Deep Learning for Robotics

Sergey Levine
Real-World Experiments

**Not** accounting for uncertainty
(higher-speed collisions)
Deep Learning: End-to-end vision

standard computer vision

features (e.g. HOG)

mid-level features (e.g. DPM)
Felzenszwalb ‘08

classifier (e.g. SVM)

Krizhevsky ‘12

deep learning
perception

Action (run away)

action
Action (run away)
sensorimotor loop
“When a man throws a ball high in the air and catches it again, he behaves as if he had solved a set of differential equations in predicting the trajectory of the ball ... at some subconscious level, something functionally equivalent to the mathematical calculations is going on.”

-- Richard Dawkins

McLeod & Dienes. Do fielders know where to go to catch the ball or only how to get there? Journal of Experimental Psychology 1996, Vol. 22, No. 3, 531-543
KAIST’s DRC-HUBO opening a door

DARPA Robotics Challenge 2015
End-to-end vision

standard computer vision

features (e.g. HOG)

mid-level features (e.g. DPM)

classifier (e.g. SVM)

Felzenszwalb ’08

Krizhevsky ’12

End-to-end robotic control

standard robotic control

observations

state estimation (e.g. vision)

modeling & prediction

motion planning

low-level controller (e.g. PD)

motor torques

deep sensorimotor learning

observations

motor torques
indirect supervision
actions have consequences
Why should we care?
Why should we care?
Goals of this lecture

• Introduce formalisms of decision making
  – states, actions, time, cost
  – Markov decision processes
• Survey recent research
  – imitation learning
  – reinforcement learning
• Outstanding research challenges
  – what is easy
  – what is hard
  – where could we go next?
Contents

Imitation learning

Imitation without a human

Reinforcement learning

Research frontiers
Terminology & notation

$x_t$ – state
$o_t$ – observation
$u_t$ – action

$p_{\theta}(u_t|o_t)$ – policy

$o_t$ – observation

$x_t$ – state
Terminology & notation

\[ o_t \] - observation
\[ x_t \] - state
\[ u_t \] - action

\[ \pi_\theta(u_t | o_t) \] - policy

Markov property independent of \( x_{t-1} \)
Contents

Imitation learning

Imitation without a human

Reinforcement learning

Research frontiers
Imitation Learning

$\mathbf{o}_t \quad \pi_\theta(\mathbf{u}_t | \mathbf{o}_t) \quad \mathbf{u}_t$

Images: Bojarski et al. ‘16, NVIDIA
Does it work? No!
Does it work?  Yes!

Video: Bojarski et al. ‘16, NVIDIA
Why did that work?

Bojarski et al. '16, NVIDIA
Can we make it work more often?

stability
Learning from a stabilizing controller

\[ p(x), \quad \text{Gaussian distribution} \]

\[ \mathcal{T} \]

(more on this later)
Can we make it work more often?

\[ \pi_\theta(u_t | o_t) \]

\[ p_{\pi_\theta}(o_t) \]

\[ p_{\text{data}}(o_t) \]

can we make \( p_{\text{data}}(o_t) = p_{\pi_\theta}(o_t) \)?
Can we make it work more often?

Can we make \( p_{\text{data}}(o_t) = p_{\pi_\theta}(o_t) \)?

Idea: instead of being clever about \( p_{\pi_\theta}(o_t) \), be clever about \( p_{\text{data}}(o_t) \)!

**DAgger: Dataset Aggregation**

goal: collect training data from \( p_{\pi_\theta}(o_t) \) instead of \( p_{\text{data}}(o_t) \)

how? just run \( \pi_\theta(u_t|o_t) \)

but need labels \( u_t \)!

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

Ross et al. ‘11
DAgger Example

Ross et al. '11
What’s the problem?

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

Ross et al. ’11
Imitation learning: recap

- Usually (but not always) insufficient by itself
  - Distribution mismatch problem
- Sometimes works well
  - Hacks (e.g. left/right images)
  - Samples from a stable trajectory distribution
  - Add more on-policy data, e.g. using DAgger
Imitation learning: questions

- Distribution mismatch does not seem to be the whole story
  - Imitation often works without Dagger
  - Can we think about how stability factors in?
- Do we need to **add** data for imitation to work?
  - Can we just use existing data more cleverly?
Contents

