certain (perhaps informally specified) objective or task, but the system that was designed and deployed for that task produced harmful and unexpected results
- How do things go wrong in AI systems?
  1. Designer can get the objective function wrong
     - Negative side effects
     - Reward/cost hacking
  2. Distributional shifts
  3. Too expensive to get input (e.g., ask a human)

Using Logical Constraints in Policy Synthesis

System Dynamics

$$x(t_{k+1}) = f_d(x(t_k), u(t_k), w(t_k))$$

state, control, noise

Run with a horizon N:

$$\xi(x_0, \mathbf{u}^N, \mathbf{w}^N) = (x_0 u_0 w_0)(x_1 u_1 w_1)(x_2 u_2 w_2)...(x_N u_N w_N),$$

What is a Logic?
Example: Signal Temporal Logic

(During) (Until)

$$\varphi ::= \pi^{\mu} \mid \neg\psi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond_{[a,b]}\ \varphi \mid \varphi_1\ \mathcal{U}_{[a,b]}\ \varphi_2,$$

Can further derive **always** ($\Box$), **unless**, **next** etc.

The μ is a signal function, allows us to turn real value into boolean:

$$\mu : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \to \mathbb{R}$$

The formula is true if sign is positive ($\boldsymbol{\pi}^{\mu}$). Semantics given on run

Controller Synthesis with Logical Constraints

Problem 1 (open-loop). Compute $\mathbf{u}^* = u_0^* u_1^* \dots u_{N-1}^*$ where

$$\begin{aligned} \mathbf{u}^* = \quad & \underset{\mathbf{u} \in \mathcal{U}^N}{\operatorname{argmin}} \, J(x_0, \mathbf{u}, \mathbf{w}, \varphi) \\ & \text{s.t. } \xi(x_0, \mathbf{u}, \mathbf{w}) \models \varphi \end{aligned}$$

SOLUTION: Use the quantitative semantics to translate into a SAT problem. We will omit the details for the purpose of this lecture. For closed loop, iterate and expect updated environment in each iteration.

WHAT DO WE GET: A receding horizon control sequence. Repeat every step. **Proved** to satisfy specification.

Experiment: Adaptive Cruise Control

$$x(t_{k+1}) = f_d(x(t_k), u(t_k), w(t_k))$$

$x = (p_e, v_e, p_a, v_a, sl)$

State includes longitudinal **p**osition and **v**elocity of ego and adversary as well as speed limit appropriate for ego.

$u = (a_e)$

Controller input is **a**cceleration of ego

$w = (a_a, sl)$

Environment (**w**) is **a**cceleration of adversary and **s**peed **l**imit at next time. Both from prediction and localization

System evolution is straightforward: updated distance is function of velocity, updated velocity is function of accel, speed limit is set as environmental prediction

Experiment: Adaptive Cruise Control

Problem: synthesise a controller that will look-ahead and plan a series of inputs satisfying formal conditions:

$\square_{[0,\infty]}\ p_a(t) - p_e(t) > \delta_{min}$ #Ego keeps minimum distance from lead car

$\square_{[0,\infty]}\ |v_a(t) - v_e(t)| < 0.1 \,.\, v_a(t) \ \vee\ v_a(t) - sl(t) > 0$ #Ego drives within 10% of lead car unless lead car breaks the limit

