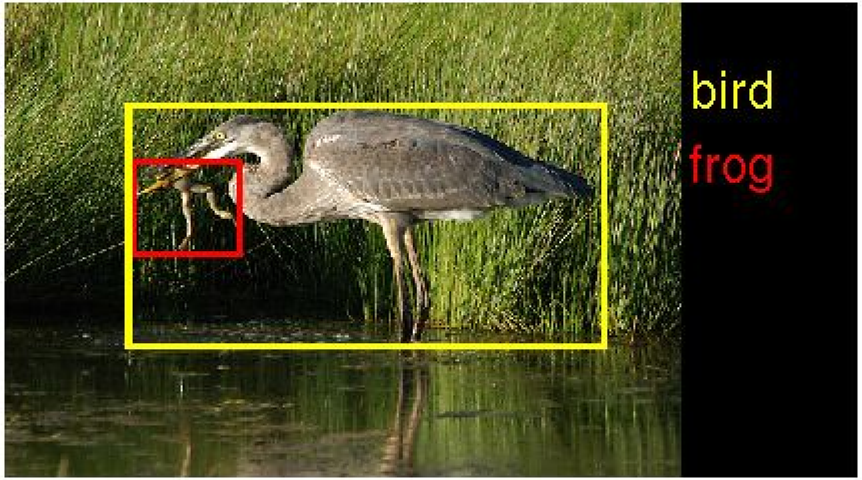
Google

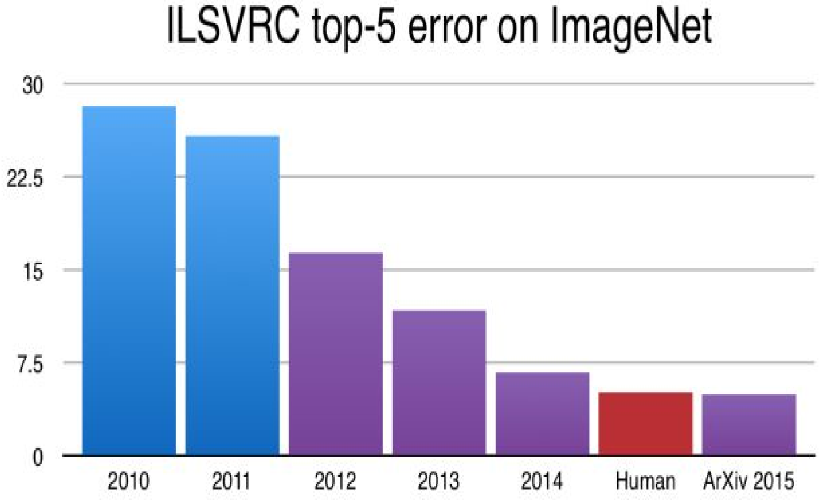
Optimization Perspectives on Robot Learning, Perception, and Control

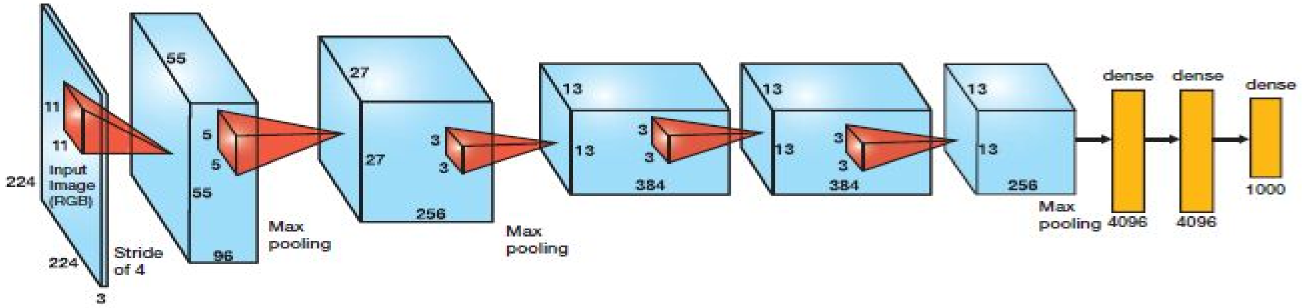
Vikas Sindhwani
Google Brain, NYC
sindhwani@google.com

There have been stunning advances in Machine Learning and Perception recently.

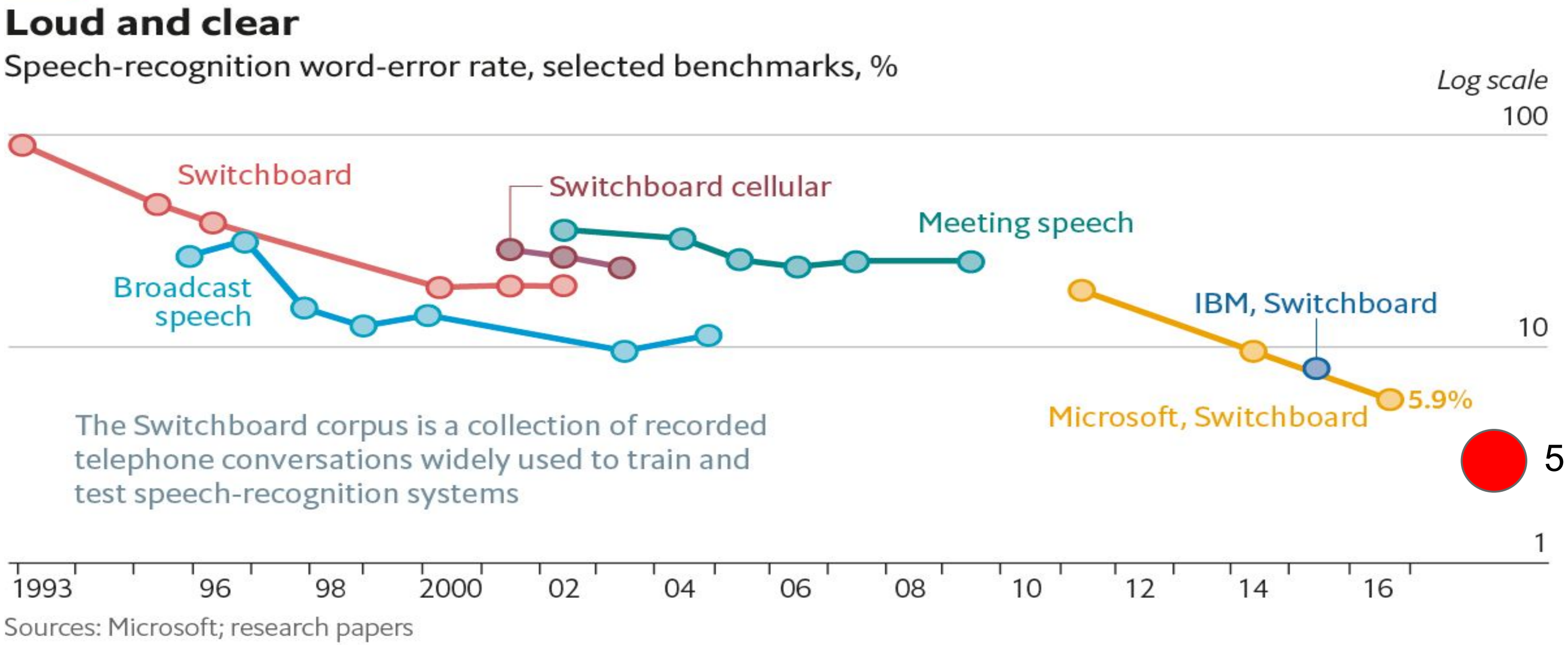
Computer Vision: The ImageNet Challenge







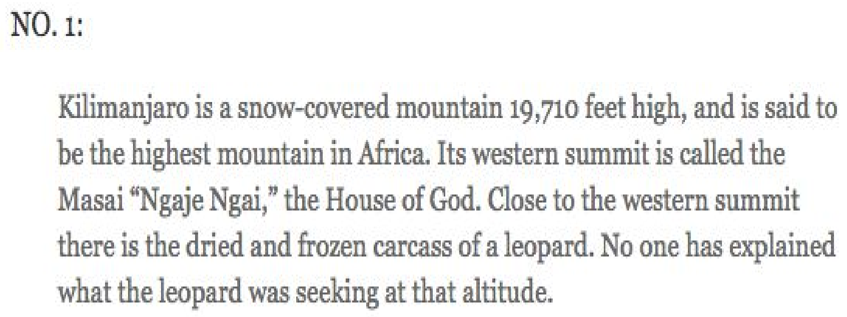
Speech Recognition



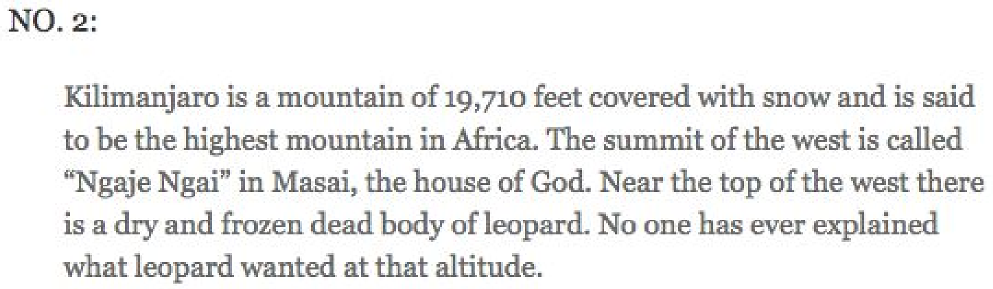
● 5.1%~Human

Machine Translation

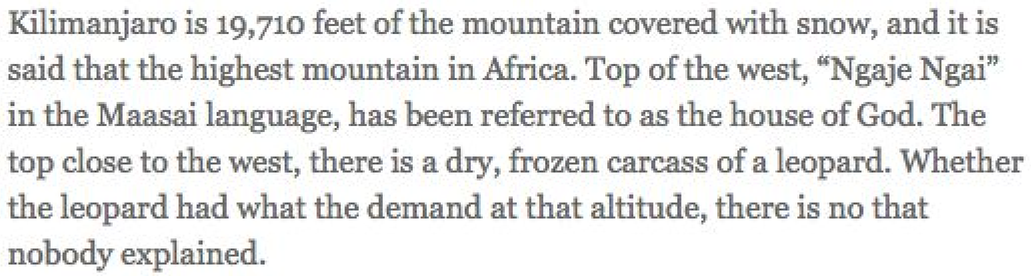
Japanese2English


NO. 1:

Kilimanjaro is a snow-covered mountain 19,710 feet high, and is said to be the highest mountain in Africa. Its western summit is called the Masai "Ngaje Ngai," the House of God. Close to the western summit there is the dried and frozen carcass of a leopard. No one has explained what the leopard was seeking at that altitude.

