Computing Science (CMPUT) 455

Search, Knowledge, and Simulations

James Wright

Department of Computing Science
University of Alberta
james.wright@ualberta.ca

Fall 2021

Part IV

Machine Learning for Heuristic Search

455 Today - Lecture 17

- Machine Learning - Introduction of Concepts
- Types of learning
- Problems in machine learning
- Learning for games
- Coursework: Finish Assignment 3, due Nov 15
- Uploads:
  - `Go4` code - simulation-based player with probabilistic simulations
  - `Go5` code - MCTS-based Go player

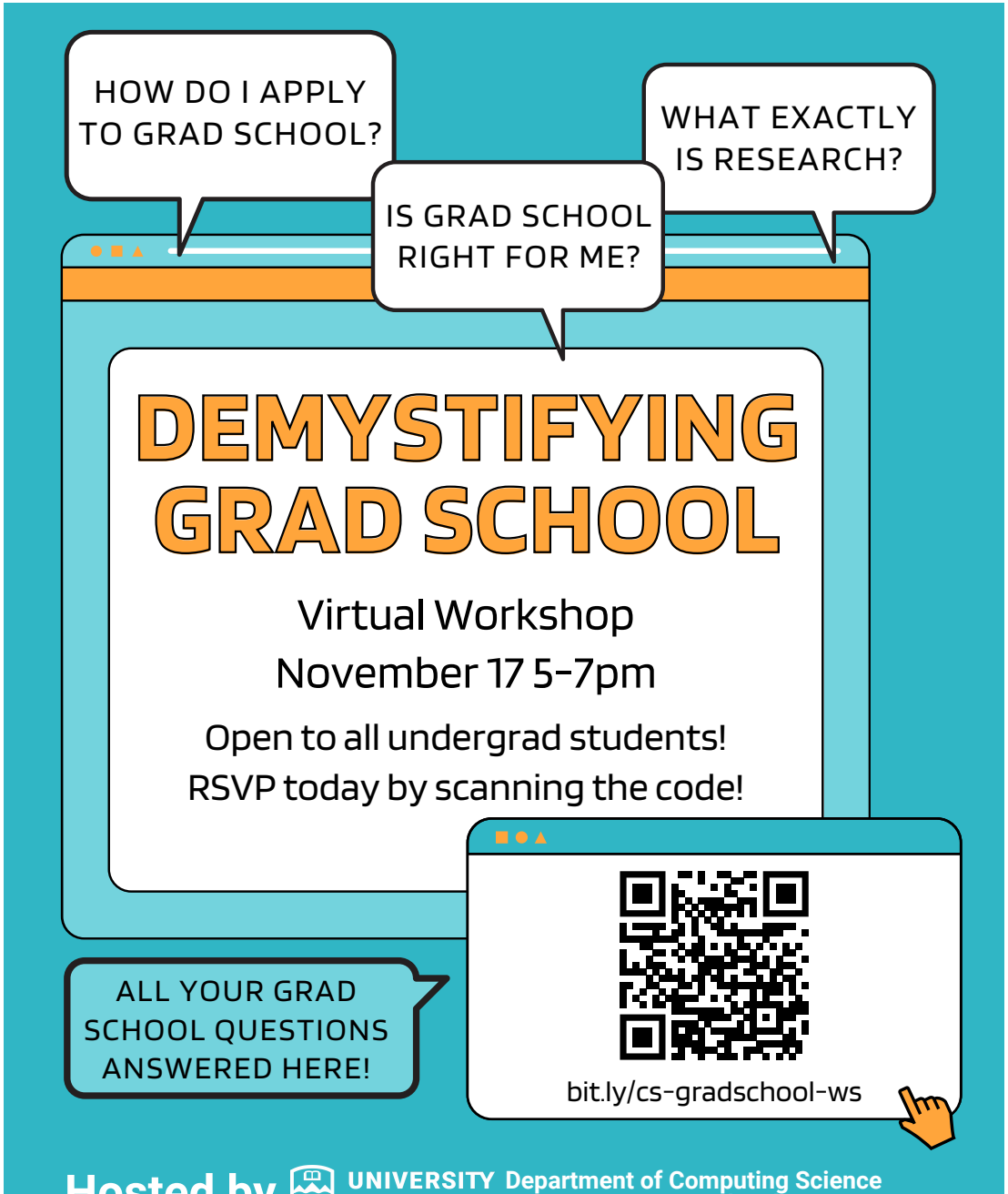
Go4

- Simulation-based player
- 1-ply search + simulations as in Go3
- Probabilistic simulation policy,
  similar to Lecture 14 and Coulom's paper
- It works but is very slow
- Experiment: empty 5 $\times$ 5 board, default settings, genmove
  - Go3, almost-random simulations: 1.8 seconds
  - Go3, rule-based simulations: 9.9 seconds
  - Go4, probabilistic simulations: 50 seconds

Go5

- MCTS-based Go player
- Options similar to `Go4`, see `Go5.py`
- MCTS implementation in `Go5/mcts.py`

Grad School!