Imitation learning

Imitation without a human

Reinforcement learning

Research frontiers
Terminology & notation

\( o_t \) – state
\( o_t \) – observation
\( u_t \) – action

\[ \pi_{\theta}(u_t | o_t) \]

\[ x_t \] – state
\[ o_t \] – observation
\[ u_t \] – action

\[ c(x_t, u_t) \] – cost function

\[ \min_{u_1, \ldots, u_T} \sum_{t=1}^{T} \log p(x_t, u_t) \text{by tiger} | u_1 f(x_t, u_T, u_{t-1}) \]
Trajectory optimization

$$\min_{u_1, \ldots, u_T} \sum_{t=1}^{T} c(x_t, u_t) \quad \text{s.t.} \quad x_t = f(x_{t-1}, u_{t-1})$$

$$\min_{u_1, \ldots, u_T} c(x_1, u_1) + c(f(x_1, u_1), u_2) + \cdots + c(f(f(\ldots)\ldots), u_T)$$

usual story: differentiate via backpropagation and optimize!

need $\frac{df}{dx_t}, \frac{df}{du_t}, \frac{dc}{dx_t}, \frac{dc}{du_t}$

in practice, it really helps to use a 2nd order method!

see differential dynamic programming (DDP) and iterative LQR (iLQR)
Probabilistic version

deterministic dynamics: $x_{t+1} = f(x_t, u_t)$

stochastic dynamics: $x_{t+1} \sim p(x_{t+1} | x_t, u_t)$

simple stochastic dynamics: $p(x_{t+1} | x_t, u_t) = \mathcal{N}(f(x_t, u_t), \Sigma)$

simple stochastic policy: $p(u_t | x_t) = \mathcal{N}(K_t x_t + k_t, \Sigma_{u_t})$

$$\min_{u_1, \ldots, u_T} \sum_{t=1}^{T} c(x_t, u_t)$$

$$\min_{K_1, k_1, \Sigma_{u_1}, \ldots, K_T, k_T, \Sigma_{u_T}} \sum_{t=1}^{T} E(x_t, u_t) \sim p(x_t, u_t) \left[ c(x_t, u_t) \right]$$

$$p(\tau) = p(x_1) \prod_{t=1}^{T} p(x_{t+1} | x_t, u_t) p(u_t | x_t)$$
Probabilistic version (in pictures)

simple stochastic dynamics: \( p(x_{t+1}|x_t, u_t) = \mathcal{N}(f(x_t, u_t), \Sigma) \)

simple stochastic policy: \( p(u_t|x_t) = \mathcal{N}(K_t x_t + k_t, \Sigma_{u_t}) \)

\[
p(\tau) = p(x_1) \prod_{t=1}^{T} p(x_{t+1}|x_t, u_t)p(u_t|x_t)
\]

\[
K_t x_t + k_t = K_t (x_t - \hat{x}_t) + \hat{u}_t
\]
DAgger without Humans

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

Ross et al. ‘11
Another problem

1. train $\pi_\theta(u_t | o_t)$ from human data $D = \{o_1, u_1, \ldots, o_N, u_N\}$
2. run $\pi_\theta(u_t | o_t)$ to get dataset $D_\pi = \{o_1, \ldots, o_M\}$
3. Ask human to label $D_\pi$ with actions $u_t$
4. Aggregate: $D \leftarrow D \cup D_\pi$
PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(u_t|o_t)$ from human data $D = \{o_1, u_1, \ldots, o_N, u_N\}$
2. run $\hat{\pi}_\phi(u_t|o_t)$ to get dataset $D_\pi = \{o_1, \ldots, o_M\}$
3. Ask computer to label $D_\pi$ with actions $u_t$
4. Aggregate: $D \leftarrow D \cup D_\pi$

**Simple stochastic policy:**

$$\hat{\pi}(u_t|x_t) = \mathcal{N}(K_t x_t + k_t, \Sigma_{u_t})$$

$$\hat{\pi}(u_t|x_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(x_{t'}, u_{t'})] + \lambda D_{KL}(\hat{\pi}(u_t|x_t) || \pi_\theta(u_t|o_t))$$
PLATO: Policy Learning with Adaptive Trajectory Optimization

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

**simple** stochastic policy: $\hat{\pi}(u_t|x_t) = \mathcal{N}(Ktx_t + k_t, \Sigma_{u_t})$

$$\hat{\pi}(u_t|x_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^T E_{\hat{\pi}}[c(x_{t'}, u_{t'})] + \lambda D_{KL}(\hat{\pi}(u_t|x_t)\|\pi_\theta(u_t|o_t))$$

---

Kahn, Zhang, Levine, Abbeel '16
PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(u_t|o_t)$ from human data $D = \{o_1, u_1, \ldots, o_N, u_N\}$
2. run $\hat{\pi}(u_t|o_t)$ to get dataset $D_\pi = \{o_1, \ldots, o_M\}$
3. Ask computer to label $D_\pi$ with actions $u_t$
4. Aggregate: $D \leftarrow D \cup D_\pi$