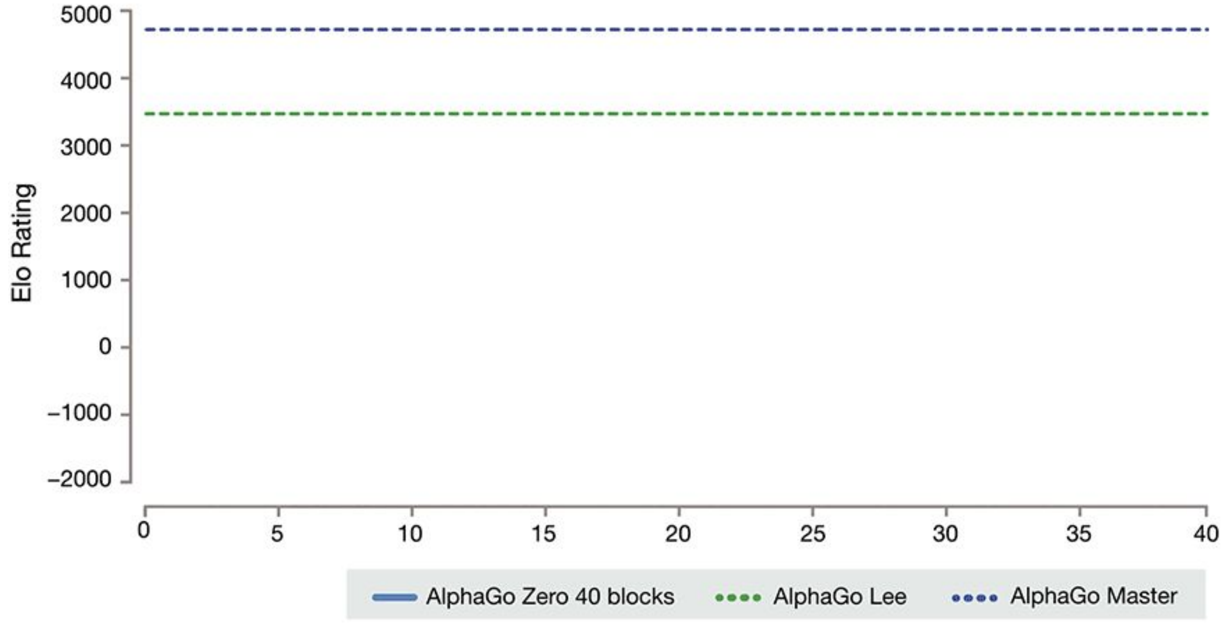

NO. 2:

Kilimanjaro is a mountain of 19,710 feet covered with snow and is said to be the highest mountain in Africa. The summit of the west is called "Ngaje Ngai" in Masai, the house of God. Near the top of the west there is a dry and frozen dead body of leopard. No one has ever explained what leopard wanted at that altitude.

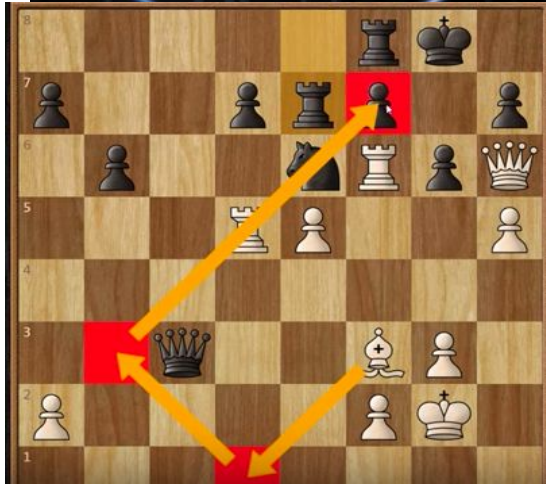

Kilimanjaro is 19,710 feet of the mountain covered with snow, and it is said that the highest mountain in Africa. Top of the west, "Ngaje Ngai" in the Maasai language, has been referred to as the house of God. The top close to the west, there is a dry, frozen carcass of a leopard. Whether the leopard had what the demand at that altitude, there is no that nobody explained.

Mastering Go and Chess -- from scratch

Train p(next-move|board), value(board) networks.



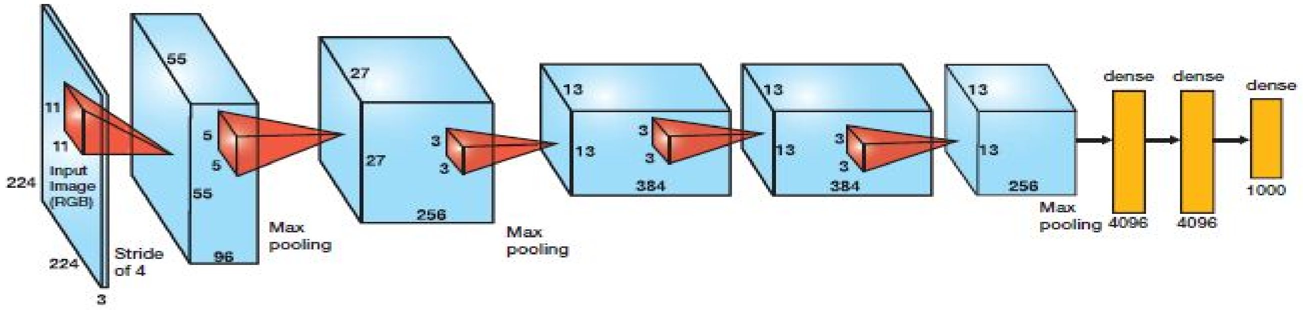
nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO



- Mastering the Game of Go without Prior Knowledge
- Mastering Chess and Shogi by Self-Play with a General RL algorithm

Whats behind this success?

- Tons of Data
- Distributed Computation
- Optimization!
- Complex end-to-end pipelines
    - Still an art with lots of open questions.
    - Mainly supervised learning.



What does this mean for the emerging world of Robotics?

Robots in Factories

- Body Shop
  - 380 robots, 450 humans
  - 6 hours per car (7500 spot welds)
- Paint shop
  - 100 robots
- Assembly line
  - ? robots, mostly manual
- 1400 cars per day

SENSE-PLAN-ACT paradigm

BMW Manufacturing Plant
Spartanburg, South Carolina

The Real World

PERCEPTION
LEARNING & ADAPTATION
CONTROL

Robotics at Alphabet

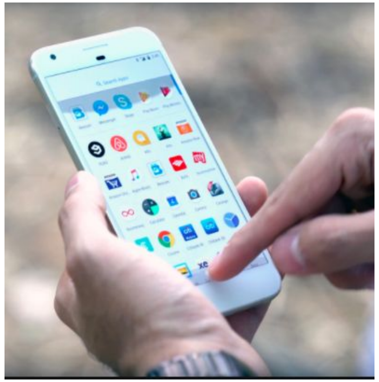
Self-driving Cars

- Early Rider Program in Phoenix, AZ
- 3M+ miles, 1B+ miles in sim (2016)
- 1.25M deaths (2014), 32K in US
- ~22K+ speeding, alcohol, distraction

Self-flying Vehicles

Arm Farm

From Smartphones to Robots: the Safety Challenge



Asimov's "Three Laws of Robotics" Handbook of Robotics, 2058 (1942)

- A robot may not injure a human being or, through inaction, allow a human being to come to harm

.

- A robot must obey orders given it by human beings except where such orders would conflict with the First Law.

- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Preliminaries and Background

7 things about Deep Learning

1. **Training and Test data drawn i.i.d from *same* distribution.**

$$\{(x_i, y_i)\}_{i=1}^{l} \sim p(x, y)$$

2. **Optimization Problem**

$$\arg\min_{\theta} \sum_{i=1}^{l} c(f_\theta(x_i), y_i) + \Omega(\theta)$$

7 things about Deep Learning

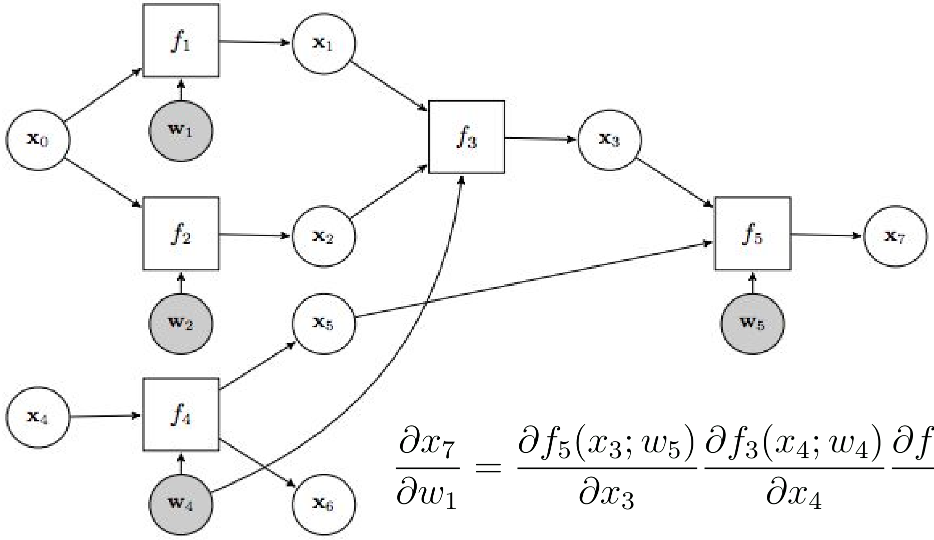
3. “Shallow” vs “Deep” Predictors

$$f_W(x) = W\phi(x) \qquad f_\theta(x) = f_{W_k}(f_{W_{k-1}}(\ldots f_{W_2}(f_{W_1}(x))\ldots)$$

$$f_W(x) = \sigma(Wx)$$

4. Computation Graphs & Backprop

Tensors flowing (B x H x W x D)



$$\frac{\partial x_7}{\partial w_1} = \frac{\partial f_5(x_3; w_5)}{\partial x_3}\frac{\partial f_3(x_4; w_4)}{\partial x_4}\frac{\partial f_1(x_1; w_1)}{\partial w_1}$$

