Course Review and AlphaGo

CS 287
Today’s Lecture

- Overview of the models/tasks covered course
- AlphaGo
Contents

Course Review

Modeling

AlphaGo
Q: Please write me a sonnet on the subject of the Forth Bridge.
A: Count me out on this one. I never could write poetry.
Q: Add 34957 to 70764.
A: (Pause about 30 seconds and then give as answer) 105621.
Q: Do you play chess?
A: Yes.
Q: I have K at my K1, and no other pieces. You have only K at K6 and R at R1. It is your move. What do you play?
A: (After a pause of 15 seconds) R-R8 mate. - Turing (1950)
Lexicons and Lexical Semantics

Zipf’ Law (1935, 1949):

*The frequency of any word is inversely proportional to its rank in the frequency table.*
The Shannon Game (Shannon and Weaver, 1949):

*Given the last n words, can we predict the next one?*

The pin-tailed snipe (Gallinago stenura) is a small stocky wader. It breeds in northern Russia and migrates to spend the __

- Probabilistic models have become very effective at this task.
- Crucial for speech recognition (Jelinek), OCR, automatic translations, etc.
(3) Compositionality of Syntax and Semantics

*Probabilistic models give no insight into some of the basic problems of syntactic structure* - Chomsky (1956)

Show me flights from NY to LA Thursday evening.

\[ \lambda x. \text{flight}(x) \land \text{from}(x, \text{ny}) \land \text{to}(x, \text{la}) \land \text{during}(x, \text{evening}) \land \text{day}(x, \text{thur}) \]
Language is not merely a bag-of-words but a tool with particular properties - Harris (1954)
It is based on the belief that in modeling language understanding, we must deal in an integrated way with all of the aspects of language syntax, semantics, and inference. - Winograd (1972)

The city councilmen refused the demonstrators a permit because they [feared/advocated] violence.

▶ Recently (2011) posed as a challenge for testing commonsense reasoning.
Contents

Course Review

Modeling

AlphaGo
Machine Learning Approaches to NLP

Many problem-specific modeling questions,

- $x$; input representation
- $y$; output representation
- Model architecture
- Objective

This Course: Focus on supervised data-driven, end-to-end approaches
Input Representations

1. Sparse Features
2. Dense Features (Embeddings)
3. Convolutional NN
4. Recurrent NN
Deep Learning for NLP

Deep Learning waves have lapped at the shores of computational linguistics for several years now, but 2015 seems like the year when the full force of the tsunami hit major NLP conferences.

- Chris Manning (Computational Linguistics and Deep Learning)
Neural Network Toolbox

**Embeddings**
sparse features $\Rightarrow$ dense features

**Convolutions**
feature n-grams $\Rightarrow$ dense features

**RNNs**
feature sequences $\Rightarrow$ dense features
Embeddings \quad \text{sparse features} \quad \Rightarrow \quad \text{dense features}
Convolutions

feature n-grams \implies dense features

\[ h_{t-1} = W \begin{bmatrix} x_{t-1} \\ x_{t-2} \\ x_{t-3} \end{bmatrix} \]
(Zeiler and Fergus, 2014)
RNNs/LSTMs feature sequences $\Rightarrow$ dense features

\[ h_{t-1} = f(Wx_{t-1} + Uh_{t-2} + b) \]
A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

(Xu et al, 2015)
I get pitched regularly by startups doing generic machine learning which is, in all honesty, a pretty ridiculous idea. Machine learning is not undifferentiated heavy lifting, it's not commoditizable like EC2, and closer to design than coding.

- Joseph Reisinger (Computational Linguistics and Deep Learning)
Pipeline Steps

- Morphological Seg
- Morphological Tagging
- Part-of-Speech
- Entity Recognition
- Syntactic Parsing
- Role Labeling
- Discourse Analysis

(Marton et al., 2010)
What model should I use?

Questions to ask:

▶ Do I have significant amounts of supervised data?
▶ Do I have prior knowledge of my problem/domain?
▶ What is the underlying metric of interest?
▶ Do I need interpretability of the model?
▶ Is the structure of the text important?
▶ Is training efficiency/prediction efficiency important?
Example: Simple Question Answering

10 Mary moved to the hallway.
11 Daniel travelled to the office.
12 Where is Daniel?  office  11

- Input is the sentences and the question
- Output is a set of possible answers.
- How might you go about selecting an answer?
Contents

Course Review

Modeling

AlphaGo
DIFFICULTY OF VARIOUS GAMES FOR COMPUTERS

EASY

SOLVED
COMPUTERS CAN PLAY PERFECTLY

SOLVED FOR ALL POSSIBLE POSITIONS

TIC-TAC-TOE

NIM

GHOST (1989)

CONNECT FOUR (1995)

COMPUTERS CAN BEAT TOP HUMANS

SOLVED FOR STARTING POSITIONS

GOMOKU

CHECKERS (2007)

SCRABBLE

COUNTERSTRIKE

REVERSI

BEER PONG (UIUC ROBOT)

FEBRUARY 10, 1996: FIRST WIN BY COMPUTER AGAINST TOP HUMAN
NOVEMBER 21, 2005: LAST WIN BY HUMAN AGAINST TOP COMPUTER

COMPUTERS STILL LOSE TO TOP HUMANS (BUT FOCUS ON R&D COULD CHANGE THIS)

CHESS

JEOPARDY!