**simple** stochastic policy: $\hat{\pi}(u_t|x_t) = \mathcal{N}(K_t x_t + k_t, \Sigma_{u_t})$

$$\hat{\pi}(u_t|x_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(x_{t'}, u_{t'})] + \lambda D_{KL}(\hat{\pi}(u_t|x_t)\|\pi_\theta(u_t|o_t))$$

Kahn, Zhang, Levine, Abbeel ‘16
PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(u_t|o_t)$ from human data $D = \{o_1, u_1, \ldots, o_N, u_N\}$
2. run $\hat{\pi}(u_t|o_t)$ to get dataset $D_\pi = \{o_1, \ldots, o_M\}$
3. Ask computer to label $D_\pi$ with actions $u_t$
4. Aggregate: $D \leftarrow D \cup D_\pi$

**simple** stochastic policy: $\hat{\pi}(u_t|x_t) = \mathcal{N}(K_tx_t + k_t, \Sigma_{u_t})$

$$\hat{\pi}(u_t|x_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(x_{t'}, u_{t'})] + \lambda D_{KL}(\hat{\pi}(u_t|x_t) || \pi_\theta(u_t|o_t))$$
PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(u_t|o_t)$ from human data $D = \{o_1, u_1, \ldots, o_N, u_N\}$
2. run $\hat{\pi}(u_t|o_t)$ to get dataset $D_\pi = \{o_1, \ldots, o_M\}$
3. Ask computer to label $D_\pi$ with actions $u_t$
4. Aggregate: $D \leftarrow D \cup D_\pi$

**Simple stochastic policy:** $\hat{\pi}(u_t|x_t) = \mathcal{N}(K_tx_t + k_t, \Sigma_{u_t})$

$$\hat{\pi}(u_t|x_t) = \arg\min_\hat{\pi} \sum_{t'=t}^T E_{\hat{\pi}}[c(x_{t'}, u_{t'})] + \lambda D_{KL}(\hat{\pi}(u_t|x_t)\|\pi_\theta(u_t|o_t))$$
PLATO: Policy Learning with Adaptive Trajectory Optimization

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

**simple** stochastic policy: $\hat{\pi}(u_t|x_t) = \mathcal{N}(K_t x_t + k_t, \Sigma_u_t)$

$$\hat{\pi}(u_t|x_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^T E_{\hat{\pi}}[c(x_{t'}, u_{t'})] + \lambda D_{KL}(\hat{\pi}(u_t|x_t) \| \pi_\theta(u_t|o_t))$$
PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(u_t|o_t)$ from human data $D = \{o_1, u_1, \ldots, o_N, u_N\}$
2. run $\hat{\pi}(u_t|o_t)$ to get dataset $D_\pi = \{o_1, \ldots, o_M\}$
3. Ask computer to label $D_\pi$ with actions $u_t$
4. Aggregate: $D \leftarrow D \cup D_\pi$

**Simple stochastic policy:**

$$\hat{\pi}(u_t|x_t) = N(Ktx_t + k_t, \Sigma_u)$$

$$\hat{\pi}(u_t|x_t) = \arg\min_{\hat{\pi}} \sum_{t' = t}^T E_{\hat{\pi}}[c(x_{t'}, u_{t'})] + \lambda D_{KL}(\hat{\pi}(u_t|x_t) \| \pi_\theta(u_t|o_t))$$

**Replanning = Model Predictive Control (MPC)**

$\pi_\theta(u_t|o_t)$ -- control from images

$\hat{\pi}(u_t|x_t)$ -- control from states

Kahn, Zhang, Levine, Abbeel ’16
PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_{\theta}(u_t|o_t)$ from human data $D = \{o_1, u_1, \ldots, o_N, u_N\}$
2. run $\hat{\pi}(u_t|o_t)$ to get dataset $D_\pi = \{o_1, \ldots, o_M\}$
3. Ask computer to label $D_\pi$ with actions $u_t$
4. Aggregate: $D \leftarrow D \cup D_\pi$

**simple** stochastic policy: $\hat{\pi}(u_t|x_t) = N(K_tx_t + k_t, \Sigma_{u_t})$

$$\hat{\pi}(u_t|x_t) = \text{arg min}_{\hat{\pi}} \sum_{t' = t}^{T} E_{\hat{\pi}}[c(x_{t'}, u_{t'})] + \lambda D_{KL}(\hat{\pi}(u_t|x_t) || \pi_{\theta}(u_t|o_t))$$