7 things about Deep Learning

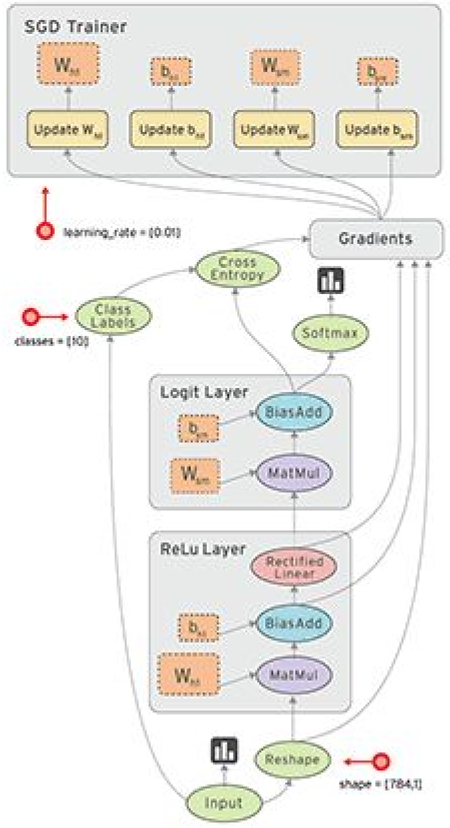
5. **Stochastic Gradient Descent**

$$\theta^{k+1} = \theta^k - \eta \sum_{i \in B_k} \nabla_\theta l(f_\theta(x_i), y_i)$$

Parallel comp; PS architecture

6. **Mature Autodiff Libraries**

E.g. TensorFlow, pytorch, caffe, ...



And the Last thing

7. “Alchemy”

- Choice of loss function
- Choice of architectures: CNNs, RNNs, DNNs,...
- Orchestration of the Optimization (e.g., learning rates)

7. It works!

Surprisingly effective despite non-convexity, with millions of parameters -- many local mins of similar quality.

Robots (+world) as Dynamical Systems

- States

$$x_t \in \mathbb{R}^n$$

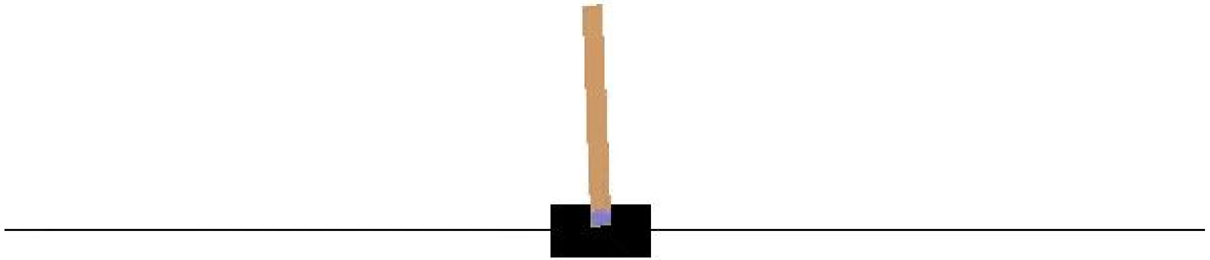
- Controls

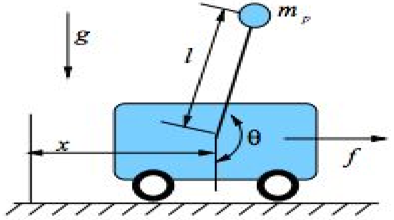
$$u_t \in \mathbb{R}^m$$

- Dynamics

$$\dot{x} = f(x, u)$$
$$x_{t+1} = f(x_t, u_t)$$

$$P(x_{t+1} | x_t, u_t)$$

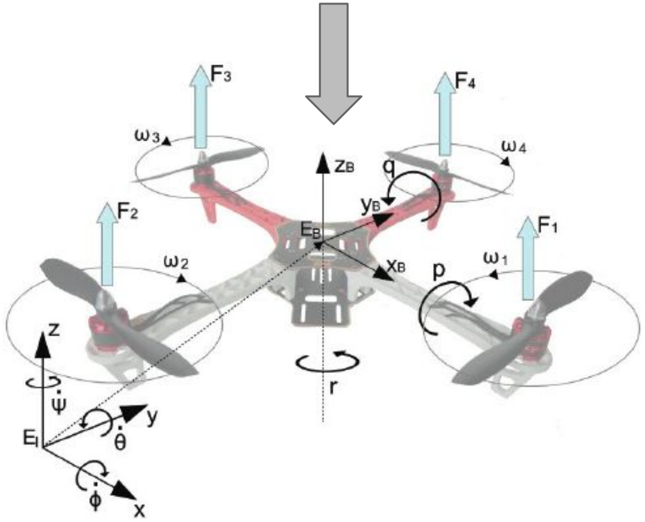




$$\ddot{x} = \frac{1}{m_c + m_p \sin^2\theta}\left[f + m_p \sin\theta (l\dot{\theta}^2 + g\cos\theta)\right]$$
$$\ddot{\theta} = \frac{1}{l(m_c + m_p \sin^2\theta)}\left[-f\cos\theta - m_p l\dot{\theta}^2 \cos\theta \sin\theta - (m_c + m_p) g \sin\theta\right]$$

Preliminaries: Robots (+world) as Dynamical Systems

RL lingo: “Agent”



Ingredients: Perception-Control Loop

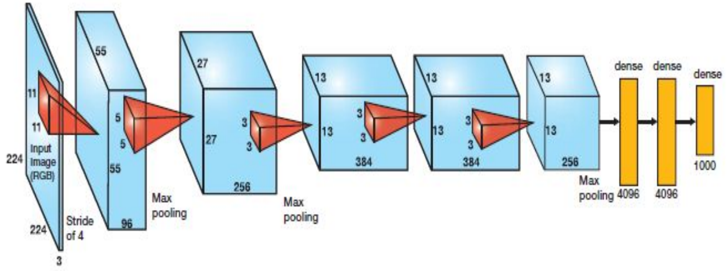
Controls/Actions

Steering
Throttle
Break

Policy

$$u_t = \pi_\theta(o_t)$$

Observation



Dynamics/"Model"

$$x_{t+1} = f(x_t, u_t)$$
$$o_t = g(x_t)$$

$$c_t(x_t, u_t)$$

Rewards/Costs

Where are the Learning Problems?

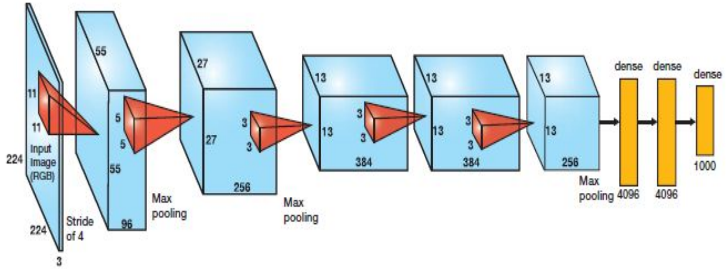
Controls/Actions

Steering
Throttle
Break

Policy

$$u_t = \pi_\theta(o_t)$$

Observation



Dynamics/"Model"

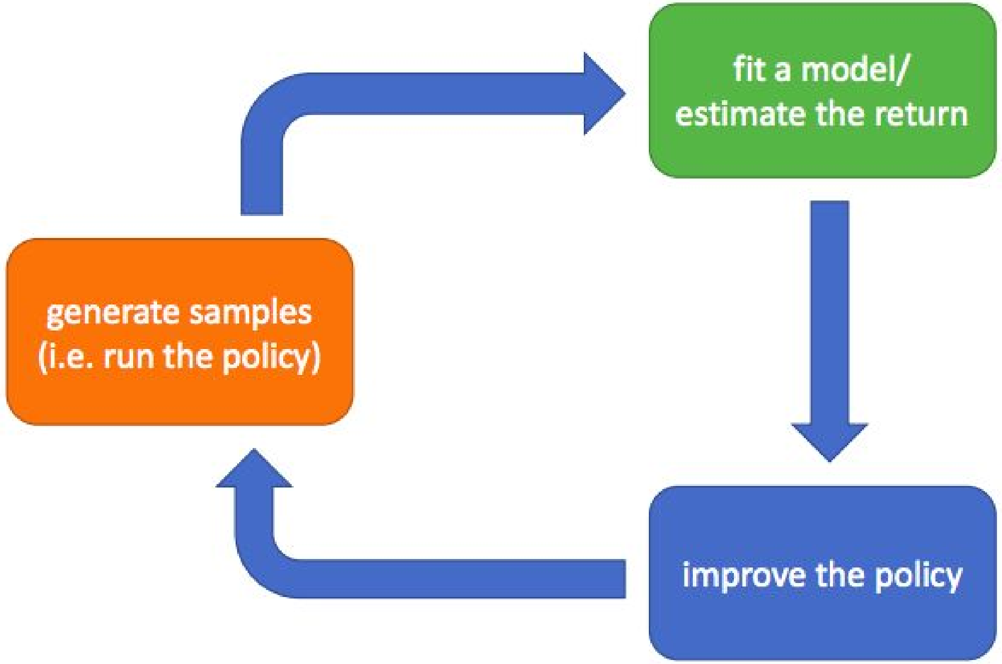
$$x_{t+1} = f(x_t, u_t)$$
$$o_t = g(x_t)$$

$$c_t(x_t, u_t)$$

Rewards/Costs

Google

Learning and Optimization



compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)

fit $Q_\phi(\mathbf{s}, \mathbf{a})$ (actor-critic, Q-learning)

estimate $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ (model-based)

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)

$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$ (Q-learning)

optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

(from Sergey Levine's slides.)