$\square_{[0,\infty]}\ sl(t) - v_e(t) \geq 0$ #Ego car doesn't break speed limit

Experiment: Adaptive Cruise Control

Position Ego

Ego Velocity

Position Adv

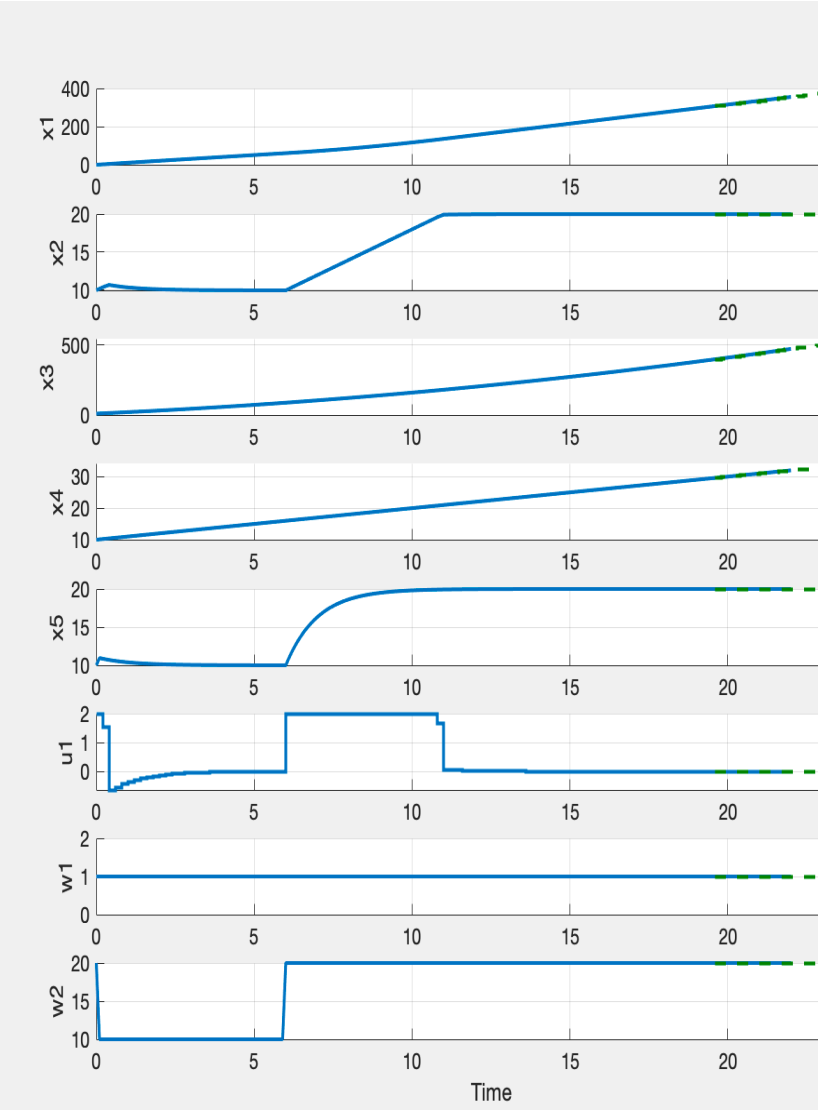
Velocity Adv

Speed Limit

Accel Ego

Accel Adv

Speed Limit Env



Interpretation:
ego *car* matches the speed of the car in front until it starts breaking the speed limit, then the ego car halts at the limit.

Experiment: One-way traffic, navigating parked cars

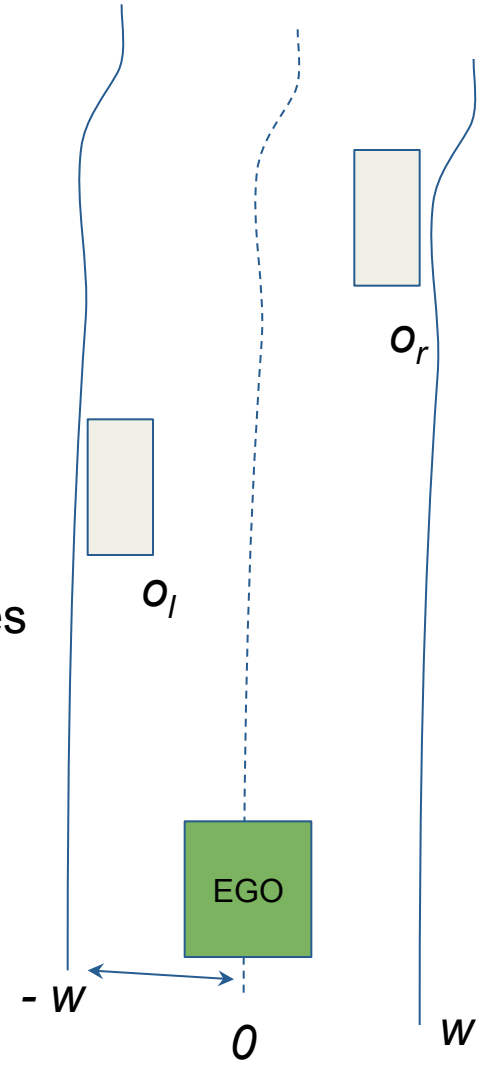
$$x(t_{k+1}) = f_d(x(t_k), u(t_k), w(t_k))$$

$x = (\ldots, l_e, w, o_l, o_r)$ State extended with **l**ateral position of ego, **w**idth of the road, and extent of **o**bstacles on **l**eft and **r**ight

$u = (\ldots, \delta_e)$ Controller input extended with delta in lateral position

$w = (\ldots, w, o_l, o_l)$ Environment (**w**) extended to have knowledge of the future width, obstacles

System evolution is as before. Cost function penalises unnecessary movement from center



