NIPS 2016

Ni Lao
2017/1/31
NIPS 2016

Largest ML conference
Barcelona, Spain
6000 attendees
Dec 5 tutorials, posters
Dec 6,7 presentations, posters
Dec 8 presentations, symposiums
Dec 9,10 workshops
AI is getting popular

- A lot of industry presence
  - Facebook, Microsoft, Amazon, NVIDIA, most of Google Brain and most of DeepMind
  - Automotive, financial, e-commerce, and all kind of companies looking to grow their ML groups
  - Startup founders and CEOs of AI companies walking around

Credit http://blog.evjang.com/2017/01/nips2016.html
The venue for big announcements

- (2015) **OpenAI** announced its existence
- **OpenAI** released their Universe platform for developing a reinforcement learning agent
- **DeepMind** open-sourced their platform for experimenting with RL agent
- **Uber** announced they had acqui-hired Geometric Intelligence in order to expedite their own AI efforts
- **Apple** announced that it will publish its research and make source-code available
Plan

- Keynote
- Generative Adversarial Nets (GAN)
- Deep RL
- Meta-Learning
Keynote

-- Yann Lecun
LeCun’s Cake

- learning needs to be mostly unsupervised
- for good data representations

“Pure” Reinforcement Learning (cherry)
- The machine predicts a scalar reward given once in a while.
- A few bits for some samples

Supervised Learning (icing)
- The machine predicts a category or a few numbers for each input
- Predicting human-supplied data
- 10→10,000 bits per sample

Unsupervised/Predictive Learning (cake)
- The machine predicts any part of its input for any observed part.
- Predicts future frames in videos
- Millions of bits per sample

(Yes, I know, this picture is slightly offensive to RL folks. But I’ll make it up)
He Really Meant End-to-End Training
Learning Agents

- Inspiration from human brain

- The essence of intelligence is the ability to predict
Generative Adversarial Nets (GAN)

-- a tutorial by Ian Goodfellow

“the biggest breakthrough in Machine Learning in the last 1-2 decades.”

-- Yann Lecun
Why study generative models?

- Realistic generation tasks
- Semi-supervised learning

“What I cannot create, I do not understand.”
—Richard Feynman

Human: what is moral?
Machine: what empowered humanity, what intellectual the essence is.
Human: what is immoral?
Machine: the fact that you have a child.
Human: what is morality?
Machine: what is altruism?
Human: what is the definition of altruism?
Machine: if you don’t believe in god, then you don’t know.
Human: ok... so what is the definition of morality?
Machine: well, the truth is, you’re not a believer in god almighty.
Human: tell me the definition of morality, i am quite upset now!
Machine: i’m not ashamed of being a philosopher!

[Vinyals, Le, 2015]
Super-Resolution

bicubic (21.59dB/0.6423)

SRResNet (23.53dB/0.7832)

SRGAN (21.15dB/0.6868)

original

Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]
Art

Figure 2: Images that combine the content of a photograph with the style of several well-known artworks. The images were created by finding an image that simultaneously matches the content representation of the photograph and the style representation of the artwork (see Methods).

[Gatys, Ecker, Bethge, 2015]
Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.
Taxonomy of Generative Models

... 

Maximum Likelihood 

Explicit density 

Tractable density 
- Fully visible belief nets 
- NADE 
- MADE 
- PixelRNN

Approximate density 
- Variational
- Markov Chain 

Implicit density 

Markov Chain 

Direct GAN 

GSN 

Real / Fake 

D 

Variational autoencoder 
Boltzmann machine 

X 

Z 

(Goodfellow 2016)
Minimax Game

\[ J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log (1 - D(G(z))) \]

\[ J^{(G)} = -J^{(D)} \]

- Equilibrium is a saddle point of the discriminator loss
- Resembles Jensen-Shannon divergence
- Generator minimizes the log-probability of the discriminator being correct
GAN as a way of regularization