HOW DO I APPLY TO GRAD SCHOOL?
IS GRAD SCHOOL RIGHT FOR ME?
WHAT EXACTLY IS RESEARCH?
DEMYSTIFYING GRAD SCHOOL
Virtual Workshop
November 17 5-7pm
Open to all undergrad students!
RSVP today by scanning the code!
ALL YOUR GRAD SCHOOL QUESTIONS ANSWERED HERE!
bit.ly/cs-gradschool-ws
Hosted by UNIVERSITY Department of Computing Science

Machine Learning for Heuristic Search - Introduction

- Main Concepts of machine learning (ML)
- Use of ML in heuristic search
- Learning as function approximation
- Overfitting problem
- Example: linear regression and least squares
- Representation and models for ML

What is Machine Learning?

- Wikipedia: Give computers the ability to learn without being explicitly programmed.
- Domingos article: Algorithms that figure out how to perform important tasks by generalizing from examples.
- Amii: Machine learning enables a computer system to independently learn from, and continuously adapt to, data without being explicitly programmed for that data.

Examples of Machine Learning Applications

- Ad placement on web pages
- Spam filters for email
- High-frequency trading
- Image, speech and text recognition
- Robotics
- Games
- Thousands of other applications, increasing rapidly

Which Kinds of ML are Most Useful for Heuristic Search?

- **Learn an evaluation function**
- **Learn a move generation policy**
- Learn search control for the tree search
  - Which parts of a tree to search (first)?
- Learn a filter - which moves to cut
- Learn time control
  - How much time to use on each move?
  - Spend more time on more important decisions

Simplest Learning - Remembering Facts

- Rote learning in humans: remember facts
- Not an issue in computers - databases can store massive amounts of facts
- This is (mostly) a solved problem
- Cache and re-use results of previous computations

Remembering Facts - Examples in Heuristic Search and Learning

- Transposition table
- Compute and store winning strategy for whole game, then follow it
  - Endgame database, as in checkers
- Tabular learning - simple kind of reinforcement learning
  - Learn a table of all states and their expected reward

Learning as Function Approximation and Generalisation

Large, complex problems: cannot learn all the facts

- What can we learn?
- Learn abstract concepts
- Learn general knowledge from (many) examples
- Find interesting trends, correlations in your data
- Learn evaluation function
- Predict moves in the game of Go

Supervised, Unsupervised, and Reinforcement Learning

- Supervised: learn from *labeled* training data - labeled with the correct result
  - Example: learn from master moves in game records. Label of position = move played by the master
- Unsupervised: unlabeled training data
  - Example: clustering, dimension reduction, feature extraction
- Reinforcement: interact with the environment, then learn from rewards
  - Example: learn from playing games.
    Reward = win/loss at the end of the game
- Semi-supervised learning:
  - Some (often small) amount of labeled data
  - Some (large) amount of unlabeled data

Learning with a Model vs Learning from Experience

- Learning with a Model: allows us to try out actions and observe results in simulator
- Learning from Experience: real world, learn from observations of the effects of real actions
- Simulator: learn from simulated experience from experiments within the simulator
- Most domains:
  - Must worry about the error from modelling
  - Garbage in - garbage out

Learning with a Model in Games

- In games we have fast and perfect simulators -
- Truthful: exact implementation of rules and actions
- Fast, efficient
- Cheap, can repeat as often as needed
- Games are ideal test beds for learning
- Used for much groundbreaking work on learning

Input, Output, and Representation

- Key questions before we start a machine learning task:
- What is the input? How is it represented?
- What is the output? How is it represented?
- What is learned? How is that represented?
- We get to choose representations.
- Good choices can make a huge difference in the results

Learn a Function from Data

- Given training examples: data points (x_i, y_i)
- Learn a function $y = f(x)$
- Goal: approximate the data as well as possible
- Use function for *prediction*:
  - Given a new x-value
  - What should the y-value be?