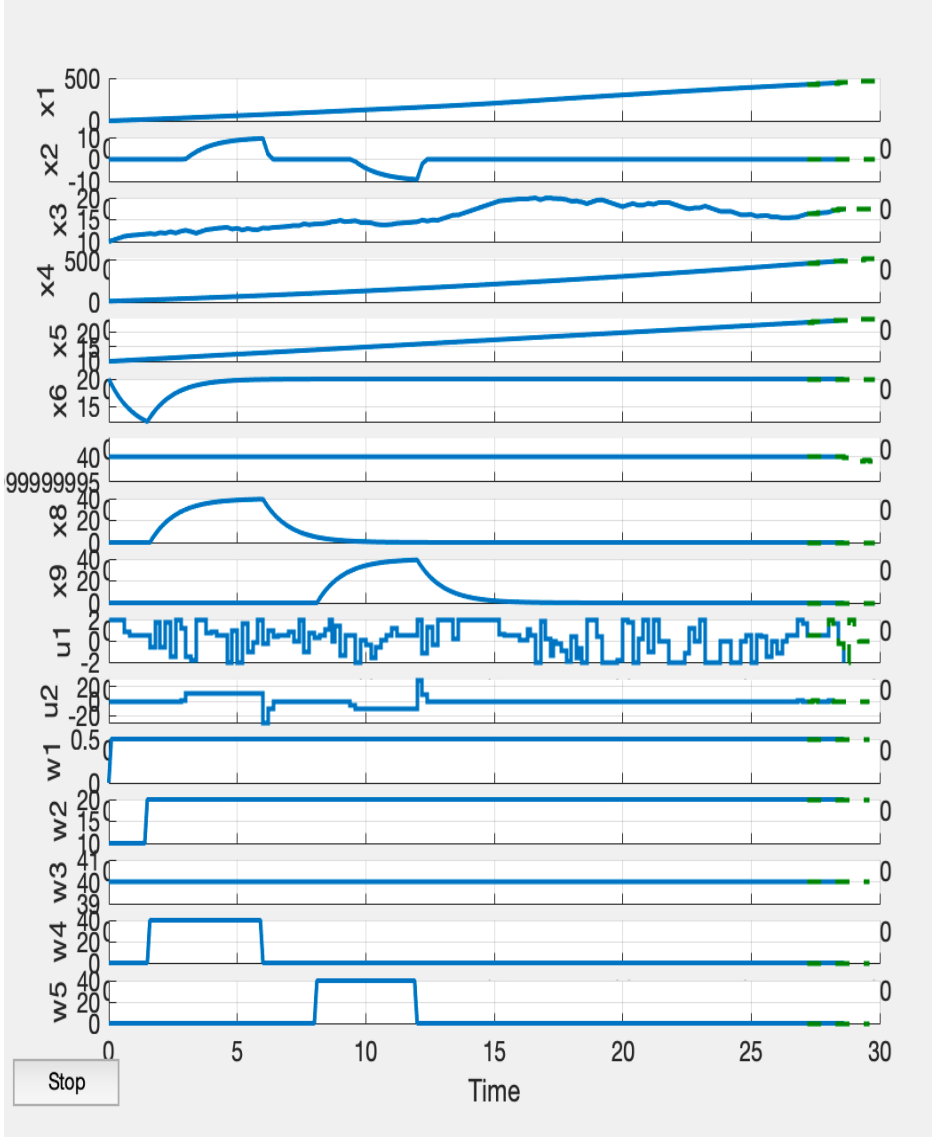
Experiment: One-way traffic, navigating parked cars

Problem: synthesise a controller that will look-ahead and plan a series of inputs satisfying formal conditions:

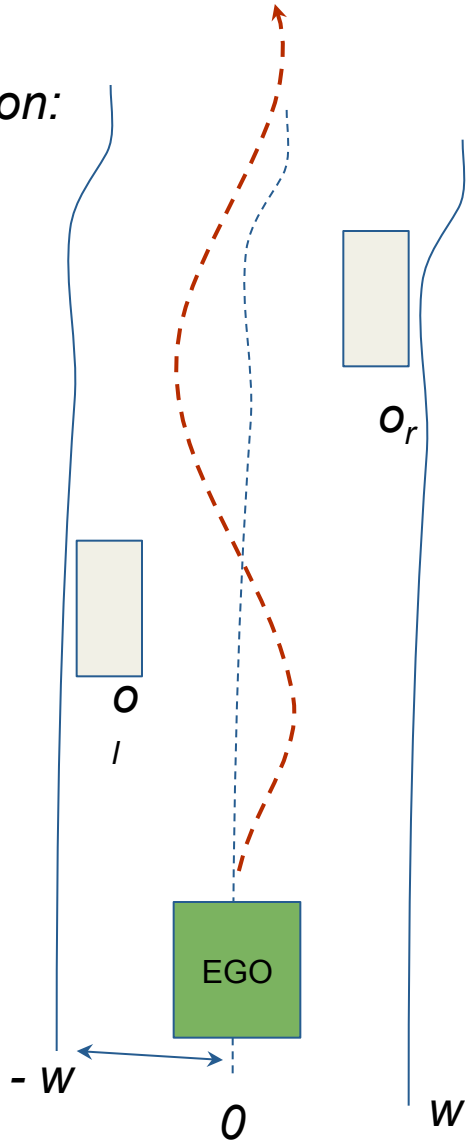
$\square_{[0,\infty]}\ l_e(t) < w(t) - o_r(t) - mc$ #Ego keeps at least the **m**inimum **c**learance from obstacles on the right of the single lane road