- $o_2$? – unknown!
- $p(x_{t+1}|x_t, u_t)$ – known
- $p(o_t|x_t)$ – unknown
PLATO: Policy Learning with Adaptive Trajectory Optimization

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

**simple** stochastic policy: $\hat{\pi}(u_t|x_t) = \mathcal{N}(K_t x_t + k_t, \Sigma_{u_t})$

$$\hat{\pi}(u_t|x_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^T E_{\hat{\pi}}[c(x_{t'}, u_{t'})] + \lambda D_{KL}(\hat{\pi}(u_t|x_t)\|\pi_\theta(u_t|o_t))$$
PLATO: Policy Learning with Adaptive Trajectory Optimization

\[
\hat{\pi}(u_t|x_t) = \arg \min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(x_{t'}, u_{t'})] + \lambda D_{KL}(\hat{\pi}(u_t|x_t) \| \pi_\theta(u_t|o_t))
\]

avoids high cost!

input substitution trick
need state at training time
but not at test time!
PLATO: Policy Learning with Adaptive Trajectory Optimization

Objective: fly through forest at 2m/s
Main sensor: 1d laser
Beyond driving & flying

\[ \hat{\pi}(u_1|o_1) \]

\[ \pi_\theta(u_1|o_1) \]
Trajectory Optimization with Unknown Dynamics

\[ p(\tau) - \text{Gaussian trajectory distribution} \]
\[ p(u_t|x_t) - \text{time-varying linear-Gaussian controller} \]
\[ \text{can execute on the robot!} \]

\[
\min_{u_1,\ldots,u_T} \sum_{t=1}^{T} c(x_t, u_t) \quad \text{s.t.} \quad x_t = f(x_{t-1}, u_{t-1})
\]

\[
\min_{u_1,\ldots,u_T} c(x_1, u_1) + c(f(x_1, u_1), u_2) + \cdots + c(f(f(\ldots), \ldots), u_T)
\]

\[ \text{need } \frac{df}{dx_t}, \frac{df}{du_t}, \frac{dc}{dx_t}, \frac{dc}{du_t} \]

[L. et al. NIPS '14]
Trajectory Optimization with Unknown Dynamics

\[ p(\tau) \] – Gaussian trajectory distribution
\[ p(u_t | x_t) \] – time-varying linear-Gaussian controller

can execute on the robot!

run \( p(u_t | x_t) \) on robot
collect \( \mathcal{D} = \{\tau_i\} \)

next iteration

\[ p(x_{t+1} | x_t, u_t) = \mathcal{N}(f(x_t, u_t), \Sigma) \]
\[ f(x_t, u_t) \approx A_t x_t + B_t u_t \]
\[ A_t = \frac{df}{dx_t} \quad B_t = \frac{df}{du_t} \]

fit dynamics \( p(x_{t+1} | x_t, u_t) \)

improve \( p(u_t | x_t) \)

\[
\min_{p(\tau)} E_p[c(\tau)] \quad \text{s.t.} \quad D_{KL}(p(\tau) \| \tilde{p}(\tau)) \leq \epsilon
\]

[L. et al. NIPS ‘14]
Learning on PR2

10x real time  autonomous execution

[L. et al. ICRA ‘15]
Combining with Policy Learning
\[
\min_{\theta} E_{\pi_{\theta}}[c(\tau)]
\]

\[
\min_{\theta, p(\tau)} E_p[c(\tau)]
\]

subject to:

\[
\pi_{\theta}(u_t | o(x_t)) = p(u_t | x_t) \quad \forall t, x_t, u_t
\]

\[
E_{p(x_t)}[D_{KL}(\pi_{\theta} || p(u_t | x_t))] = 0 \quad \forall t
\]

L. et al. ICML '14 (dual descent)
can also use BADMM (L. et al.'15)

\[
\mathcal{L}(\theta, p, \lambda) = E_p[c(\tau)] + \sum_{t=1}^T \lambda_t D_t(\pi_{\theta}, p)
\]

optimize \( \mathcal{L}(\theta, p, \lambda) \) w.r.t. \( p(\tau) \)

optimize \( \mathcal{L}(\theta, p, \lambda) \) w.r.t. \( \theta \)

update \( \lambda \) with subgradient descent:

\[
\lambda_t \leftarrow \lambda_t + \eta D_t(\pi_{\theta}, p)
\]
Guided Policy Search

trajectory-centric RL

supervised learning
run $p(u_t | x_t)$ on robot
collect $\mathcal{D} = \{\tau_i\}$

fit dynamics $p(x_{t+1} | x_t, u_t)$

improve $p(u_t | x_t)$

next iteration

train $\pi_\theta(u_t | o_t)$

[see L. et al. NIPS '14 for details]
training time \(\mathbf{x}_t \rightarrow \mathbf{u}_t\)

test time \(\mathbf{o}_t \rightarrow \mathbf{u}_t\)

