imitation learning

Recruent

Neural Networks
imitation learning

Recurrent Neural Networks

Non-differentiable
Discontinuous
Non-backpropable
Discrete Choices?
Examples of structured joint prediction
Sequence labeling

x = the monster ate the sandwich
y = Dt Nn Vb Dt Nn

x = Yesterday I traveled to Lille
y = - PER - LOC
NLP algorithms use a kitchen sink of features.
Segmentation
Simultaneous (machine) interpretation

- Dozens of defendants
- Judges from four nations (three languages)
- Status quo: speak, then translate
- After Nuremberg, simultaneous translations became the norm
- Long wait → bad conversation

Nuremburg Trials
Why simultaneous interpretation is hard

- Human languages have vastly different word orders
  - About half are OV, the other half are VO
  - This comes with a lot more baggage than just verb-final

Running (German/English) Example:

Ich bin mit dem Zug nach Ulm gefahren
I am with the train to Ulm traveled
I (...... waiting. ......) traveled by train to Ulm
Model for interpretation decisions

- We have a set of actions (predict / translate)
  - Wait
  - Predict clause-verb
  - Predict next word
  - Commit ("speak")

- In a changing environment (state)
  - The words we've seen so far
  - Our models' internal predictions

- With well-defined notions of:
  - Reward (or loss) at the end
  - Optimal action at training time
Example of interpretation trajectory

Big Challenges:
No supervision about when to “wait”
Complicated loss/reward functions

Ich bin mit dem Zug nach Ulm gefahren
I am with the train to Ulm traveled
I [waiting.....] traveled by train to Ulm
Back to the original problem...

- How to optimize a discrete, joint loss?

- Input: \( x \in X \)

- Truth: \( y \in Y(x) \)

- Outputs: \( Y(x) \)

- Predicted: \( \hat{y} \in Y(x) \)

- Loss: \( \text{loss}(y, \hat{y}) \)

- Data: \( (x,y) \sim D \)
Back to the original problem...

- How to optimize a discrete, joint loss?

- Input: \( x \in X \)
- Truth: \( y \in Y(x) \)
- Outputs: \( Y(x) \)
- Predicted: \( \hat{y} \in Y(x) \)
- Loss: \( \text{loss}(y, \hat{y}) \)
- Data: \( (x,y) \sim D \)

Goal:

find \( h \in H \)
such that \( h(x) \in Y(x) \)
minimizing

\[
E_{(x,y) \sim D}[\text{loss}(y, h(x))] 
\]

based on \( N \) samples

\[
(x_n, y_n) \sim D
\]
Search spaces

- When $y$ decomposes in an ordered manner, a sequential decision making process emerges.
Search spaces

- When $y$ decomposes in an ordered manner, a sequential decision making process emerges.

Encodes an output $\hat{y} = \hat{y}(e)$ from which $\text{loss}(y, \hat{y})$ can be computed (at training time).
Policies

- A policy maps observations to actions

\[ \pi(o) = a \]

- input: \( x \)
- timestep: \( t \)
- partial traj: \( \tau \)
- ... anything else

obs.
An analogy from playing Mario

From Mario AI competition 2009

Input:

Output:
- Jump in \{0,1\}
- Right in \{0,1\}
- Left in \{0,1\}
- Speed in \{0,1\}

High level goal:
Watch an expert play and learn to mimic her behavior
Training (expert)

Sample Expert Trajectories
Warm-up: Supervised learning

1. Collect trajectories from expert $\pi^\text{ref}$
2. Store as dataset $\mathbf{D} = \{ (\mathbf{o}, \pi^\text{ref}(\mathbf{o},y)) \mid \mathbf{o} \sim \pi^\text{ref} \}$
3. Train classifier $\pi$ on $\mathbf{D}$

- Let $\pi$ play the game!
Test-time execution (sup. learning)
What's the (biggest) failure mode?

The expert never gets stuck next to pipes

⇒ Classifier doesn't learn to recover!
Kittens, revisited.

(Held & Hein, 1936)
Warm-up II: Imitation learning

1. Collect trajectories from expert $\pi^{\text{ref}}$
2. Dataset $D_0 = \{ (o, \pi^{\text{ref}}(o,y)) \mid o \sim \pi^{\text{ref}} \}$
3. Train $\pi_1$ on $D_0$
4. Collect new trajectories from $\pi_1$
   ➢ But let the expert steer!
5. Dataset $D_1 = \{ (o, \pi^{\text{ref}}(o,y)) \mid o \sim \pi_1 \}$
6. Train $\pi_2$ on $D_0 \cup D_1$

• In general:
  • $D_n = \{ (o, \pi^{\text{ref}}(o,y)) \mid o \sim \pi_n \}$
  • Train $\pi_{n+1}$ on $\bigcup_{i \leq n} D_i$

If $N = T \log T$,\n$L(\pi_n) < T \varepsilon_N + O(1)$ for some $n$
Test-time execution (DAGger)
What's the biggest failure mode?

Classifier only sees *right* versus *not-right*

- No notion of *better* or *worse*
- No *partial credit*
- Must have a single *target* answer
Learning to search: LOLS

1. Let learned policy $\pi$ drive for $t$ timesteps to obs. $o$

2. For each possible action $a$:
   - Take action $a$, and let expert $\pi^{\text{ref}}$ drive the rest
   - Record the overall loss, $c_a$

3. Update $\pi$ based on example:
   - $(o, \langle c_1, c_2, \ldots, c_K \rangle)$

4. Goto (1)

Side note: can also be run in “bandit” mode w/ sampling
So..... what's the connection?

This looks a lot like an RNN!
Two quick results

- If you *don't* backprop through time:
  - POS tagging: no change
  - Named entity recognition: marginal improvement
  - Dependency parsing: 1% gain over strong baseline

- If you *do* backprop through time:
  - Synthetic sequence labeling data, Gaussian obs
  - Cannot exactly fit (most) generated datasets
  - Mashup: 10.4% error 1.4% error on training data!

Code at http://hal3.name/tmp/rnnlols.py (thanks to autograd folks for autograd!)
Simultaneous machine interpretation is a super fun problem and you should work on it!

Not being able to backprop something isn't always the end of the world – you're not stuck with RL!

RNN+LOLS mashup appears promising!

Thanks! Questions?

I am on the job market! umiacs.umd.edu/~hhe

Alekh Agarwal
Kai-Wei Chang
Akshay Krishnamurthy
John Langford
He He

ICPR '10  EMNLP'13  ICC'15
CVPR '11  EMNLP'14  Fusion'15
EMNLP'12  NIPS '14  EMNLP'15
NIPS '12  SLT '14 + more