$\square_{[0,\infty]}\ o_r(t) - w(t) + mc < l_e(t)$ #Ego keeps at least the **m**inimum **c**learance from obstacles on the left of the single lane road

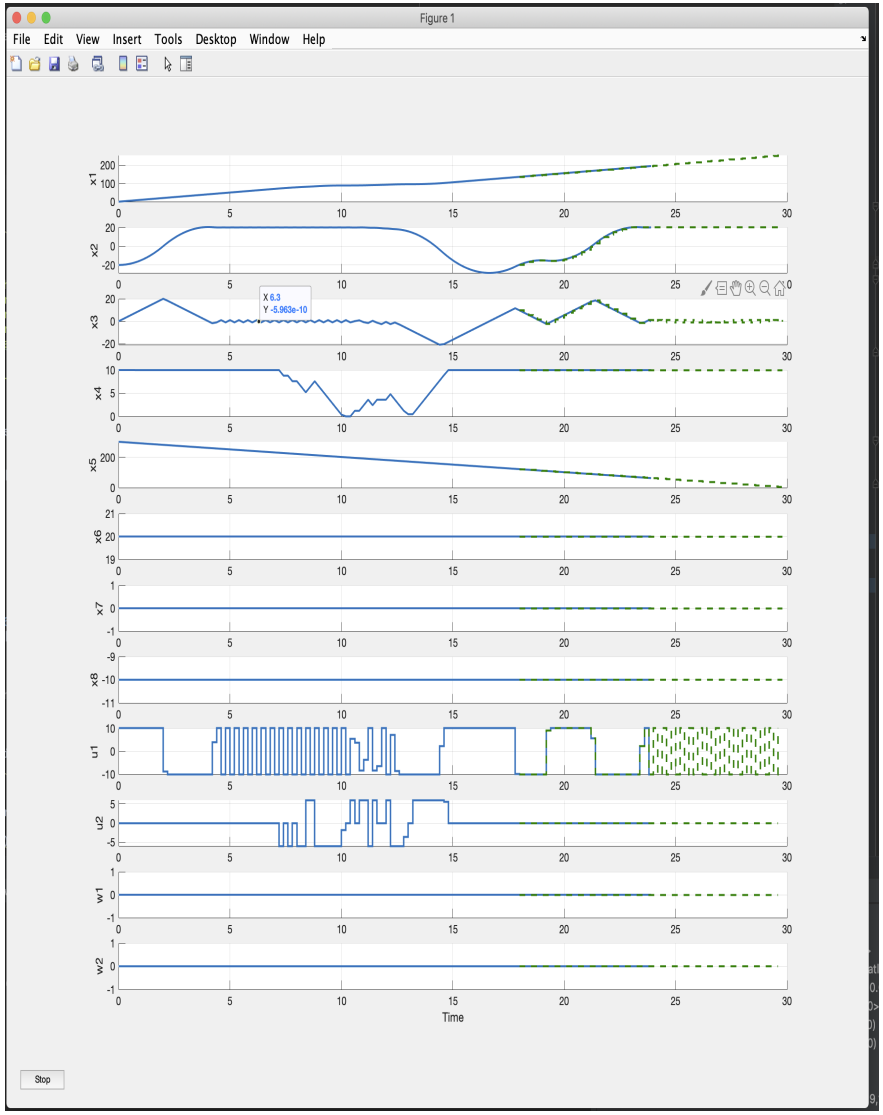
Experiment 2: One-way, navigating parked cars