L.*, Finn*, Darrell, Abbeel ‘16
~ 92,000 parameters
Experimental Tasks

Learned Visuomotor Policy: Shape sorting cube
Imitating optimal control: questions

- Any difference from standard imitation learning?
  - Can change behavior of the “teacher” programmatically

- Can the policy help optimal control (rather than just the other way around?)
Can we avoid dynamics completely?
A simple reinforcement learning algorithm

\[ J(\theta) \]

\[ \min_{\theta} \mathbb{E}_{\tau \sim p(\tau)} \left[ \sum_{t=1}^{T} \log p(x_t, u_t | x_{t-1}, u_{t-1}) \right] \]

\[ p(\tau) = p(x_1, u_1, \ldots, x_T, u_T) = p(x_1) \prod_{t=1}^{T} p(x_{t+1} | x_t, u_t) \pi_\theta(u_t | x_t) \]

\[ \nabla_{\theta} J(\theta) \]

\[ \nabla_{\theta} p(\tau) \]

\[ \nabla_{\theta} J(\theta) = \int p(\tau) [\nabla_{\theta} \log p(\tau)] c(\tau) d\tau \]

\[ \nabla_{\theta} \log p(\tau) = \left[ \sum_{t=1}^{T} \log p(x_t, u_t | x_{t-1}, u_{t-1}) \right] \log p(x_{t+1} | x_t, u_t) + \log \pi_\theta(u_t | x_t) \]
REINFORCE
likelihood ratio policy gradient

\[ \nabla_\theta J(\theta) = E \left[ \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(u_t|x_t) \right) \left( \sum_{t=1}^{T} c(x_t, u_t) \right) \right] \]

example: \( \pi_\theta(u_t, x_t) = \mathcal{N}(f_{\text{neural network}}(x_t); \Sigma) \)

\[ \log \pi_\theta(u_t|x_t) = -\frac{1}{2} \| f(x_t) - u_t \|_\Sigma^2 + \text{const} \]

\[ \nabla_\theta \log \pi_\theta(u_t|x_t) = -\frac{1}{2} \Sigma^{-1}(f(x_t) - u_t) \frac{df}{d\theta} \]

how? just backpropagate \(-\frac{1}{2} \Sigma^{-1}(f(x_t) - u_t)\)

REINFORCE algorithm:

1. sample \( \{ \tau^i \} \) from \( \pi_\theta(u_t|x_t) \) (run it on the robot)
2. \( \nabla_\theta J(\theta) \approx \sum_i \left( \sum_t \nabla_\theta \log \pi_\theta(u_t^i|x_t^i) \right) \left( \sum_t c(x_t^i, u_t^i) \right) \)
3. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)

Williams '92
What the heck did we just do?

\[
\nabla_\theta J(\theta) = E \left[ \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t) \right) \left( \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \right) b \right]
\]

\[b = E \left[ \sum_{t} c(\mathbf{x}_t, \mathbf{u}_t) \right]\]

one more piece…

\[E[\nabla \log p(y)b]\]
Policy gradient challenges

\[ \nabla_\theta J(\theta) = E \left[ \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(u_t|x_t) \right) \left( \sum_{t=1}^{T} c(x_t, u_t) - b \right) \right] \]

- High variance in gradient estimate
  - Smarter baselines
- Poor conditioning
  - Use higher order methods (see: natural gradient)
- Very hard to choose step size
  - Trust region policy optimization (TRPO)

Schulman, L., Moritz, Jordan, Abbeel ‘15
Value functions

\[ \nabla_\theta J(\theta) = \sum_{t=1}^{T} \left( E \left[ \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(u_t|x_t) \right] \right) \left[ \sum_{t=1}^{T} \log \pi_\theta(u_t+1|x_{t+1}) \right] \]

total remaining cost of executing \( \pi_\theta(u_t|x_t) \)

\[ Q^\pi(x_t, u_t) \]

\( Q^\pi(x_t, u_t) \) – total cost of running \( \pi \) after taking action \( u_t \) in state \( x_t \)

\[ Q^\pi(x_t, u_t) = c(x_t, u_t) + E_{x_{t+1} \sim p(x_{t+1}|x_t, u_t)}[V^\pi(x_{t+1})] \]