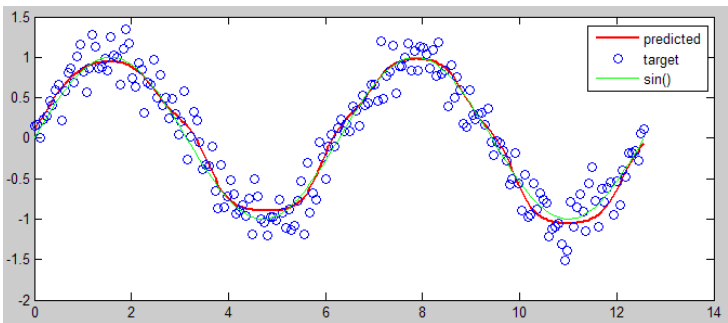


Image source: `http://stackoverflow.com/questions/1565115/`

Goals:

- Good approximation
- Robust against noisy data
- Avoid overfitting

Training Data vs Test Data

- Training data
  - Data used for the machine learning process
- Test data
  - Data used to evaluate the quality of the learned function
- Training data and test data should be *independent* samples from the same learning problem
- One rule of thumb:
  split data into 90% training, 10% test
- Example: given 100,000 master-level Go games:
  - Use 90,000 games for training
  - 10,000 games for test

Overfitting

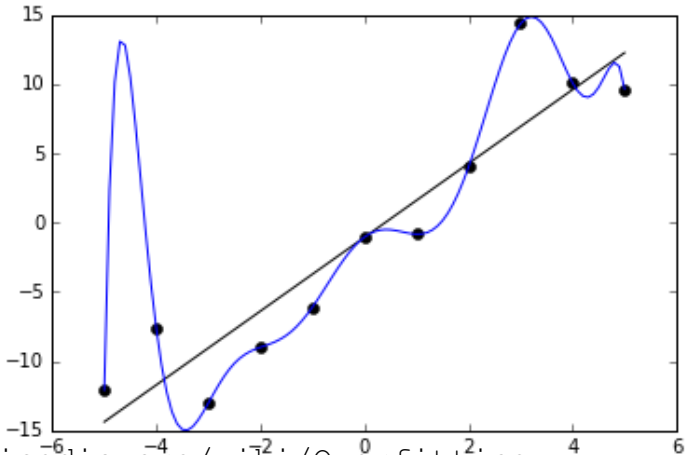


Image source: `https://en.wikipedia.org/wiki/Overfitting`

- Problem: learning the noise as well as the data
- If you fit the noisy data too closely,
  it will not generalize well to new data
- Example: fit exact polynomial through noisy data points
- Depending on the noise level in the data,
  a simple linear function may be better

- Opposite problem: underfitting - cannot see the regularity
  in the data from the learned function

More Overfitting

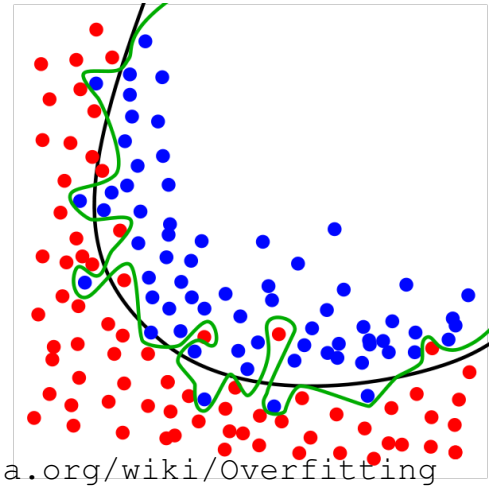


Image source: `https://en.wikipedia.org/wiki/Overfitting`

- Complicated green line:
  separates red from blue data points exactly
- What if there is some noise in the data?
  The green line overfits to the outlier data points
- Likely the simpler black line is the better separator
- More examples in the Domingos paper

Error from Overfitting

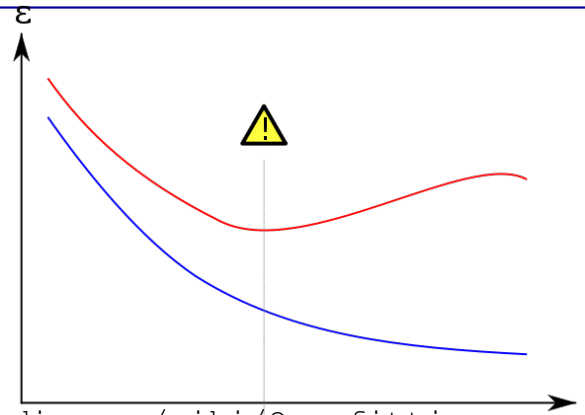


Image source: `https://en.wikipedia.org/wiki/Overfitting`

- x-axis = training time
  y-axis = error of the learned function
- Blue line: error on training data
- Red line: error on test data
- In the beginning, learning from training data works well on the test data
- At some point, it begins to overfit to the training data
- The generalization performance on the test data becomes worse

