Self-Driving DeCal Syllabus

CS 198-95 Spring 2020
Location: Soda 306
Time: Tuesdays, 6:30-8:30 PM
2 Units

Instructors

Aidan Abdulali: aidana@berkeley.edu
Arjun Sripathy: arjunsripathy@berkeley.edu
Brandon Trabucco: btrabucco@berkeley.edu
Rishabh Krishnan: rishabh.krishnan@berkeley.edu
Max Smolin: maximsmol@berkeley.edu
Osher Lerner: oshlern@berkeley.edu

Class Format

Small-group work sessions, personalized instructor mentors, and weekly lectures.

Course Objective

We aim to teach the technological foundation of autonomous driving, the development of new algorithms in this field, and how to implement these algorithms to create an end-to-end autonomous vehicle. Students should leave the course with their own fully functional self driving car in simulation and a rich understanding of the algorithms necessary to pursue car automation further, whether in industry, research, or just for fun. Students will pick up the fundamentals of machine learning, object detection and segmentation, trajectory planning, and motion control. Students will become proficient using important libraries in these domains.

Key Learning Outcomes

- Become knowledgeable about the various systems in place that facilitate self driving cars and how they interact and compliment each other
- Develop a strong understanding of machine learning: what is it and how can I apply it to various classification, regression, and decision making problems

- Leave with a fully functional simulated self driving car which can perceive its surroundings, plan paths, and follow a path using both classical control and reinforcement learning

Guiding Principles

- Tether curriculum to a single simulator (duckietown) and project
    - Allows the assignments to build on work done in previous weeks
    - Maintains engagement by providing a central goal
    - Cuts down the number of python packages and APIs students must learn
- Incrementally build machine learning knowledge in simulator
    - For example: map pedal inputs to acceleration to calculate speed via linear regression. Can add a nonlinearity to promote the use of a fully connected network in this system during the following week.
- Emphasize interaction between teachers and students
    - Vastly helps with frustration and efficiency when students are working on hard portions of their project
    - Helps personalize the course and assure that no students fall through the cracks / lose sight of the goal

Methods of Instruction

- Combination of lecture and group work
- 75 min lecture to go over systems, math, and problem approaches
- 45 min group work so students can discuss ideas amongst themselves and ask for help with their assignments / understanding
- Project to create a fully functional self driving car in simulation

Meeting Frequency

- Weekly 2 hour meetings
- Office hours on the weekends

Grading Policy

- (30%) Homework: checkpoints for the following week will be discussed at the end of each class and teachers will walk around the room and evaluate each group's programming work and attendance. Also groups will push their codebase to github every week and it will be verified that they are on track
- (30%) Final project: does the final project successfully navigate the car from point A to point B? If not, how many modules in the system work?
- (20%) Attendance: One free absence; warning at two unexcused absences; three = NP
    - Excuses: Medical; Family Emergency; Mandatory Previously Unknown Obligation (sporting tournament in a different state, etc.)
- (20%) Participation: is this person contributing to their group, asking questions, being active in class, and
- Pass: students must have at least a 70% in the class and meet attendance policy

Lecture Topics/Schedule

Week 1: Overview of the Course and Duckietown
Week 2: Introduction to Machine Learning and Linear Regression
Week 3: Introduction to Deep Neural Networks

Week 4: Object Detection with CNNs
Week 5: Image Segmentation with CNNs
Week 6: Integrating Vision and System ID with Duckietown

Week 7: Trajectory Planning
Week 8: Following Waypoints with PID and LQR
Week 9: Integration of Planning and Controls with Duckietown

Week 10: Learning to Drive by Imitation
Week 11: Model-Based RL from Pixels
Week 12: Guest Lecture

Week 1: Overview of the Course and Duckietown

Outline:

- Introduction to the course and motivation for learning about self driving cars
- Go over how we are teaching: in a small class setting emphasizing group work and instructor interaction
- What are the goals of self drivings cars? They are a service to humans which provide full automation of locomotion and control; aka a hand a mind free experience
- What are the expectations of students and how can they be successful in this course?
- Demonstration of the duckietown environment and instructions on how to install and get familiar
- Transitions into the group finding portion of this lesson
- Demonstration of the duckietown environment and our self driving setup… what you will achieve by the end of this course

Assignment:

- Set up the duckietown environment, install all the dependencies, and capture a video of you manually exploring a provided simulation environment

Recommended Readings:

- Motivation (MIT Technology Review): https://www.technologyreview.com/s/612754/self-driving-cars-take-the-wheel/