\( V^\pi(x_t) \) – total cost of running \( \pi \) from state \( x_t \)

\[ Q^\pi(x_t, u_t) = c(x_t, u_t) + E_{x_{t+1} \sim p(x_{t+1}|x_t, u_t)}[E_{u_{t+1} \sim \pi_\theta(u_{t+1}|x_{t+1})}[Q^\pi(x_{t+1}, u_{t+1})]] \]
Value functions

\[ \nabla_{\theta} J(\theta) = \sum_{t=1}^{T} E \left[ \nabla_{\theta} \log \pi_{\theta}(u_t | x_t) ] \hat{Q}_{\phi}^\pi \left[ \sum_{t' = t}^{T} u_t(x_{t'}, u_{t'}) \right] \right] \]

\[ \hat{Q}_{\phi}^\pi (x_t, u_t) \approx E \left[ \sum_{t' = t}^{T} c(x_{t'}, u_{t'}) \right] \]

equation: \[ \hat{Q}_{\phi}^\pi (x_t, u_t) \] is a neural network, trained via regression

\[ \phi \leftarrow \arg \min_{\phi} \sum_{i=1}^{N} \sum_{t=1}^{T} \left\| \hat{Q}_{\phi}^\pi (x_t^i, u_t^i) - \left[ \sum_{t'=t}^{T} c(x_{t'}^i, u_{t'}^i) u_t^i \hat{Q}_{\phi}^\pi (x_{t'+1}^i, u_{t'+1}^i) \right] \right\|^2 \]

Policy gradient with value function approximation:

1. sample \( \{ \tau^i \} \) from \( \pi_{\theta}(u_t | x_t) \) (run it on the robot)
2. Use samples \( \{ \tau^i \} \) to fit \( \hat{Q}_{\phi}^\pi \)
3. \( \nabla_{\theta} J(\theta) \approx \sum_i \sum_t \nabla_{\theta} \log \pi_{\theta}(u_t^i | x_t^i) \hat{Q}_{\phi}^\pi (x_t^i, u_t^i) \)
4. \( \theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta) \)
Value functions challenges

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E \left[ \nabla_\theta \log \pi_\theta(u_t|x_t) \hat{Q}^\pi_\phi(x_t, u_t) \right]$$

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E \left[ \nabla_\theta \log \pi_\theta(u_t|x_t) E \left[ \sum_{t'=t}^{T} c(x_{t'}, u_{t'}) \right] \right]$$

- The usual problems with policy gradient
  - Poor conditioning (use natural gradient)
  - Hard to choose step size (use TRPO)
- Bias/variance tradeoff
  - Combine Monte Carlo and function approximation: Generalized Advantage Estimation (GAE)
- Instability/overfitting
  - Limit how much the value function estimate changes per iteration

Schulman, Moritz, L. Jordan, Abbeel ’16
Generalized advantage estimation
Online actor-critic methods

1. Sample a decision $\pi_{\theta}(\text{next step})$ with on the $\theta(\text{old})$ add to buffer $D$
2. Sample a batch to fit $Q^\pi_\phi$ from $D$, fit $Q^\pi_\phi$
3. $\nabla_{\theta} J(\theta) \approx \sum_i \sum_t \log \pi^i_t(\mathbf{u}_t^i|\mathbf{x}_t^i) \nabla_{\theta} \log Q^\pi_\phi(\mathbf{x}_t^i, \mathbf{u}_t^i)$
4. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$

Deep deterministic policy gradient (DDPG)

Continuous Control with Deep Reinforcement Learning (Lillicrap et al. ‘15)
Just the Q function?

\[ \nabla_\theta J(\theta) = \sum_{t=1}^{T} E \left[ \nabla_\theta \log \pi_\theta(u_t | x_t) \hat{Q}_\phi(x_t, u_t) \right] \]

\[ \phi \leftarrow \arg \min_\phi \sum_{i=1}^{N} \sum_{t=1}^{T} \left\| \hat{Q}_\phi(x_t^i, u_t^i) - \left[ c(x_t^i, u_t^i) + \gamma \hat{Q}_{\phi_{\text{old}}} \left. \hat{Q}_\phi(x_{t+1}^i, u_{t+1}^i) \right| u_{t+1}^i \right\|^2 \right\|^2 \]

\[ \pi(u_t | x_t) \propto \exp(-\hat{Q}_\phi(x_t, u_t)) \quad \pi(u_t | x_t) \propto \epsilon + \delta(u_t = \arg \min_{u_t} \hat{Q}_\phi(x_t, u_t)) \]