Many Sources of Complexity

- What is the current state?
  - State Estimation from high-dimensional observations from noisy sensors
- What is the Dynamics of the system?
  - Known (games, factories)
  - Stochastic
  - Discontinuous -- problems involving contact
  - May be completely unknown
- What costs/rewards should we use to elicit desired behavior?
- What policy parameterization to use?
- Exploitation-vs-Exploration?
- How to optimize?

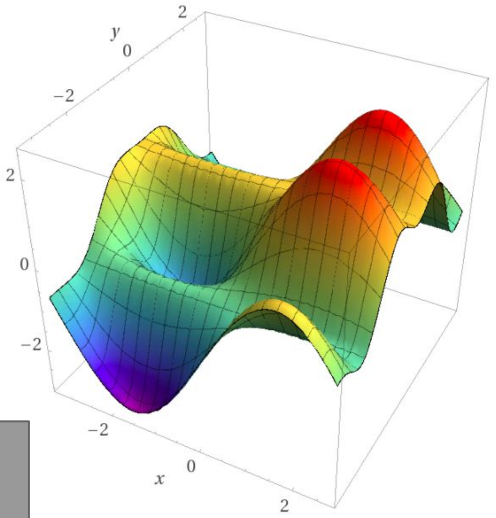
Optimization for Robot Learning, Perception, Control

Non-convex optimization

Behavior Cloning & Imitation Learning
Wing, Arm Farm, Self-driving cars

Structured Nonlinear Programming

Nonlinear Optimal Control
Iterative LQR, TROSS, MPC
Guided Policy Search



Dynamic Programming, Optimization & Sampling

Reinforcement Learning
Value-based methods
Policy Gradient
Model-based RL

Derivative Free Optimization

Learning policies in Simulation

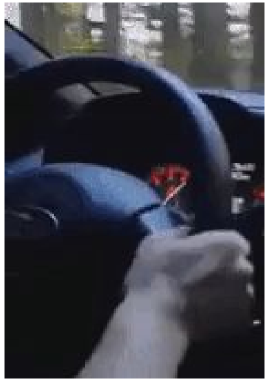
Research Tools

Polynomial Optimization & SDPs

Safety and Stability
Stability of Dynamical Systems
Polynomial Safety Shields

Imitation Learning and Behavior Cloning

- Learning policies by mimicking human decisions and behaviors



Driving: large datasets

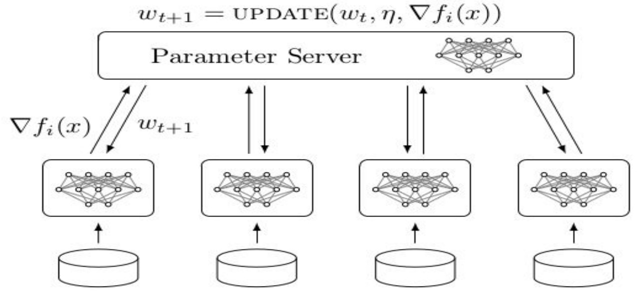
Manipulation (Daydream coffee study)

Doesnt work for all problems, e.g. bipedal locomotion

- Does not need dynamics-free -- and supervised learning works well!

Imitation Learning and Behavior Cloning

- Optimization Problem -- supervised learning



Demonstrations

$$\{(o_t^i, u_t^{i*})\}_{t,i}$$

Training

$$\min_\theta \sum_{i,t} loss(u_t^{i*}, \pi_\theta(o_t^i)) + \Omega(\theta)$$

SGD-based Optimization

$$\theta^{k+1} = \theta^k - \sum_{i,t \in B_k} \nabla_\theta loss(u_t^{i*}, \pi_\theta(o_t^i))$$

Distributed Training: Parameter-server architecture

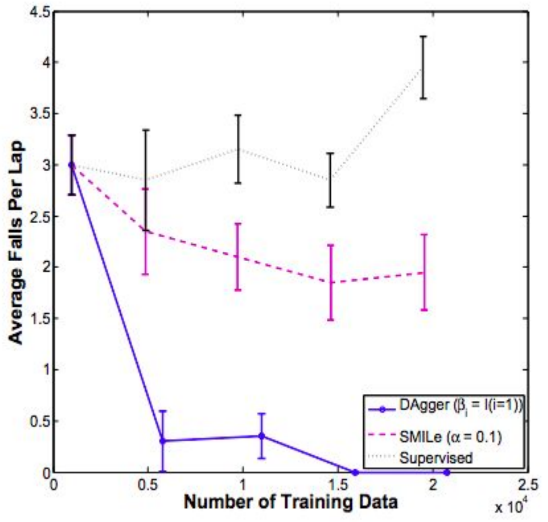
Naive Supervised Learning

Ross, Gordon and Bagnell, 2011 -- Cart racing experiments

Imitation Learning =/= Supervised Learning

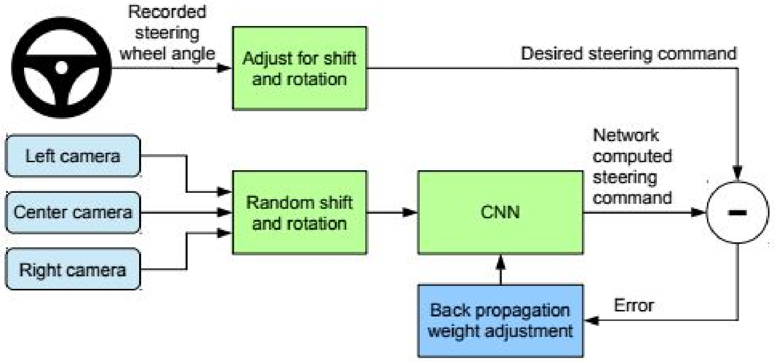
- Training/Test Distribution mismatch
    - With perception in the loop, encountered states become functions of the policy -- contrast with supervised learning (e.g. photo tagging).
- Dealing with cascading errors
- DAGGER algorithm

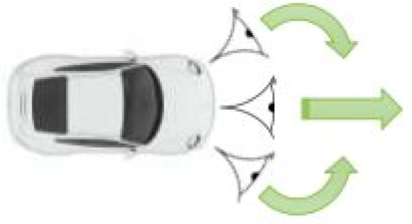
1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \ldots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

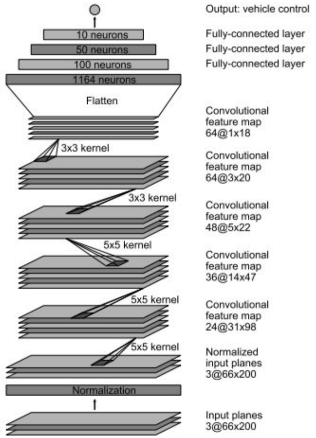


NVIDIA Self-driving

How do they deal with cascadling errors?


Recorded steering wheel angle
Adjust for shift and rotation
Desired steering command
Left camera
Center camera
Right camera
Random shift and rotation
CNN
Network computed steering command
-
Back propagation weight adjustment
Error





72 hours driving data, 250K parameters, 2 interventions every 10 mins

Safer Air Sensing on Self-flying Vehicles

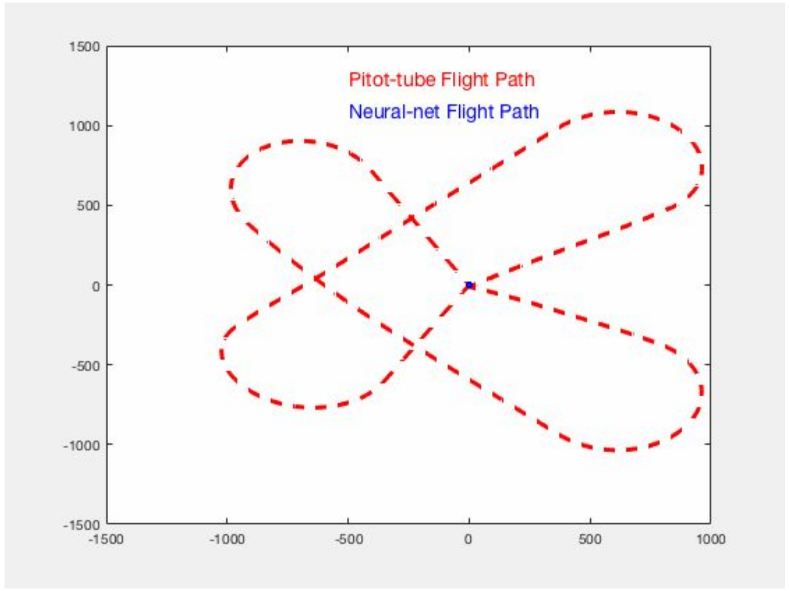
go/aerial_robotics

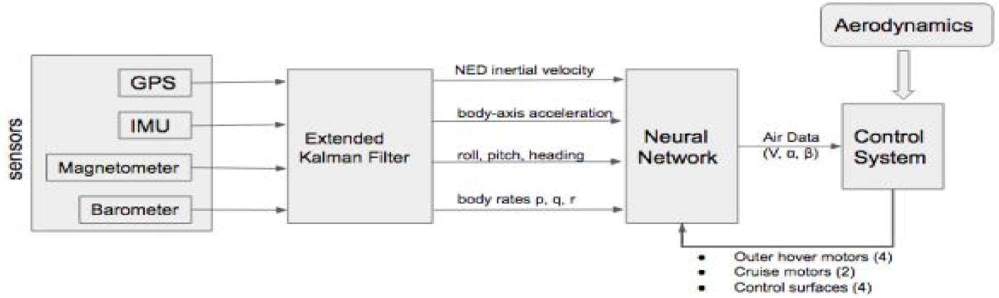
- Accurate sensing of relative motion wrt air is critical for safe & efficient control of UAVs on high-speed outdoor missions.
- Pitot tubes are airspeed sensors, prone to failure (Air France 2009) and hard to Maintain on small UAVs.
- Can we clone the Pitot tube using a neural net trained on hundreds of flight logs?