- Less incentive to fit individual data points
Deep Convolutional (DC) GAN

[Radford+ 2016]
Deep Convolutional (DC) GAN

Table 2: SVHN classification with 1000 labels

<table>
<thead>
<tr>
<th>Model</th>
<th>error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN</td>
<td>77.93%</td>
</tr>
<tr>
<td>TSVM</td>
<td>66.55%</td>
</tr>
<tr>
<td>M1+KNN</td>
<td>65.63%</td>
</tr>
<tr>
<td>M1+TSVM</td>
<td>54.33%</td>
</tr>
<tr>
<td>M1+M2</td>
<td>36.02%</td>
</tr>
<tr>
<td>SWWAE without dropout</td>
<td>27.83%</td>
</tr>
<tr>
<td>SWWAE with dropout</td>
<td>23.56%</td>
</tr>
<tr>
<td>DCGAN (ours) + L2-SVM</td>
<td>22.48%</td>
</tr>
<tr>
<td>Supervised CNN with the same architecture</td>
<td>28.87% (validation)</td>
</tr>
</tbody>
</table>
Deep RL

-- tutorials by Pieter Abbeel and John Schulman
Reinforcement Learning

- Any ML problem can be formulated as a RL problem
Policy Optimization

- Consider control policy parameterized by parameter vector $\theta$

$$\max_\theta \mathbb{E} \left[ \sum_{t=0}^{H} R(s_t) | \pi_\theta \right]$$

- Often stochastic policy class (smooths out the problem):

$$\pi_\theta(u|s) : \text{probability of action } u \text{ in state } s$$
A relatively new field with recent successes

Kohl and Stone, 2004
Ng et al, 2004
Tedrake et al, 2005
Kober and Peters, 2009

Mnih et al, 2015 (A3C)
Silver et al, 2014 (DPG)
Lillicrap et al, 2015 (DDPG)
Schulman et al, 2016 (TRPO + GAE)
Levine*, Finn*, et al, 2016 (GPS)
Silver*, Huang*, et al, 2016 (AlphaGo**)
The RL landscape

- Simple
- Stable
- Data efficient
Cross-Entropy Method

\[
\max_{\theta} \ U(\theta) = \max_{\theta} \ E\left[\sum_{t=0}^{H} R(s_t) | \pi_\theta \right]
\]

- Can work surprisingly well
- not data efficient

```
CEM:
for iter i = 1, 2, ...
    for population member e = 1, 2, ...
        sample \( \theta^{(e)} \sim P_{\mu^{(i)}}(\theta) \)
        execute roll-outs under \( \pi_{\theta^{(e)}} \)
        store \((\theta^{(e)}, U(e))\)
    endfor
\mu^{(i+1)} = \arg \max_{\mu} \sum_{e} \log P_{\mu}(\theta^{(e)})
    where \( \tilde{e} \) indexes over top \( p \) %
endfor
```

Table 1: Average Tetris Scores of Various Algorithms.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean Score</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonreinforcement learning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hand-coded</td>
<td>631,167</td>
<td>Dellacherie (Fahey, 2003)</td>
</tr>
<tr>
<td>Genetic algorithm</td>
<td>586,103</td>
<td>(Böhm et al., 2004)</td>
</tr>
<tr>
<td>Reinforcement learning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relational reinforcement</td>
<td>\approx 50</td>
<td>Ramon and Driessens (2004)</td>
</tr>
<tr>
<td>learning+kernel-based regression</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy iteration</td>
<td>3183</td>
<td>Bertsekas and Tsitsiklis (1996)</td>
</tr>
<tr>
<td>Least squares policy iteration</td>
<td>&lt;3000</td>
<td>Lagoudakis, Parr, and Littman (2002)</td>
</tr>
<tr>
<td>Linear programming + Bootstrap</td>
<td>4274</td>
<td>Farias and van Roy (2006)</td>
</tr>
<tr>
<td>Natural policy gradient</td>
<td>\approx 6800</td>
<td>Kakade (2001)</td>
</tr>
<tr>
<td>CE+RL</td>
<td>21,252</td>
<td></td>
</tr>
<tr>
<td>CE+RL, constant noise</td>
<td>72,705</td>
<td></td>
</tr>
<tr>
<td>CE+RL, decreasing noise</td>
<td>348,895</td>
<td></td>
</tr>
</tbody>
</table>
Likelihood Ratio Policy Gradient

We let $\tau$ denote a state-action sequence $s_0, u_0, \ldots, s_H, u_H$.

- Optimizing the expected utility
  \[ U(\theta) = \mathbb{E}\left[ \sum_{t=0}^{H} R(s_t, u_t); \pi_\theta \right] = \sum_{\tau} P(\tau; \theta) R(\tau) \]

- Is almost the same as MLE except for a weight $P(t; \theta) R(t)$

- Valid even if $R$ and sample space are discrete!!