Q-learning for deep RL

1. make one decision (time step) with \( u \sim \pi_\theta(u | x) \), add to buffer \( \mathcal{D} \)
2. sample minibatch \( \{ x^i, u^i \} \) from \( \mathcal{D} \), fit \( \hat{Q}_\phi \)
3. \( \nabla_\theta J(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla \log \pi_\theta(u_t^i | x_t^i) \hat{Q}_\phi(x_t^i, u_t^i) \hat{Q}_\phi(x_t^i, u_t^i) \propto \exp(-\hat{Q}_\phi(x_t, u_t)) \)
4. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)
Discrete Q-learning
Continuous Q-learning

\[ \phi \leftarrow \arg\min_{\phi} \sum_{i=1}^{N} \sum_{t=1}^{T} \left\| \hat{Q}_{\phi}(x_t^i, u_t^i) - \left[ c(x_t^i, u_t^i) + \gamma \min_{u_{t+1}} \hat{Q}_{\phi_{old}}(x_t^{i}, u_{t+1}) \right] \right\|^2 \]

\[
\hat{Q}_{\phi}(x, u) = \frac{1}{2} (u - \mu(x))^T P(x) (u - \mu(x)) + V(x)
\]
## Normalized Advantage Functions

<table>
<thead>
<tr>
<th>Domains</th>
<th>Random</th>
<th>DDPG episode</th>
<th>NAF episode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cartpole</td>
<td>-2.1</td>
<td>-0.601 420</td>
<td>-0.604 190</td>
</tr>
<tr>
<td>Reacher</td>
<td>-2.3</td>
<td>-0.509 1370</td>
<td>-0.331 1260</td>
</tr>
<tr>
<td>Peg</td>
<td>-11</td>
<td>-0.950 690</td>
<td>-0.438 130</td>
</tr>
<tr>
<td>Gripper</td>
<td>-29</td>
<td>1.03 2420</td>
<td>1.81 1920</td>
</tr>
<tr>
<td>GripperM</td>
<td>-90</td>
<td>-20.2 1350</td>
<td>-12.4 730</td>
</tr>
<tr>
<td>Canada2d</td>
<td>-12</td>
<td>-4.64 1040</td>
<td>-4.21 900</td>
</tr>
<tr>
<td>Cheetah</td>
<td>-0.3</td>
<td>8.23 1590</td>
<td>7.91 2390</td>
</tr>
<tr>
<td>Swimmer6</td>
<td>-325</td>
<td>-174 220</td>
<td>-172 190</td>
</tr>
<tr>
<td>Ant</td>
<td>-4.8</td>
<td>-2.54 2450</td>
<td>-2.58 1350</td>
</tr>
<tr>
<td>Walker2d</td>
<td>0.3</td>
<td>2.96 850</td>
<td>1.85 1530</td>
</tr>
</tbody>
</table>

*Random, DDPG, NAF policies: final rewards and episodes to converge*
Policy Learning with Multiple Robots: Deep RL with NAF

$$Q(x, u|\theta_Q) = A(x, u|\theta^A) + V(x|\theta^V)$$

$$A(x, u|\theta^A) = -\frac{1}{2}(u - \mu(x|\theta^\mu))^T P(x|\theta^P) (u - \mu(x|\theta^\mu))$$

Shane Gu, Ethan Holly, Tim Lillicrap
Deep RL with Policy Gradients

- Unbiased but high-variance gradient
- Stable
- Requires many samples
- Example: TRPO [Schulman et al. ‘15]

\[
\nabla_\theta J(\theta) = E_{\pi_\theta} [\nabla_\theta \log \pi_\theta (u_t|x_t) \hat{Q}(x_t, u_t)]
\]

\[
\hat{Q}(x_t, u_t) \approx \sum_{t'=t}^{\infty} \gamma^{t-t'} r(x_t, u_t)
\]
Deep RL with Off-Policy Q-Function Critic

- **Low-variance but biased** gradient
- **Much more efficient** (because off-policy)
- **Much less stable** (because biased)
- Example: DDPG [Lillicrap et al. ‘16]

\[
Q_w \leftarrow \min_w E[(Q_w(x_t, u_t) - (r(x_t, u_t) + \gamma Q_w(x_{t+1}, \pi_\theta(x_t)))^2]
\]

\[
\nabla_\theta J(\theta) = E[\nabla_{u_t} Q_w(x_t, \pi_\theta(x_t)) \nabla_\theta \pi_\theta(x_t)]
\]
Improving Efficiency & Stability with Q-Prop