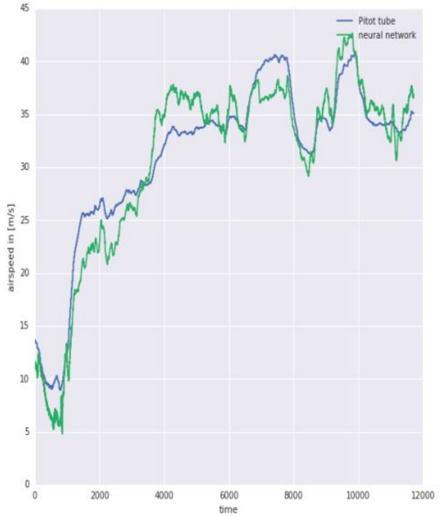
Pitot Tube
(Henri Pitot, 1695-1771)

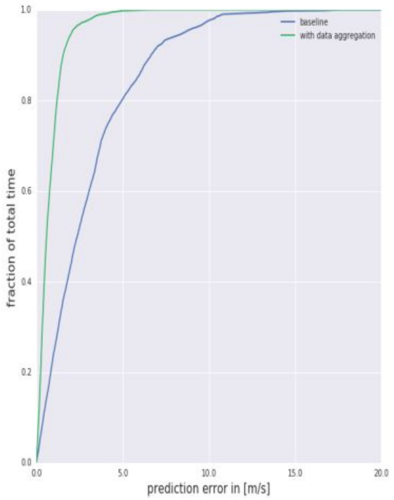
Safer Air Sensing on Self-flying Vehicles

go/aerial_robotics

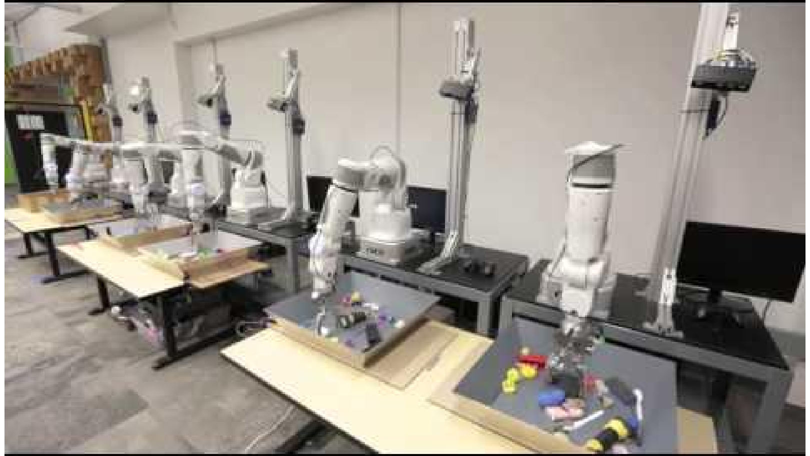


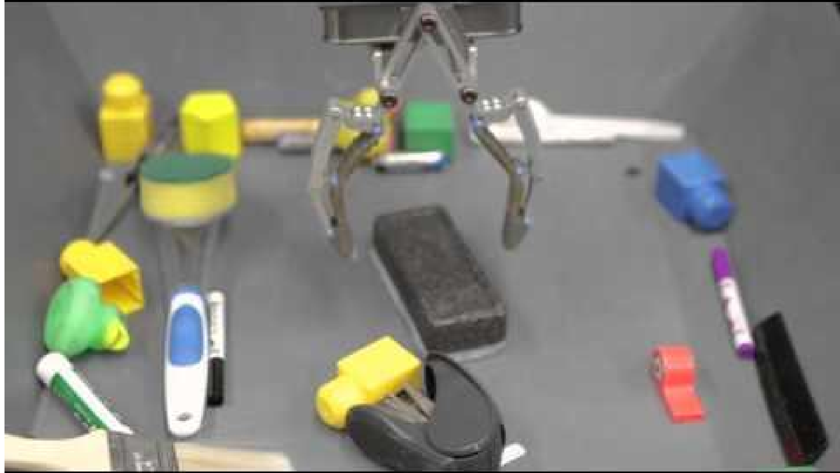






The Arm Farm: Self supervision





Training in 800K grasp attempts on 14 robots.

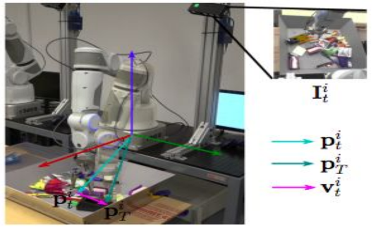


Figure 3. Diagram of the grasp sample setup. Each grasp i consists of T time steps, with each time step corresponding to an image $\mathbf{I}_t^i$ and pose $\mathbf{p}_t^i$. The final dataset contains samples $(\mathbf{I}_t^i, \mathbf{p}_T^i - \mathbf{p}_t^i, \ell_i)$ that consist of the image, a vector from the current pose to the final pose, and the grasp success label.

Tools 1: Supervised Learning in TensorFlow

- Colaboratory
- TensorFlow

Optimal Control and Trajectory Optimization

(example due to tassa@)

states

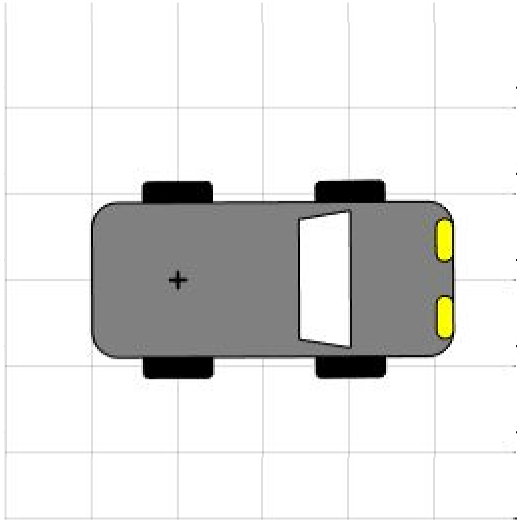
$$\mathbf{x} = (p_x, p_y, \theta, v)$$

controls

$$\mathbf{u} = (\omega, a)$$

dynamics

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \begin{pmatrix} b(v,\omega)cos(\theta) \\ b(v,\omega)sin(\theta) \\ sin^{-1}\left(sin(\omega)f(v)d^{-1}\right) \\ ha \end{pmatrix}$$



$$-0.5\ rad \le \omega \le 0.5\ rad$$
$$-2m/s^2 \le a \le 2m/s^2$$

$$\min_{\mathbf{u}_0 \dots \mathbf{u}_{T-1}} \sum_{t=0}^{T-1} c_t(\mathbf{x}_t, \mathbf{u}_t) + c_T(\mathbf{x}_T)$$

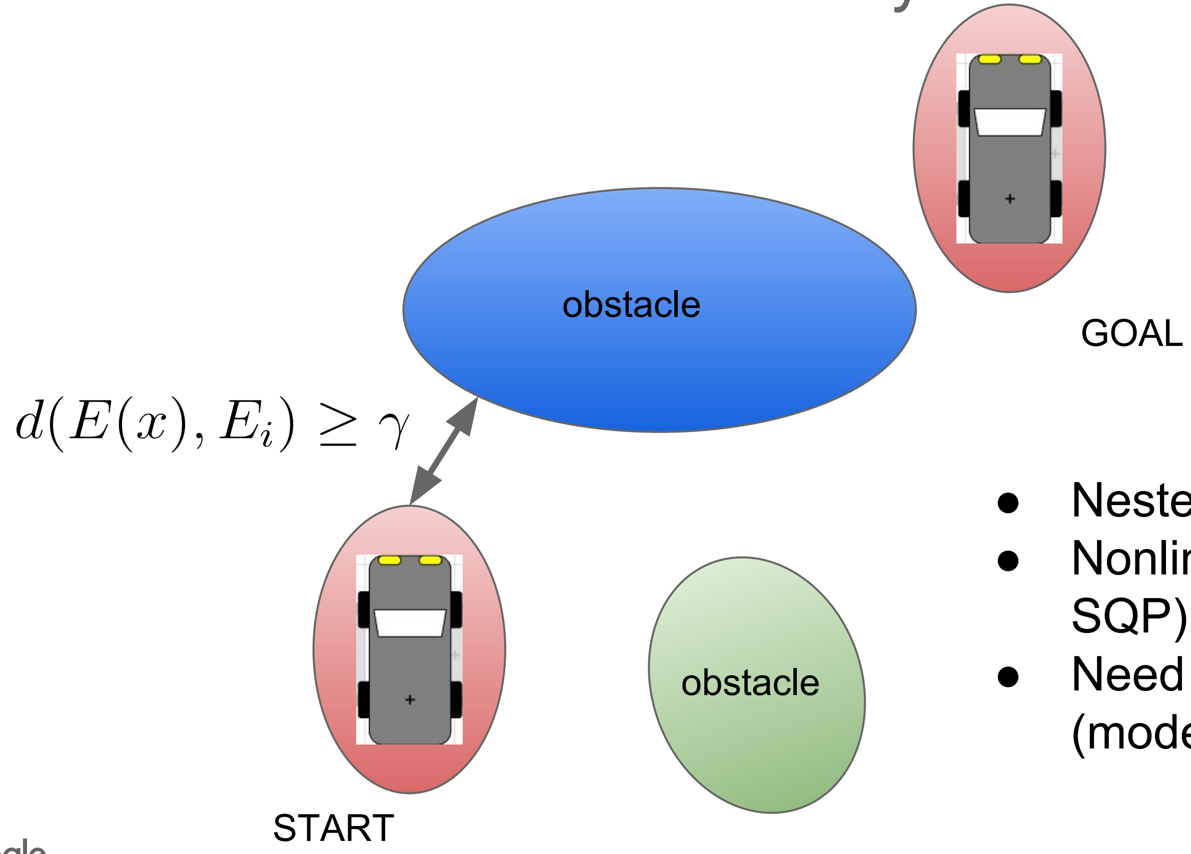
subject to:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$$

$$\begin{pmatrix} -0.5 \\ -2.0 \end{pmatrix} \le \mathbf{u}_t \le \begin{pmatrix} +0.5 \\ +2.0 \end{pmatrix}$$

dynamics: stochastic, discontinuous, complex simulations, unknown

Obstacle Avoidance with Safety Shields



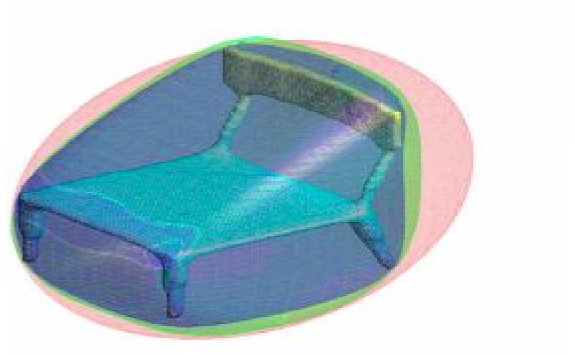
- Nested Optimization
- Nonlinear Programming (e.g., SQP) *with rich structure*.
- Need for real-time optimization (model predictive control)