STARRAFT

POKER

ARIMAA

GO

COMPUTERS MAY NEVER OUTPLAY HUMANS

SNAKES AND LADDERS

MAO

SEVEN MINUTES IN HEAVEN

CALVINBALL
https://www.youtube.com/watch?v=Jq5S0bMdV3o
Google’s Computer Program Beats Lee Se-dol in Go Tournament

By CHOE SANG-HUN  MARCH 15, 2016

Lee Se-dol with his daughter Lee Hye-lim on his way to the last Go match with Google's AlphaGo artificial intelligence program in Seoul, South Korea. Photo by Yonhap/Newsis
AlphaGo Overview

1. Learn a model to predict one-step move from experts
2. Refine by self-play reinforcement learning
3. Use as part of game-tree search.
Policy Setup

Given current board state $s$, distribution over actions $a$.

- Learn a policy, $p(a|s)$
- Estimate distribution with softmax.
- Gives a one-step Go player.
(1) Policy Network

Learned from 29.4 million positions from 160,000 expert games

Two models:

- $p_\pi(a|s)$; multiclass logistic regression (pattern+sparse features)
- $p_\sigma(a|s)$; deep convolutional network
$p_{\alpha|\rho}(a|s)$
Deep Convolutional Network

The first hidden layer zero pads the input into a 23x23 image, then convolves $k$ filters of kernel size 5x5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21x21 image, then convolves $k$ filters of kernel size 33 with stride 1, again followed by a rectifier nonlinearity.
Deep Convolutional Network

The first hidden layer zero pads the input into a 23×23 image, then convolves $k$ filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves $k$ filters of kernel size 33 with stride 1, again followed by a rectifier nonlinearity.
The step size $\alpha$ was initialized to 0.003 and was halved every 80 million training steps, without momentum terms, and a mini-batch size of $m = 16$. Updates were applied asynchronously on 50 GPUs using DistBelief 61; gradients older than 100 steps were discarded. Training took around 3 weeks for 340 million training steps.
(2) Reinforcement Learning

- Refine one-step player by playing against itself.
- Popular technique for stochastic games (TD-Gammon)
- Reinforcement learning objects to account for single-step bias
Self-Play with Policy Gradient

Start with $p_\sigma$ and play against itself to learn:

- $p_\rho(a|s)$; deep convolution network (policy gradient)

Process: Training epoch $J + 1$

1. Sample opponent from previous version of model $j < J$
2. Play game between players $p_\rho^J$ and $p_\rho^j$
3. Update weights using policy gradient on RL objective

$$
\Delta \rho \propto \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} z_t
$$

where $z_t \in \{-1, 1\}$ represents the final outcome of the game.
Value Network

- Policy network trains only move at state.
- Useful also to know the value of a state.

\[ v(s) = E_{p_{\rho}}[z_t | s] \]

Generally done using game-specific heuristics.
Value Network

Apply similar architecture for computing state value,

- $v_\theta$; deep CNN regression
- Trained on self-play data set.
- Minimize MSE with final self-play result.
When trained on the KGS data set in this way, the value network memorized the game outcomes rather than generalizing to new positions, achieving a minimum MSE of 0.37 on the test set, compared to 0.19 on the training set. To mitigate this problem, we generated a new self-play data set consisting of 30 million distinct positions, each sampled from a separate game. Each game was played between the RL policy network and itself until the game terminated.
Policy network

\[ p_{\sigma/\rho}(a|s) \]

Value network

\[ v_{\theta}(s') \]
Rollout policy $p_\pi$
SL policy network $p_\sigma$
RL policy network $p_\rho$
Value network $v_\theta$

Human expert positions
Self-play positions
(3) Game Search

Utilize the learned models with an advanced game-search algorithm

- Similar to standard game tree algorithms (CS182)
- Monte Carlo Tree Search (MCTS)
  - Select
  - Expand
  - Eval
  - Update/Backup
- Progressively expands the search space based on models
Select and Expansion

- \( Q(s, a) \); current expected value of taking action \( a \) at \( s \)
- \( u(s, a) \); prior for taking \( a \) at \( s \) defined by \( p_\sigma \)

**Selection** step at state \( s \),

\[
\arg\max_a Q(s, a) + u(s, a)
\]

Based on selection, either move to seen node or **expand**.
Game Search

(a) Selection

\[ Q + u(P) \rightarrow \text{max} \rightarrow Q + u(P) \]

(b) Expansion

\[ p_o \left( \begin{array}{c} \text{node} \end{array} \right) \]

\[ P \rightarrow \text{node} \rightarrow P \]
State Evaluation

Reached “leaf” state $s_L$, want to **evaluate**

- Compute value as $V(s_L)$,

  $$V(s_L) = (1 - \lambda) v_\theta(s_l) + \lambda (R(s_L))$$

  where $R$ is a rollout. Monte carlo simulation using $p_\pi$

- Convex combination of value network and simulation under simple model

Why not $p_\sigma$? Where did $p_\rho$ go?
Move Selection

- After leaf evaluation all previous $Q$ values are **updated** based on $V(s_L)$.
- Process is run many times.
- Actual play is based on most commonly taken action.
Results

Value network
Results

![Graph showing mean squared error on expert games over move number for different policies: Uniform random rollout policy, Fast rollout policy, Value network, SL policy network, and RL policy network.]
Results

![Graph showing AlphaGo win rate (%) against training accuracy on KGS dataset (%). The graph includes lines for different numbers of filters: 128, 192, 256, and 384 filters. Each line represents the win rate for a specific number of filters, increasing as the training accuracy increases.](image-url)
Results