Policy gradient:
\[ \nabla_{\theta} J(\theta) = E_{\pi_\theta}[\nabla_{\theta} \log \pi_\theta(u_t|x_t)\hat{Q}(x_t,u_t)] \]

Q-Prop:
\[ \nabla_{\theta} J(\theta) = E_{\pi_\theta}(x_t,u_t)[\nabla_{\theta} \log \pi_\theta(u_t|x_t)(\hat{Q}(x_t,u_t) - \bar{Q}(x_t,u_t)) + \nabla_{u_t} Q(x_t,\mu_\theta(x_t))(u_t - \mu_\theta(x_t))] \]

Q-function critic:
\[ \nabla_{\theta} J(\theta) = E[\nabla_{u_t} Q_w(x_t,\mu_\theta(x_t))\nabla_{\theta} \mu_\theta(x_t)] \]

- **Unbiased** gradient, stable
- **Efficient** (uses off-policy samples)
- Critic comes from off-policy data
- Gradient comes from on-policy data
- Automatic variance-based adjustment
Comparisons

- Works with smaller batches than TRPO
- More efficient than TRPO
- More stable than DDPG with respect to hyperparameters
  - Likely responsible for the better performance on harder task
Sample complexity

- Deep reinforcement learning is very data-hungry
  - DQN: about 100 hours to learn Breakout
  - GAE: about 50 hours to learn to walk
  - DDPG/NAF: 4-5 hours to learn basic manipulation, walking
- Model-based methods are more efficient
  - Time-varying linear models: 3 minutes for real world manipulation
  - GPS with vision: 30-40 minutes for real world visuomotor policies

<table>
<thead>
<tr>
<th>task</th>
<th>trajectory pretraining</th>
<th>end-to-end training</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>coat hanger</td>
<td>120</td>
<td>36</td>
<td>156</td>
</tr>
<tr>
<td>shape cube</td>
<td>90</td>
<td>81</td>
<td>171</td>
</tr>
<tr>
<td>toy hammer</td>
<td>150</td>
<td>90</td>
<td>240</td>
</tr>
<tr>
<td>bottle cap</td>
<td>180</td>
<td>108</td>
<td>288</td>
</tr>
</tbody>
</table>
Reinforcement learning tradeoffs

• Reinforcement learning (for the purpose of this slide) = *model-free* RL

• Fewer assumptions
  – Don’t need to model dynamics
  – Don’t (in general) need state definition, only observations
  – Fully general stochastic environments

• Much slower (model-based acceleration?)

• Hard to stabilize
  – Few convergence results with neural networks
Contents

Imitation learning

Imitation without a human

Reinforcement learning

Research frontiers
ingredients for success in learning:

<table>
<thead>
<tr>
<th>Supervised Learning:</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ computation</td>
</tr>
<tr>
<td>✓ algorithms</td>
</tr>
<tr>
<td>✓ data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learning Sensorimotor Skills:</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ computation</td>
</tr>
<tr>
<td>~ algorithms</td>
</tr>
<tr>
<td>? data</td>
</tr>
</tbody>
</table>

L., Pastor, Krizhevsky, Quillen ‘16
Grasping with Learned Hand-Eye Coordination

- 800,000 grasp attempts for training (3,000 robot-hours)
- monocular camera (no depth)
- 2-5 Hz update
- no prior knowledge

L., Pastor, Krizhevsky, Quillen ‘16
Using Grasp Success Prediction

L., Pastor, Krizhevsky, Quillen ‘16
Open-Loop vs. Closed-Loop Grasping

open-loop grasping

failure rate: 33.7%

open loop
1x real time

depth + segmentation
failure rate: 35%

Pinto & Gupta, 2015

closed-loop grasping

failure rate: 17.5%

our method
1x real time

L., Pastor, Krizhevsky, Quillen ‘16
Grasping Experiments
Learning what Success Means

can we learn the cost with visual features?

\[ c(x, u) = w_1 f_{\text{target}}(x) + w_2 f_{\text{torque}}(u) \]
Learning what Success Means

with C. Finn, P. Abbeel
Learning what Success Means

c(x, u) = ???
Challenges & Frontiers

• Algorithms
  – Sample complexity
  – Safety
  – Scalability

• Supervision
  – Automatically evaluate success
  – Learn cost functions

\[ c(x, u) = ??? \]
Acknowledgements

Greg Kahn  Tianhao Zhang  Chelsea Finn  Trevor Darrell  Pieter Abbeel

John Schulman  Philipp Moritz  Michael Jordan

Shane Gu  Peter Pastor  Alex Krizhevsky  Deirdre Quillen