- Unstable, need good model initialization and ways to reduce gradient variances

[Aleksandrov, Sysoyev, & Shemeneva, 1968]
[Rubinstein, 1969]
[Glynn, 1986]
[Reinforce, Williams 1992]
[GPOMDP, Baxter & Bartlett, 2001]
The Step Size Problem

Why are step sizes a big deal in RL?

- Supervised learning
  - Step too far → next updates will fix it
- Reinforcement learning
  - Step too far → bad policy
  - Next batch: collected under bad policy
  - Can’t recover, collapse in performance!

- Bad stability
Surrogate Objective

- Collect data with an old policy (for stability)
- Reweight examples by importance sampling
  - The probability ratio between the new policy and the old policy

\[
L(\pi) = \mathbb{E}_{\pi_{\text{old}}} \left[ \frac{\pi(a \mid s)}{\pi_{\text{old}}(a \mid s)} A^{\pi_{\text{old}}}(s, a) \right]
\]

\[\nabla_\theta L(\pi_\theta) \bigg|_{\theta_{\text{old}}} = \nabla_\theta \eta(\pi_\theta) \bigg|_{\theta_{\text{old}}} \quad \text{(policy gradient)}\]

[Kakade and Langford 2002]
[Schulman+ 2015]
Experience Replay

- Keep a set of (hard to find, or human generated) good examples
- Repeatedly use them for training (together with recent bad examples)
  - E.g., 1M replay buffer for DQN Artari training
  - E.g., Neural symbolic machines keep track of the best program for each query

Some good things never last but good memories remains forever.
- Lailani
The Delayed Reward Problem

- With policy gradient methods, we are confounding the effect of multiple actions:

\[ \hat{A}_t = r_t + r_{t+1} + r_{t+2} + \cdots - b(s_t) \]

mixes effect of \( a_t, a_{t+1}, a_{t+2}, \ldots \)

- SNR of \( \hat{A}_t \) scales roughly as \( 1/T \)
  - Only \( a_t \) contributes to signal \( A^\pi(s_t, a_t) \), but \( a_{t+1}, a_{t+2}, \ldots \) contribute to noise.

- Bad data efficiency
Bootstrapping

- use the value function to estimate future rewards

  - Subtracting out baselines, we get advantage estimators

    \[
    \hat{A}^{(1)}_t = r_t + \gamma V(s_{t+1}) - V(s_t) \\
    \hat{A}^{(2)}_t = r_t + r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t) \\
    \vdots \\
    \hat{A}^{(\infty)}_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots - V(s_t)
    \]

  - \( \hat{A}^{(1)}_t \) has low variance but high bias, \( \hat{A}^{(\infty)}_t \) has high variance but low bias.

  - Using intermediate \( k \) (say, 20) gives an intermediate amount of bias and variance
Advantage Actor-Critic

- Minimize reward loss and value function error at the same time

\[
\text{for iteration=1, 2, \ldots do} \\
\text{Agent acts for } T \text{ timesteps (e.g., } T = 20), \\
\text{For each timestep } t, \text{ compute} \\
\hat{R}_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_t) \\
\hat{A}_t = \hat{R}_t - V(s_t)
\]

\(\hat{R}_t\) is target value function, in regression problem
\(\hat{A}_t\) is estimated advantage function
Compute loss gradient \(g = \nabla_\theta \sum_{t=1}^{T} \left[ - \log \pi_\theta(a_t | s_t) \hat{A}_t + c(V(s) - \hat{R}_t)^2 \right]\)
\(g\) is plugged into a stochastic gradient descent variant, e.g., Adam.

\text{end for}
Meta-Learning
Meta-Generative Models

- Two models works better than one

Variational Autoencoders (VAE)
Kingma and Welling
[1312.6114]

Generative Adversarial Networks (GAN)
Goodfellow et al.
[1406.2661]
Meta-RL Models

- Two learning systems:
  - one lower-level system that learns relatively quickly, and which is primarily responsible for adapting to each new task;
  - and a slower higher-level system that works across tasks to tune and improve the lower-level system.
Meta-RL Models

