Chuhuan Huang

500 W University Pkwy, Baltimore, Maryland, 21210
213-512-7073 chuan129@jh.edu linkedin.com/in/chuhuan-huang-957270268/ chh172.github.io

Education

Johns Hopkins University **08/2023 – present**
Doctor of Philosophy in Mathematics *Baltimore, MD*

University of Southern California **08/2021 – 05/2023**
Master of Arts in Applied Mathematics *Los Angeles, CA*

University of California, San Diego **08/2016 – 03/2020**
Bachelor of Science in Mathematics-Computer Science *San Diego, CA*

Technical Skills

Languages: Python(more proficient), LaTeX, C++, Chinese (mother tongue), English (daily working/living language)
Relevant Courses: Stochastic Differential Equations, Probability Theory and Stochastic Processes, Real Analysis w/ Abstract Measure Theory, Functional Analysis, Applied Matrix Analysis, Convex Analysis, Mathematical Statistics, Algorithm Design and Analysis, Time Series, Abstract Algebra w/ Category Theory, Advanced Data Structure, Reinforcement Learning Theory&Algorithms
Machine Learning Frameworks: Pytorch, HuggingFace, Diffrax, Signatory
Research Interests: Neural Differential Equations, Long Time Series Modeling

Internship

Penghua Fund Management **06/2023 – 08/2023**
Quantitative Researcher *Shenzhen, CHN*

- Design and implement the framework for a Transformer-based reinforcement learning funds recommender system, together with several benchmarks.
- Design the industrial category standard transferring mapping using an Adam-trained Variational AutoEncoder; Encapsulate a ready-to-use the preprocessing workflow, Bokeh-based exploratory data analysis, and SHAP-based feature importance analysis for XGBoost-based model.

Projects

Deep RL Funds Recommender System | *independent* |*supervised by Dr. B. Kou @ Penghua Fund* **2023**

- Our core model of the recommender system is built with two standard Transformers, together with a pre-trained BERT-based preprocessing layer and a pre-trained XGBoost-based reward model, and trained in the Asynchronous Advantage Actor Critic algorithm (A3C), where one Transformer acts as the actor and the other acts as the critic.
- Our benchmark model 1 is built with a BERT-based preprocessing layer and a Wide&Deep submodel, while the benchmark model 2 is built with a OneHotEncoding preprocessing layer and a XGBoost submodel; Both are trained by an Adam Optimizer.

A Survey on the Computational Hardness of Linear-structured MDP | *Master's Thesis* **2023**

- We are investigating i) the equivalent condition for the computational-statistical gap in Reinforcement learning; ii) the relationship between the computational hardness of the linear-structured Markov decision processes and the rank of the transition matrix in the corresponding Markov chain.

Bochner Integral and its application in Stochastic Processes | *independent* | *supervised by Professor J. Zhang* **2022**

- We extended the classical Luzin theorem to Banach-valued functions in a bottom-up fashion by constructing the Banach-valued Dominated Convergence theorem, Banach-valued Egorov's theorem, and so forth.
- We used the extended Luzin to prove that for stochastic processes with the same finite distribution, their integral over a finite time interval has the same distribution.

Approaching MAX-CUT thru Reinforcement Learning | *independent* | *supervised by Professor S. Heilman* **2022**

- We approach the MAX-CUT problem by using Actor-Critic algorithm-trained LSTM-based pointer networks and compared with known Semidefinite programming benchmarks, using LSTM framework in Keras.
- Our approach is averagely 33% faster SDP benchmark in predicting and reaches 86% accuracy of the benchmark.
- Our approach is unfortunately unstable: when graphs with more than 150 vertices are fed, we sometimes run into NaN error in the training phase and sometimes poor performance in predicting phase.

Simulation of MDP and Decision-generating thru Value Iteration | *independent project* **2021**

- Modeling the transition distribution and cumulative reward in a game-related stochastic process using the Markov decision process, and implementing the policy generating mechanism through Value Iteration, a dynamic-programming-based algorithm, in Python.
- Our model successfully leads to around 500% rate of return with 2k dollar profits.