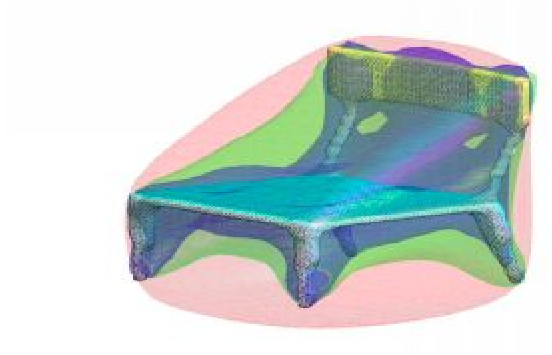
Constructing Safety Shields in 3D Environments (RSS-2017)

$$D = \{x_1, x_2, \ldots x_m\} \quad \subseteq \quad S_p = \{x \in \mathbb{R}^3 | p(x) \leq 1\}$$

convex, increasing degree

increasing non-convexity





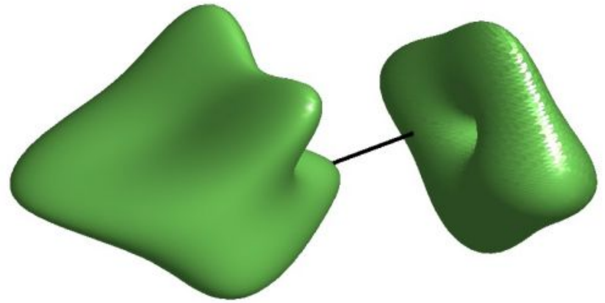


Fig. 6: Minimum distance between two (nonconvex) sublevel sets of degree-4 polynomials

Polynomial Optimization ⇒ Semi-definite Programming relaxations.

Back to Optimal Control: Linear Quadratic Regulators

$$\min_{\mathbf{u}_0 \dots \mathbf{u}_{T-1}} \sum_{t=0}^{T-1} c_t(\mathbf{x}_t, \mathbf{u}_t) + c_T(\mathbf{x}_T)$$

<u>subject to:</u>

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$$

$$\min_{u_0 \dots u_{T-1}} \sum_{t=0}^{T-1} x_t^T Q x_t + u_t^T R u_t + x_T^T Q_T x_T$$

$$x_{t+1} = Ax_t + Bu_t$$

$$x_0 \;\; \text{is given}$$

LQR: Value Function

Define sequence of functions,

$$V_t : \mathbb{R}^n \mapsto \mathbb{R}, t = 0 \dots T$$

that are the minimum values achieved for the “tail subproblems” from a given state,

$$V_t(z) = \min_{u_t \dots u_{T-1}} \sum_{t=t}^{T-1} x_t^T Q x_t + u_t^T R u_t + x_T^T Q_T x_T, \qquad x_t = z$$

Notice,

$$V_0(x_0) \qquad\qquad V_T(z) = z^T Q_T z$$

LQR Dynamic Programming

Bellman Equation

$$V_t(z) = \min_u \left[z^T Q z + u^T R u + \underbrace{\min_{u_{t+1} \ldots u_{T-1}} \sum_{t+1}^{T-1} x_t^T Q x_t + u_t^T R u_t + x_T^T Q_T x_T}_{V_{t+1}(Az+Bu)} \right]$$

- cost at time t for taking action u at state z.
- min cost-to-go for where you land at t+1 as a consequence

Assume: $$V_{t+1}(z) = z^T P_{t+1} z, \qquad P_T = Q_T$$

LQR Algebra

- by DP,

$$V_t(z) = z^T Qz + \min_w \left(w^T Rw + (Az + Bw)^T P_{t+1}(Az + Bw)\right)$$

- can solve by setting derivative w.r.t. w to zero:

$$2w^T R + 2(Az + Bw)^T P_{t+1}B = 0$$

- hence optimal input is

$$w^* = -(R + B^T P_{t+1}B)^{-1}B^T P_{t+1}Az$$

$$\begin{aligned} V_t(z) &= z^T Qz + w^{*T} Rw^* + (Az + Bw^*)^T P_{t+1}(Az + Bw^*) \\ &= z^T \left(Q + A^T P_{t+1}A - A^T P_{t+1}B(R + B^T P_{t+1}B)^{-1}B^T P_{t+1}A\right) z \\ &= z^T P_t z \end{aligned}$$

where

$$P_t = Q + A^T P_{t+1}A - A^T P_{t+1}B(R + B^T P_{t+1}B)^{-1}B^T P_{t+1}A$$

LQR solution

```
1. set $P_N := Q_f$

2. for $t = N, \ldots, 1,$

       $$P_{t-1} := Q + A^T P_t A - A^T P_t B (R + B^T P_t B)^{-1} B^T P_t A$$

3. for $t = 0, \ldots, N-1$, define $K_t := -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A$

4. for $t = 0, \ldots, N-1$, optimal $u$ is given by $u_t^{\mathrm{lqr}} = K_t x_t$
```

Extensions: Time-varying LQR, Infinite-horizon LQR

LQR is readily extended to handle time-varying systems

$$x_{t+1} = A_t x_t + B_t u_t$$

and time-varying cost matrices

$$J = \sum_{\tau=0}^{N-1} \left(x_\tau^T Q_\tau x_\tau + u_\tau^T R_\tau u_\tau \right) + x_N^T Q_f x_N$$

Steady State Solution for Infinite Horizon Problems: Algebraic Ricatti Equation

$$P_{\rm ss} = Q + A^T P_{\rm ss} A - A^T P_{\rm ss} B (R + B^T P_{\rm ss} B)^{-1} B^T P_{\rm ss} A$$

Tools: Cartpole Balancing on OpenAI Gym

$$\ddot{x} = \frac{1}{m_c + m_p \sin^2\theta}\left[f + m_p \sin\theta(l\dot{\theta}^2 + g\cos\theta)\right]$$

$$\ddot{\theta} = \frac{1}{l(m_c + m_p \sin^2\theta)}\left[-f\cos\theta - m_p l\dot{\theta}^2\cos\theta\sin\theta - (m_c + m_p) g \sin\theta\right]$$

- Power of Linearization

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) \approx \mathbf{f}(\mathbf{x}^*,\mathbf{u}^*) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]_{\mathbf{x}=\mathbf{x}^*,\mathbf{u}=\mathbf{u}^*}(\mathbf{x}-\mathbf{x}^*) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right]_{\mathbf{x}=\mathbf{x}^*,\mathbf{u}=\mathbf{u}^*}(\mathbf{u}-\mathbf{u}^*)$$

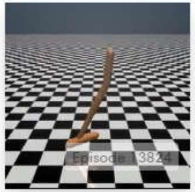
- Surprisingly large basins of attraction
- Why does it fail outside that basin?

Other Gym Tasks

- Average reward per episode over N episodes
- Total number of episodes

HalfCheetah-v0
Make a 2D cheetah robot run.

Swimmer-v0
Make a 2D robot swim.

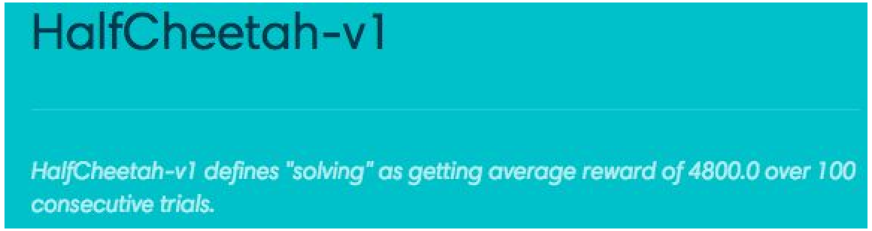


Hopper-v0
Make a 2D robot hop.

Walker2d-v0
Make a 2D robot walk.

Ant-v0
Make a 3D four-legged robot walk.

Humanoid-v0
Make a 3D two-legged robot walk.



HalfCheetah-v1

HalfCheetah-v1 defines "solving" as getting average reward of 4800.0 over 100 consecutive trials.