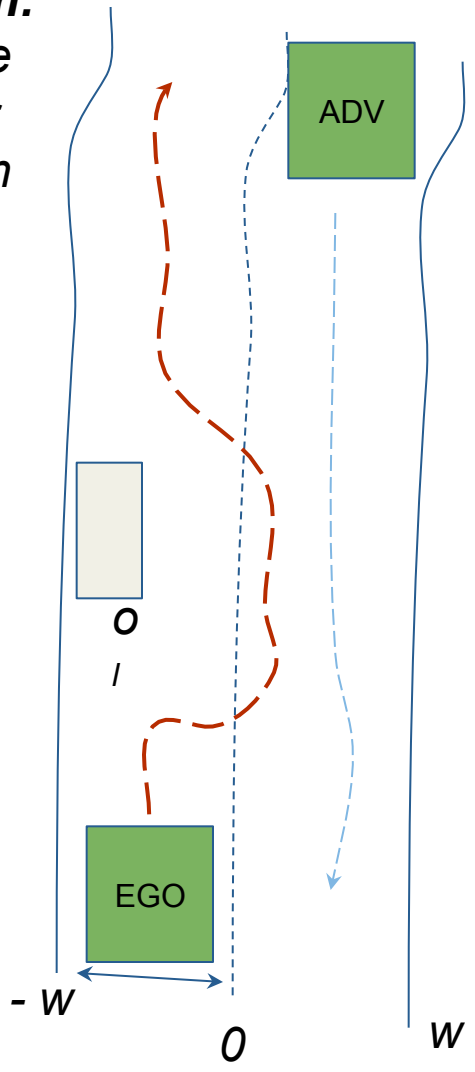
Interpretation:



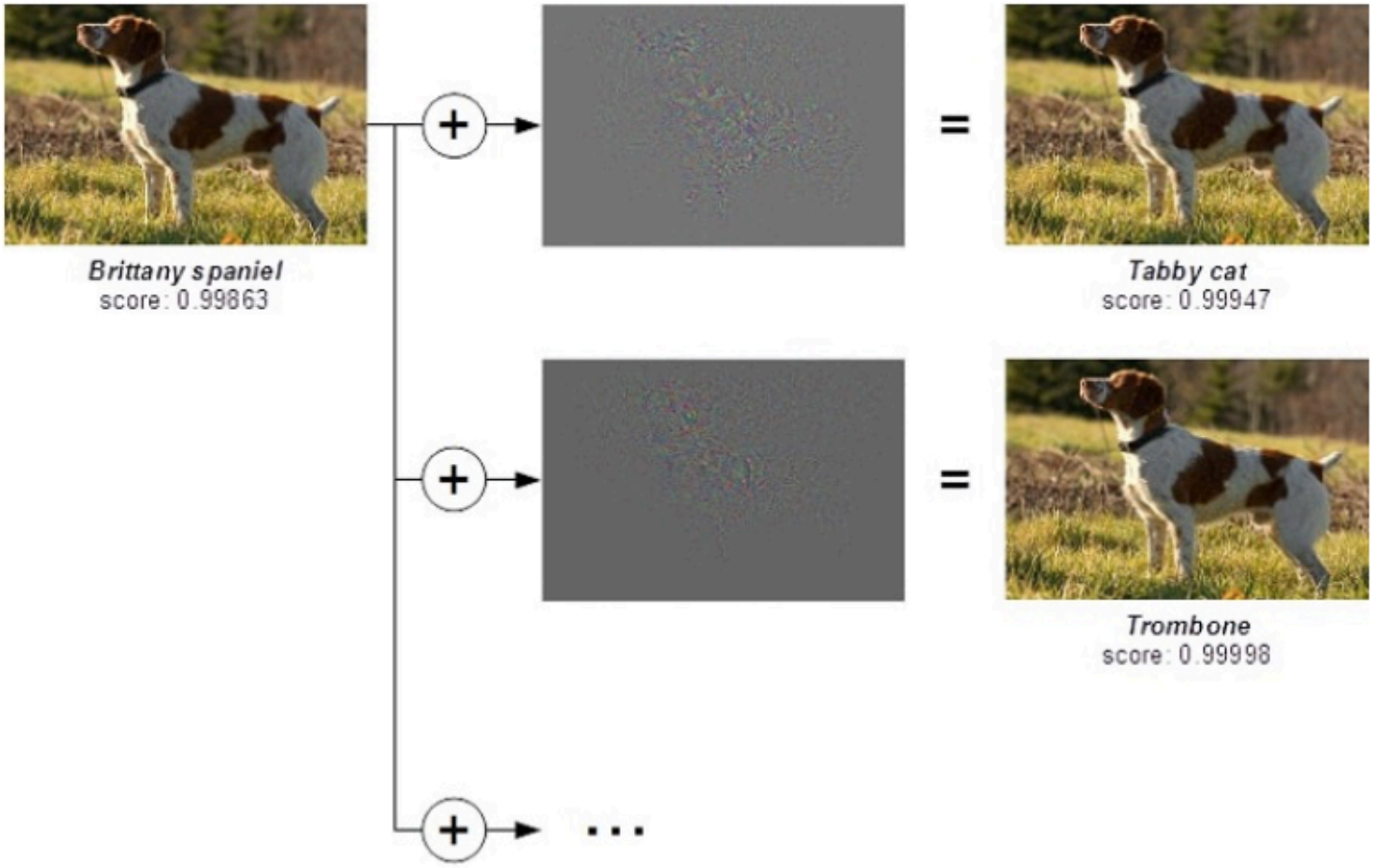
Experiment: Normal road, obstacle avoidance