Linear Regression

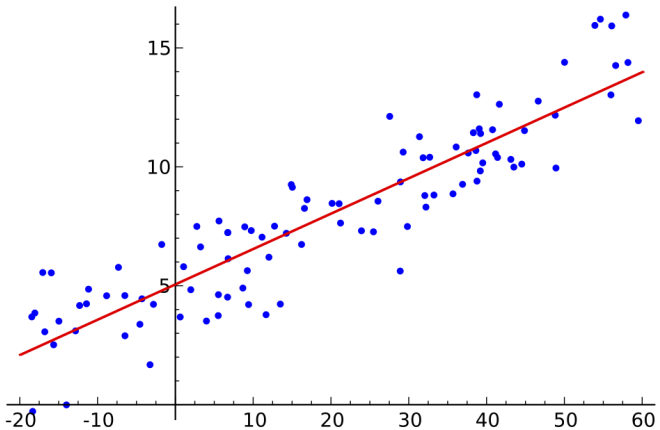


Image source:

`https://en.wikipedia.org/wiki/`

`https://en.wikipedia.org/wiki/`

`Regression_analysis`

- Find best linear function $y = ax + b$ to approximate the data
- Only two parameters a, b to optimize
- Standard approach to solve: least squares

Least Squares

- Given data points (x_i, y_i)
- Find a linear function $y = ax + b$ which approximates the data as well as possible
- Error for point i = difference between y_i and $ax_i + b$
- $|y_i - (ax_i + b)|$
- Difficult to optimize with absolute values, so *squared error* is much more popular
- Find a, b which minimize $\sum_i (y_i - (ax_i + b))^2$

Least Squares (2)

- Find a, b which minimize $\sum_i (y_i - (ax_i + b))^2$
- Closed-form solution using simple calculus
  - Best values for a and b computed as functions of $\{(x_i, y_i)\}_i$
- Many python sample codes on the net, e.g. `http://machinelearningmastery.com/implement-simple-linear-regression-scratch-pytho`
- Details: `https://en.wikipedia.org/wiki/Simple_linear_regression`

Generalizations

- Linear regression is the most basic model
- Many more general statistical models exist
- Fit nonlinear functions, e.g. polynomials
- **Linear functions of more than one variable**
  - Examples in next class
- **Nonlinear functions of more than one variable**
  - Example: neural networks
- Minimize error functions other than least squares

Example for Different Error Function

- Move prediction problem in Go
- Learning task: Learn a value for how good each move is
- Pick the move with highest value
- We do not care about the function itself
- We only care about the move ordering it produces
- We do not directly have a target function to approximate
- Just count the number of correctly predicted moves
- Evaluate the set of function values of all legal moves:
  - Value 1 if master move has highest evaluation among all moves, 0 otherwise
  - This is a *classifier* as discussed in Domingos' article
  - "is best move" vs "is not best move"

Representation and Models for Machine Learning

Main questions:

- How is the input represented?
- **How is the learned model represented?**
- What is the format of the output?

Representation of the Input

- Raw input
  - Location of stones on the board, or sequence of moves
- *Features* which represent (hopefully) useful concepts that will facilitate learning
  - Go examples: selfatari, liberties, proximity to last move, local pattern
  - Simplest case: *binary* features, only two values 0 (off, false) and 1 (on, true)
- Popular for learning in heuristic search:
  - Machine-learned weights
  - Hand-designed features
  - Example: Coulom's MM

Representation of the Model

- What kind of functions are we trying to learn?
- It is our choice, we can use some general principles
- Principle 1: simple is good - helps avoid overfitting
- Principle 2: as complex as needed to represent what we need
  - Example: Linear functions are not enough to give a good evaluation for Go
- Principle 3: functions which represent general assumptions about our world
  - See detailed discussion in Domingos' paper
- Principle 4: functions for which we have efficient learning algorithms
- Choice is closely tied to choice of input representation

Examples of Models

- Linear model
  - Features f_i given as input
  - Learn weights w_i
  - Evaluation: $\sum_i w_i f_i$
- Neural network
  - Simple features or raw data as input
  - (Many) layers of neurons and nonlinear *activation functions*
  - Weights represent strength of connection between neurons

Developing the Model

In practice, we do (many) iterations of:

- Create data
- Develop model
- Use machine learning to learn weights
- Evaluate model
- Find problems or weaknesses with data and/or model
- Repeat

Output of Learning Process

Result of learning in games:

- Classifiers: good/bad move, filter/don't filter for search,...
- Move evaluation or move probabilities
- State, position evaluation
- Local evaluation, e.g. territory maps
  - How likely is point p going to be Black/White/neutral?

Summary

- Introduced some basic concepts and examples of machine learning
- Focus on games
- Touched upon how to apply learning to Go
- Next:
  - Machine learning with simple features in Go
  - First algorithms for learning