Week 2: Introduction to Machine Learning and Linear Regression

Outline:

- What is the fundamental principle behind all machine learning algorithms? Instead of writing a function to go from input to output like F(x) = y, we can learn the function by looking at inputs and outputs and realizing patterns
- How do we define this problem mathematically? We need a sense of how well we are doing (least squares) and a way to get better over time (optimizers)
- What is a loss function, linearity vs nonlinearity, and how can we reduce our loss?
- We approach solving the linear regression problem from the standpoint of linear algebra (derive least squares regression on the board)
- Derive the least squares solution to linear regression.
- Formulate a simple least squares problem on the board and then solve it using hard coded functions in python

Assignment:

- Linear regression to map pedal input to acceleration. Assignment sheet and office hours will be released after lecture

Recommended Readings:

- Linear Algebra: https://medium.com/@andrew.chamberlain/the-linear-algebra-view-of-least-squares-regression-f67044b7f39b

Week 3: Introduction to Deep Neural Networks

Outline:

- With the understanding of linear regression, we can now look at more interesting scenarios like nonlinearities
- We cannot always model data with lines (quadratic patterns, jumps in the data)
- We can use a fully connected neural network with nonlinear activations to model
    - What is a neural network?
        - What is an activation function?
        - What does a node represent
        - What does an edge represent
    - What does fully connected mean?
- How do we train a neural network?
    - Gradient descent overview (backprop, partial derivatives, ...)

Assignment Description:

- Fully connected networks for nonlinear data
- Assigned week 3; due week 4

Recommended Readings:

- Backpropagation: https://colah.github.io/posts/2015-08-Backprop/

Week 4: Object Detection with CNNs

Outline:

- Why are fully connected networks bad for images?
    - How can we reduce the number of trainable parameters?

- What is a convolution?
- How does a convolution work over an image?
- How can we learn filters for a specific task using what we already know?
- Can we apply a CNN to a problem in duckietown?
    - Yes; duck detection

Assignment:

- Building and training a duck detector
- Assigned week 4; due week 5

Recommended Readings:

- CNNs: https://colah.github.io/posts/2014-07-Conv-Nets-Modular/
- Convolutions: https://colah.github.io/posts/2014-07-Understanding-Convolutions/

Week 5: Image Segmentation with CNNs

Outline:

- Images are rich with information: we do not need all of it
    - Can we compress duckietown images into more useful numbers?
- What if we aren't just trying to answer a yes / no question but rather looking to classify what is in an image?
    - Classification
    - How do we deal with this?
        - Softmax function
        - Outputs as probabilities
    - We can now see if a more general object is in a duckietown image
- What if there are multiple objects?
    - Segmentation
    - Multiple class softmax

Assignment:

- Implement a segmentation algorithm and classify each segment
- Assigned week 5; due week 6

Recommended Readings:

- Image segmentation: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/

Week 6: Integrating Vision and System ID with Duckietown

Outline:

- Recap of what we have done so far and why we have learned about regressions, fully connected networks, and CNNs
- How do we tie these threads together to do something interesting in duckietown?
    - Learn about the environment while manually driving around
- Lots of time during class this day for debugging and instructor help

Assignment:

- Segment, detect, and classify objects in the duckietown environment in real time
- Assigned week 6; due week 7

Recommended Readings:

- How do Self-Driving Cars See? https://towardsdatascience.com/how-do-self-driving-cars-see-13054aee2503

Week 7: Trajectory Planning

Outline:

- Optimized motion planning introduction
    - Leverage benefits of randomness
- Trajectory Optimization
    - Defining constraints
    - The formal math
- What computational tools can we use to solve these problems?
- Introduction to trajopt package
    - How are we going to use trajopt so we don't need to repeat the math we did today?

Assignment:

- Plan a series of paths from a set of As to a set of Bs using trajopt

- Assigned week 7; due week 8

Recommended Readings:

- TrajOpt paper: http://joschu.net/docs/trajopt-paper.pdf
- Textbook Chapter: http://underactuated.mit.edu/underactuated.html?chapter=trajopt
- Code release: https://github.com/joschu/trajopt

Week 8: Following Waypoints with PID and LQR

Outline:

- Demonstrate a simple Proportional Controller controlling a point mass following randomly generated 2D waypoints in a jupyter notebook.
- On Latex slides, derive the Proportional Feedback Controller from an error function, and study the empirical stability of Proportional Control.
- Demonstrate the instability of Proportional Control when the waypoint is constantly moving away, to motivate the PD and PI Controllers.
- Derive the PD controller, and study the empirical stability of it. Demonstrate the PD controller in the same point mass environment.
- Derive the PI controller, and study the empirical stability of it. Demonstrate the PI controller in the same point mass environment.
- Assemble everything into the PID controller. Allow students to experiment with various settings of the tuning coefficients, and discuss automatic tuning methods, see the recommended readings.
- Demonstrate LQR to control the same pointmass with randomly generated waypoints, and have students compare it to PID.
- Derive LQR, and have students code in their own cost function.