Interpretation: *Slow to let the oncoming car pass and then overtake the obstacle*



Concept of Adversarial Examples in ML: Small Perturbation can Yield Major Misclassification



[https://www.kdnuggets.com/2018/10/adversarial-examples-explained.html]

Implications for Robotics

[N. Das et al., Shield: Fast, Practical Defense and Vaccination for Deep Learning using JPEG Compression, KDD 2018]

Many Ways to Attack ML Systems

- Evasion attacks: malicious objects are deliberately transformed to evade detection (detection being the prediction that these are malicious)
  - Attack on the learned model
- Data poisoning attacks: the adversary introduces malicious modifications to the data used for training
  - Attack on the algorithm

Adversarial examples can be thought of under the category of evasion attacks

How can an attack be constructed? Simple Example: Fast Gradient Sign Method (FGSM)

- Consider the classification problem defined by a loss function *L*, which takes image *x* to output a label *y*, using parameters *q*
- The gradient of this loss function is: $\nabla_x L(\theta, x, y)$

- Define a new image as:

$$x_{adv} = x + \epsilon \text{ sign } (\nabla_x L(\theta, x, y))$$

- This is a simple "white-box" attack. To defend against this, one could generate many such images at training time and augment the datasets
  - This wouldn't work for more sophisticated white box attacks, or for black box attacks

DIFFERENTIAL PRIVACY

What are the Issues?

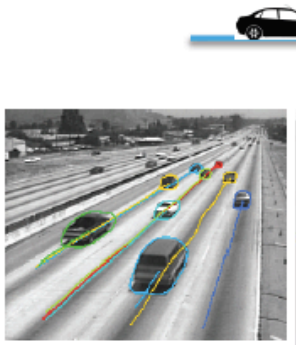
Consider an application such as traffic estimation:

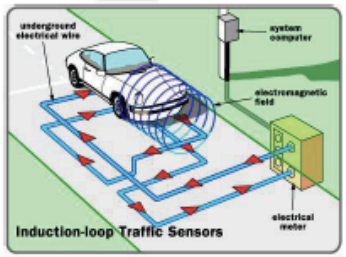
- Automotive traffic monitoring using probe vehicles with GPS receivers promises significant improvements in cost, coverage, and accuracy.
- They require participants to reveal their positions to an external traffic monitoring server.
- One solution is to use a 'Virtual Trip Line', chosen locations where vehicles provide updates (avoid sensitive areas)
- Can still be vulnerable to inference attacks, e.g., through good Kalman filter (or Bayes filter of some kind) based atacks

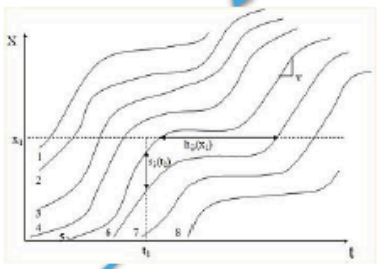
Privacy for Location based Services is Different from Anonymity, etc.

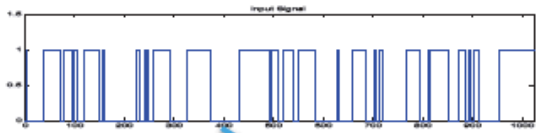
- The data collection process (communications, third party aggregator) is not the only risk
- Risks associated to data collection can typically be addressed by anonymization, cryptographic means and other system design tricks.
- Useful system cannot avoid a fundamental disclosure of information (ex: real-time traffic density map, travel time estimates, etc.)
- We need to ensure this does not leak too much information about **individuals**

A Problem: Can we get aggregate statistics without compromising individual privacy?