Figure 7: Sample gameplay by our agent on Montezuma’s Revenge: The four quadrants are arranged in a temporally coherent manner (top-left, top-right, bottom-left and bottom-right). At the very beginning, the meta-controller chooses key as the goal (illustrated in red). The controller then tries to satisfy this goal by taking a series of low level actions (only a subset shown) but fails due to colliding with the skull (the episode terminates here). The meta-controller then chooses the bottom-right ladder as the next goal and the controller terminates after reaching it. Subsequently, the meta-controller chooses the key and the top-right door and the controller is able to successfully achieve both these goals.
Meta-Optimizer

- Control NN parameter updates using LSTMs

\[ L(\phi) = \mathbb{E}_f \left[ \sum_{t=1}^{T} w_t f(\theta_t) \right] \]

where

\[ \theta_{t+1} = \theta_t + g_t, \quad \begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi) \]
Multiresolution Caption Model

[Yang+ 2016]
Multiresolution Dialogue Models

- Sorry I can't find their poster online

Table 5: Twitter Coarse Sequence Examples

<table>
<thead>
<tr>
<th>Natural Language Tweets</th>
<th>Noun Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;firstSpeaker&gt; at pinkberry with my pink princess enjoying a precious moment &lt;url&gt;</td>
<td>present_tenses pinkberry</td>
</tr>
<tr>
<td></td>
<td>princess moment</td>
</tr>
<tr>
<td>&lt;secondSpeaker&gt; they are adorable, alma still speaks about emma bif sis . hugs</td>
<td>present_tenses alma emma</td>
</tr>
<tr>
<td></td>
<td>bif sis hugs</td>
</tr>
<tr>
<td>&lt;firstSpeaker&gt; &lt;at&gt; when you are spray painting, where are you doing it? outside? in your apartment? where?</td>
<td>present_tenses spray painting</td>
</tr>
<tr>
<td></td>
<td>apartment</td>
</tr>
<tr>
<td></td>
<td>present_tenses spray stuff</td>
</tr>
<tr>
<td></td>
<td>bathroom</td>
</tr>
<tr>
<td>&lt;secondSpeaker&gt; &lt;at&gt; mostly spray painting outside but some little stuff in the bathroom</td>
<td>present_tenses spray stuff</td>
</tr>
<tr>
<td></td>
<td>bathroom</td>
</tr>
</tbody>
</table>

MrRNNs generalize better by having z’s:
- follow latent stochastic process (z’s are marginally independent w’s)
- use factorial representation (z_n is a sequence of discrete tokens)
- incorporate prior knowledge (e.g., technical support representations)

Hierarchical generation process helps generate more meaningful and on-topic (goal-driven) responses.
Multiresolution Dialogue Models

Table 6: Ubuntu Coarse Sequence Examples

<table>
<thead>
<tr>
<th>Natural Language Dialogues</th>
<th>Activity-Entity Coarse Dialogues</th>
</tr>
</thead>
<tbody>
<tr>
<td>if you can get a hold of the logs, there's stuff</td>
<td>future_tenses get_activity install_activity</td>
</tr>
<tr>
<td>from <strong>unknown</strong> about his inability to install</td>
<td>amd64_entity no_cmd</td>
</tr>
<tr>
<td>amd64</td>
<td>no_tenses check_activity no_cmd</td>
</tr>
<tr>
<td>I’ll check fabbione’s log, thanks sounds like he</td>
<td>past_present_tenses none_activity no_cmd</td>
</tr>
<tr>
<td>had the same problem I did ew, why? ...</td>
<td>no_tenses none_activity no_cmd</td>
</tr>
<tr>
<td>upgrade lsb-base and acpid</td>
<td>no_tenses upgrade_activity lsb_entity</td>
</tr>
<tr>
<td>i’m up to date</td>
<td>acpid_entity no_cmd</td>
</tr>
<tr>
<td>what error do you get?</td>
<td>no_tenses none_activity no_cmd</td>
</tr>
<tr>
<td>i don’t find error :/ where do i search from?</td>
<td>present_tenses get_activity no_cmd</td>
</tr>
<tr>
<td>acpid works, but i must launch it manually in a root</td>
<td>present_tenses discover_activity no_cmd</td>
</tr>
<tr>
<td>stem ...</td>
<td>present_future_tenses work_activity</td>
</tr>
<tr>
<td></td>
<td>acpid_entity root_entity no_cmd</td>
</tr>
</tbody>
</table>
Thanks
Reference

3. Keynote: https://t.co/LDzqac7na1
5. GAN: http://www.slideshare.net/indicods/deep-advancements-in-generative-modeling