Iterative LQR/Differential Dynamic Programming

Nonlinear Optimal Control Problem:

$$\min_{\mathbf{u}_0 \ldots \mathbf{u}_{T-1}} \sum_{t=0}^{T-1} c_t(\mathbf{x}_t, \mathbf{u}_t) + c_T(\mathbf{x}_T)$$

$$\text{where: } \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \quad t = 0 \ldots (T-1)$$

Equivalent unconstrained problem:

$$\min_{u_0 \ldots u_{T-1}} J(U) = c_0(x_0, u_0) + c_1(f(x_0, u_0), u_1) + c_2(f(f(x_0, u_0), u_1), u_2) \ldots$$

Newton's Method: $U_{k+1} = U_k - [\nabla^2 J(U_k)]^{-1} [\nabla J(U_k)]$

Iterative LQR/DDP

- Initialize $u_0^{(0)}, \dots u_{T-1}^{(0)}$
- For k=0, ...until convergence
  - Simulate $x_{t+1}^{(k)} = f(x_t^{(k)}, u_t^{(k)}), \quad t = 0 \dots (T-1)$
  - Linearize Dynamics around current trajectory:

$$x_{t+1}^{(k)} = f(x_t^{(k)}, u_t^{(k)}) + \left[\frac{\partial f(x_t^{(k)}, u_t^{(k)})}{\partial x}\right](x - x_t^{(k)}) + \left[\frac{\partial f(x_t^{(k)}, u_t^{(k)})}{\partial u}\right](u - u_t^{(k)})$$

  - Solve Time-Varying LQR Problem

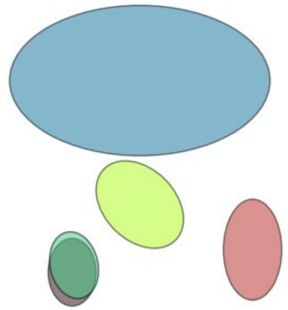
$$\begin{aligned} J &= \sum_{\tau=0}^{N-1} (x_\tau + \delta x_\tau)^T Q (x_\tau + \delta x_\tau) \\ &+ \sum_{\tau=0}^{N-1} (u_\tau + \delta u_\tau)^T R (u_\tau + \delta u_\tau) \end{aligned}$$

$$\delta x_{t+1} = A_t \delta x_t + B_t \delta u_t$$

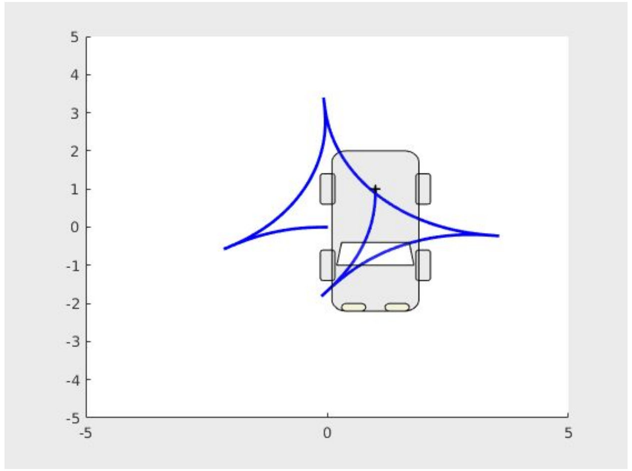
  - Set $u_t := u_t + \delta u_t$

Examples

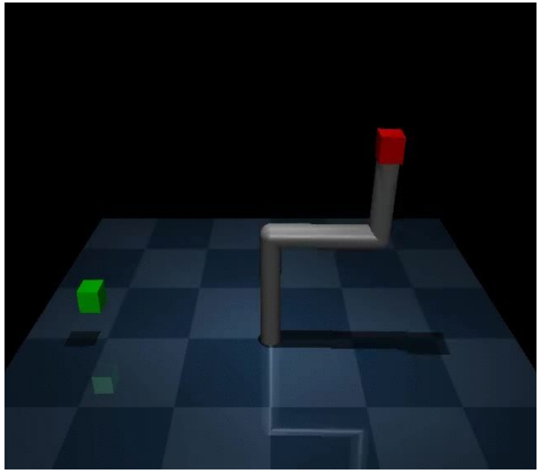
With an extension of iLQR to handle constraints.



path length
control sparsity
obstacle avoidance



Car parking with control limits.



In Mujoco: Torque control of a manipulator.

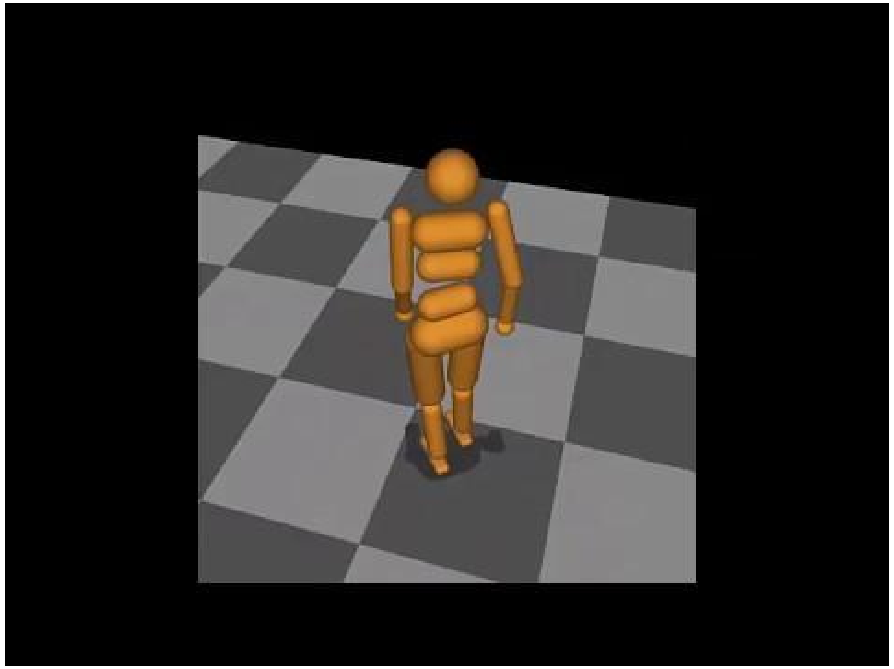
Model Predictive Control / Receding Horizon Control

- Due to errors in dynamics/ change in the environment, executing "open-loop" controls $u_0 \ldots u_{T-1}$

  may no longer be optimal.

- Replanning into the future at every time-step, defines a closed-loop policy:

$$u_t^* = \pi_{MPC}(x_t)$$



$$\arg \min_{u_t, \ldots u_{t+T}} \sum_{t=t}^{T-1} c_t(x_t, u_t) + c_T(x_T) \quad \text{subject to } x_{t+k+1} = f(x_{t+k}, u_t), \quad x_t \text{ given}$$

Model-based Reinforcement Learning

- Collect a bunch of trajectories by executing e.g. random controls

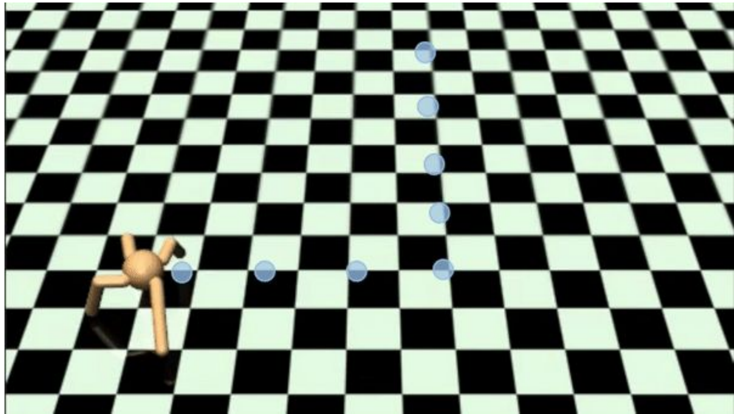
$$\tau^1, \tau^2 \dots \tau^N \qquad \tau^i = \{x_0^i, u_0^i, x_1^i, u_1^i, \dots x_{T_i}^i, u_{T_i}^i\}$$

- Fit a Dynamics Model

$$\hat{f}_\theta = \arg\min_\theta \sum_{t,k} \|x_{t+1}^k - f_\theta(x_t^k, u_t^k)\|_2^2$$

Nagabandi et al, 2017

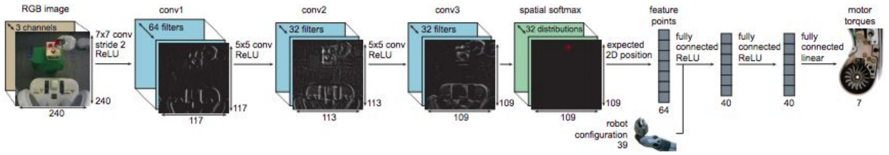
- Solve Optimal Control wrt current dynamics
- Collect new trajectories
- Refit new dynamics model
- Repeat



Guided Policy Search: Vision from Optimal Control

End-to-End Training of Deep Visuomotor Policies

Key idea: Use perfect-state optimal controllers to supervise learning of visual policies.



Derivative-Free Optimization

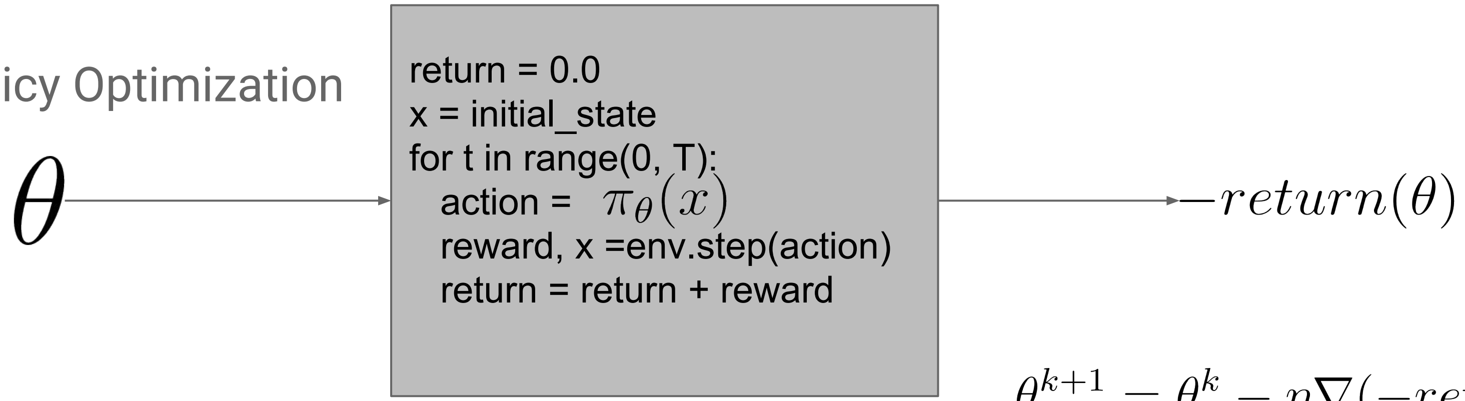
- Optimization problems where gradients are unavailable are pervasive.
  - Complex simulators
  - Legacy Code
  - Inner optimization routines
- How can we still do gradient descent?