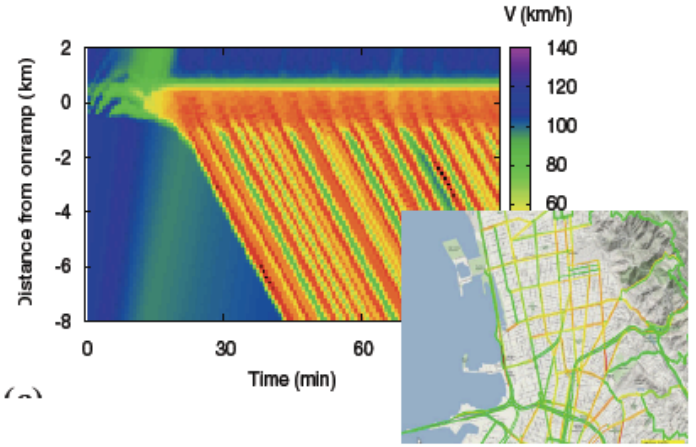






$$x_{i,t+1} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} x_{i,t} + \sigma_{i1} \begin{bmatrix} T_s^2/2 & 0 \\ T_s & 0 \end{bmatrix} w_{i,t},$$

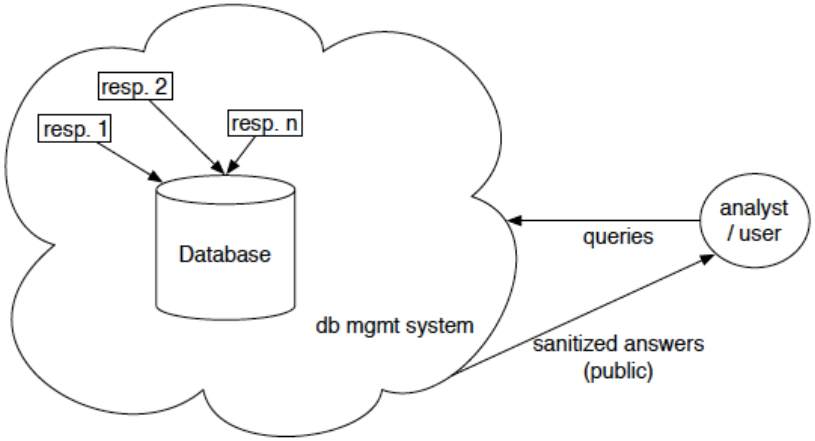
$$\frac{\partial \rho}{\partial t} + \frac{\partial q}{\partial x} = 0$$



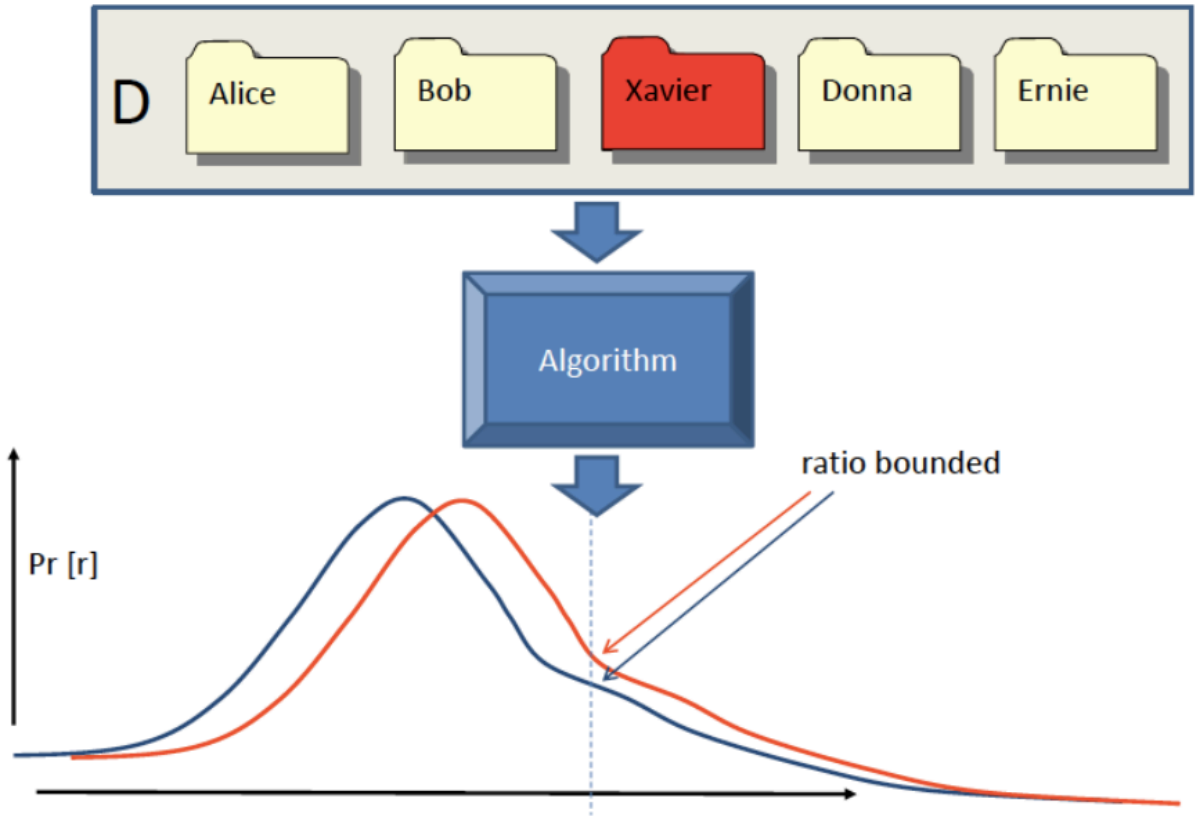
[Source: J.L.Ny, Polytechnique Montreal and GERAD]