Assignment:

- Students will implement a PID controller in python, and use PID to control the duckietown car to follow waypoints in the duckietown simulator.
- Students will leverage an existing iterative LQG library to control the duckietown car to follow waypoints, by defining their own cost function

Recommended Readings:

- Automatic tuning of PID: https://ieeexplore.ieee.org/document/7898617
- LQR, DDP, and LQG: http://cs229.stanford.edu/notes/cs229-notes13.pdf
- Blog summary of LQR: https://medium.com/@jonathan_hui/rl-lqr-ilqr-linear-quadratic-regulator-a5de5104c750

- Blog summary of PID: https://medium.com/@mattia512maldini/pid-control-explained-45b671f10bc7

Week 9: Integration of Planning and Controls with Duckietown

Outline:

- Quickly recap the software stack that students have built up so far: System ID, Object and Road Detection, Path Planning, and Waypoint Reaching.
- Demonstrate a complete project, which plans a path to follow, using the detected position of the road, uses feedback to follow the path, and maps accelerations to motor powers using the System ID.
- Break down the block box communication that occurs between each module, and the abstractions that are being relied on.
- Present how to integrate everything in reverse order. Begin with controls, and guide students to connect this with their state-based path planner.
- Then, present how to migrate from a state-based path planner to a purely vision path planning algorithm (involves localization).

Assignment:

- Students will integrate their controller with their path planning algorithm in the duckietown simulator. The choice of PID or LQR is up to the student.
- Students will integrate path planning with their vision processing pipeline. Images are mapped to locations in a known world model, and object are avoided by weighting these regions with high cost to the planner.

Recommended Readings:

- Using SLAM for localization: https://medium.com/slamcore-blog/the-cumulative-levels-of-slam-competence-5576f33c1c2a
- Dynamic obstacle avoidance: https://www.ri.cmu.edu/pub_files/pub4/ferguson_david_2008_3/ferguson_david_2008_3.pdf

Week 10: Learning to Drive by Imitation

Outline:

- Discuss the limitations of the previously employed methods for controlling UAVs. Namely, the need for a fixed and prespecified driving map.
- Present a demo of ChauffeurNet, based on the released simulator and the provided visualizations scripts with the paper.
- Explain how ChauffeurNet begins to address the limitations of the path planning algorithm, in isolation of localization and controls.
- Discuss the limitations and prospects of algorithms like ChauffeurNet in a panel format, with the whole class.

Assignment:

- There is no technical assignment this week. Students are to continue working on system integration of the components presented last week.
- There will be a short paragraph reflection due about the limitations of the algorithms that students are currently using.

Recommended Readings:

- Chauffeur Net paper: https://arxiv.org/pdf/1812.03079.pdf
- Learning to Drive: Beyond Pure Imitation: https://medium.com/waymo/learning-to-drive-beyond-pure-imitation-465499f8bcb2
- Nvidia DAVE-2: https://devblogs.nvidia.com/deep-learning-self-driving-cars/

Week 11: Model-Based RL from Pixels

Outline:

- Discuss the limitations of the previously employed methods for controlling UAVs. Namely, the need for a robust controller and localization mechanism.
- Present a demo of SOLAR, trained instead on a simple duckietown task, that requires learning how to stop at a stop sign and make a turn.
- Explain how SOLAR begins to address the limitations of modularized algorithms for controlling vehicles, but also introduces new limitations.
- Discuss the limitations and prospects of algorithms like SOLAR in a panel format, with the whole class.

Assignment:

- There is no technical assignment this week. Students are to continue working on system integration of the components presented last week.
- There will be a short paragraph reflection due about the limitations of the algorithms that students are currently using.

Recommended Readings:

- SOLAR paper: https://arxiv.org/abs/1808.09105
- Blog post about SOLAR from BAIR: https://bair.berkeley.edu/blog/2019/05/20/solar/

Week 12: Guest Lecture

[To Be Determined]