$$\min_x f(x)$$

$$x^{k+1} = x^k - \eta \nabla f(x^k)$$

DFO for Policy Optimization and Optimal Control

- Policy Optimization



```
return = 0.0
x = initial_state
for t in range(0, T):
    action = $\pi_\theta(x)$
    reward, x =env.step(action)
    return = return + reward
```

$$\theta^{k+1} = \theta^k - \eta \nabla(-return(\theta^k))$$

- Optimal Control: Need Jacobians for Linearization Step

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \approx \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]_{\mathbf{x}=\mathbf{x}^*, \mathbf{u}=\mathbf{u}^*} (\mathbf{x} - \mathbf{x}^*) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right]_{\mathbf{x}=\mathbf{x}^*, \mathbf{u}=\mathbf{u}^*} (\mathbf{u} - \mathbf{u}^*)$$

DFO: Finite Differences

Taylor Approximation: For any perturbation direction p,

$$f : \mathbb{R}^n \to \mathbb{R}^m \quad : \qquad f(x^k + p) \approx f(x^k) + \frac{\partial f(x^k)}{\partial x} p \dots$$

So,

$$p = \epsilon d, \qquad \frac{\partial f(x^k)}{\partial x} d \approx \frac{1}{\epsilon} \left[f(x^k + \epsilon d) - f(x^k) \right]$$

Classic Finite Differences:

$$d = e_i, \qquad \left[\frac{\partial f(x^k)}{\partial x} \right]_{:,i} = \frac{\partial f(x^k)}{\partial x} e_i \approx \frac{1}{\epsilon} \left[f(x^k + \epsilon e_i) - f(x^k) \right]$$

DFO: Linear Regression

- Choose a set of perturbation directions $d_1 \ldots d_k$
- Compute finite differences

$$r_i = \left[f(x^k + \epsilon d_i) - f(x^k) \right]$$

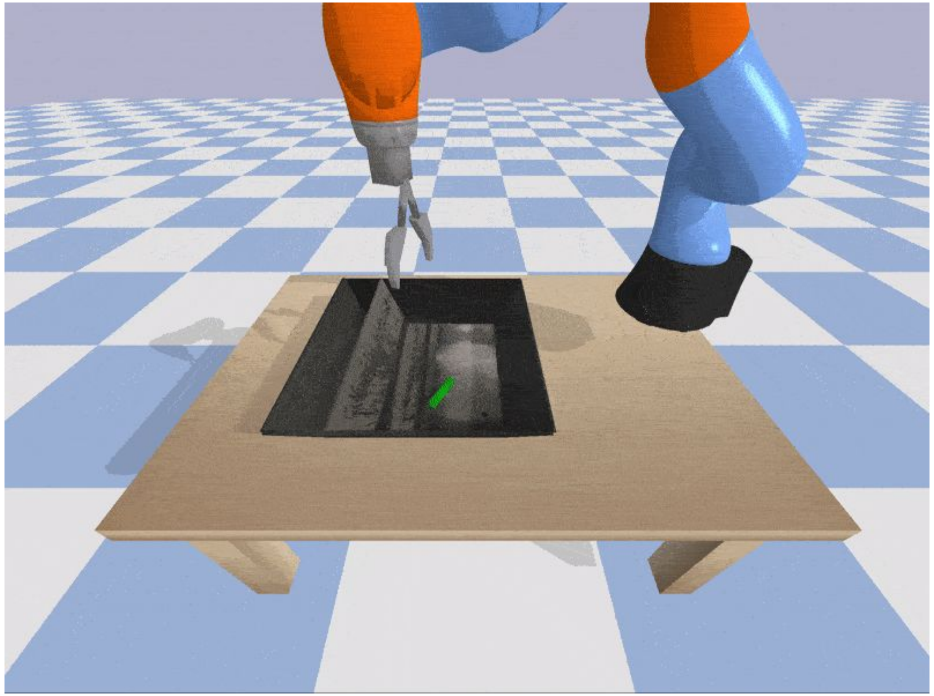
- Solve least squares regression problem: can improve sample efficiency by using priors such as sparsity.

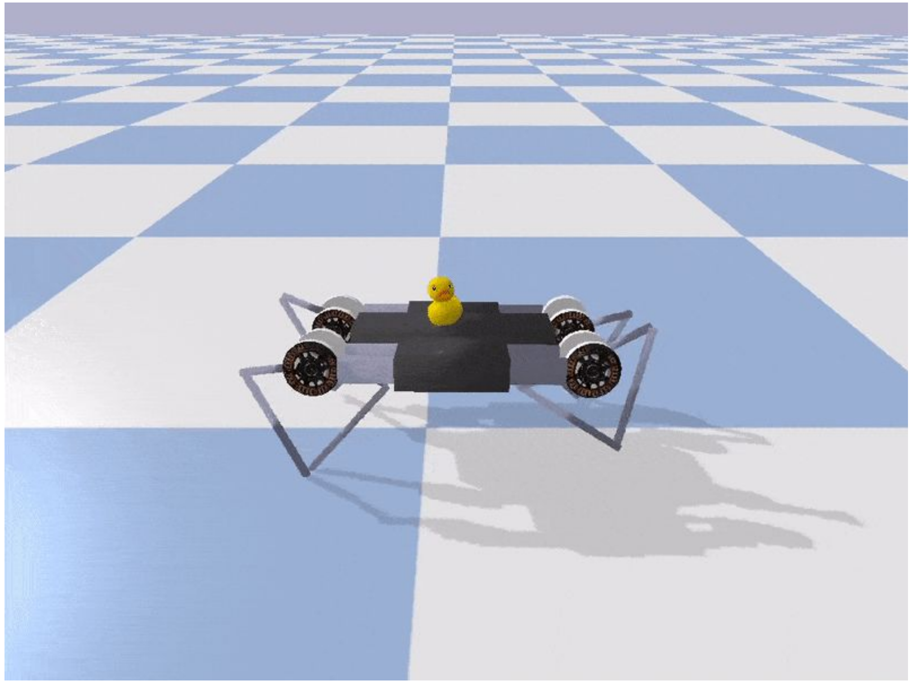
$$\arg\min_A \sum_i \|Ad_i - r_i\|_2^2 + \gamma \|A\|_{fro}^2$$

Training Neural Net Policies with DFO

Evolution Strategies as a Scalable Alternative to RL

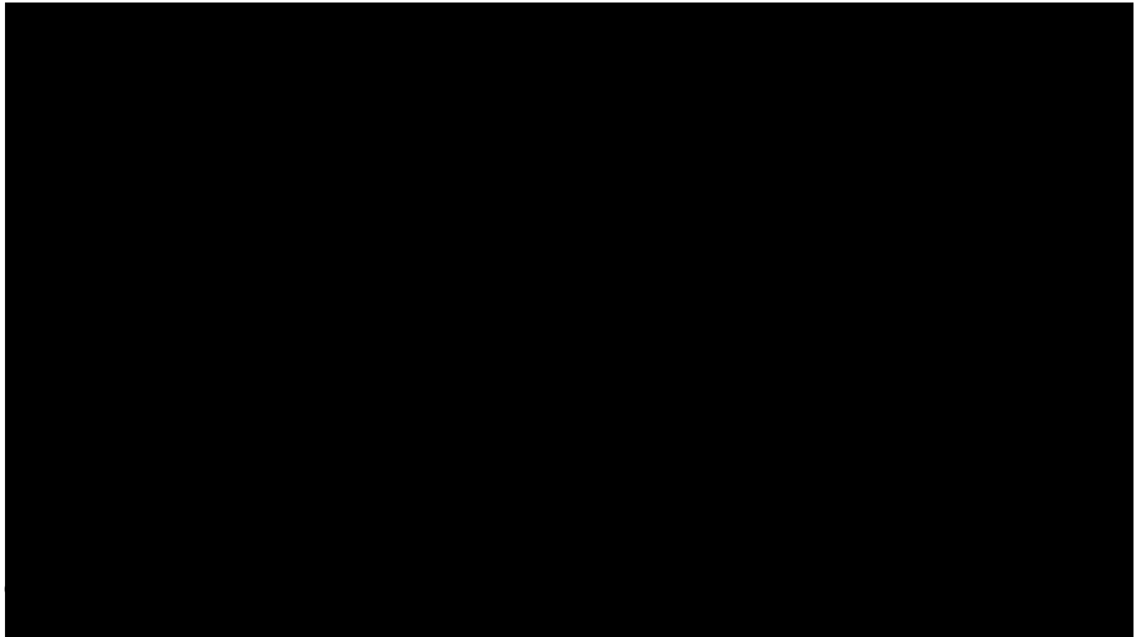
http://blog.otoro.net/2017/10/29/visual-evolution-strategies/

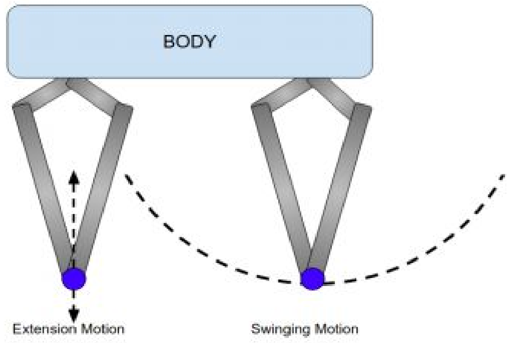




Quadruped Locomotion with DFO: Sim2Reality Transfer

- Improved Finite-difference derivative approximations (ICRA-2018)
  - Quadruped Locomotion with Speed Limits





S = A sin(t v)
E = B sin(t v + phi_leg)

Summary

- Exciting advances in Machine Learning
- Robotics is a great source of ML and Optimization Problems
- Topics we discussed
  - Imitation Learning
    - Supervised learning with care
  - Optimal Control
    - Structured QPs, Nonlinear Programming
  - Model-based Reinforcement Learning
    - Iterative Learning and Optimization
  - Derivative Free Optimization
    - Common situation when working with simulators