Differential Privacy: Basic Idea

- A differentially private mechanism randomly perturbs its answer to a query so that the output distribution over answers does not vary much if any given individual changes its data (or even chooses not to participate)
- Hard to infer if the specific data of any individual was used or not to answer the query



Where Information Leakage Happens?



A Bit More Formally

- Adjacent datasets differ by data of a single individual
- A mechanism M that acts on the dataset, is (ε,δ)-differentially private if, for all valid sets S and nearby datasets defined by an adjacency relation $Adj(d,d')$:

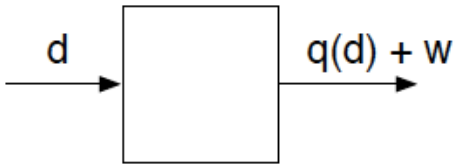
$$\mathrm{P}(\mathrm{M}(\mathrm{d}) \in S) \leq e^{\epsilon} P(M(d') \in S) + \delta$$

- Typically, ε is small (e.g., 0.1) and δ is very small (e.g., 0.01).
- If δ is 0 then the mechanism is said to be ε–differentially private (ε-DP)
- Privacy definition depends on adjacency definition: pair of datasets we wish to keep indistinguishable

A Gaussian DP Mechanism

- Let d be a dataset of salaries, d_i
- Let the query q be the average salary calculation: $q(d) = \frac{1}{n}\sum_{i=1}^{n} d_i$

- A Gaussian mechanism as below is (ε,δ)-differentially private:



$$q(d) + \kappa_{\delta,\epsilon}\, \Delta_2 q\, \mathcal{N}(0, I_m)$$

$$\kappa_{\delta,\epsilon} \in O(\sqrt{\ln(1/\delta)}/\epsilon)$$

where the L_p-sensitivity of a query is:

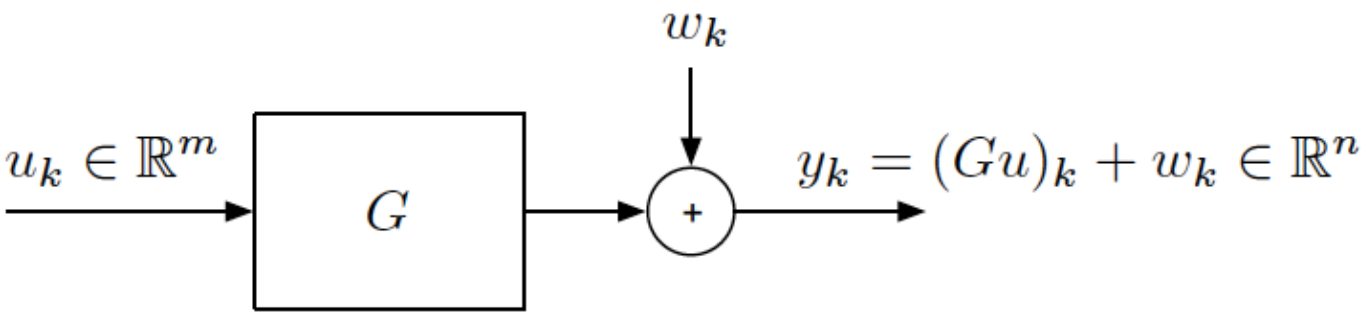
$$\Delta_p q := \max_{d,d':\mathrm{Adj(d,d')}} \|q(d) - q(d')\|_p$$

An Application to a Linear Dynamic System

- $\mathrm{Adj}(u,u')$ is the relation between nearby input signals
- The query is the action of the dynamics, G
- The dynamic system $y = Gu + w$ is ε-differentially private if we choose w to be Laplace white noise: $w_{k,i} \sim Lap(\Delta_1 G/\epsilon)$

Lap(b) pdf: $\frac{1}{2b}e^{-|x|/b}$, std. dev. $\sqrt{2}b$, iid on each component

$$\Delta_p G = \max_{\mathrm{Adj(u,u')}} \|Gu - Gu'\|_p$$



Discuss…

How does this help us address the traffic estimation case?