Artificial Intelligence Nanodegree Syllabus

Congratulations on considering the Artificial Intelligence Nanodegree program!

Before You Start

Educational Objectives: This program will teach you all the tools needed to succeed in your journey into the world of AI.

Make sure to set aside adequate time on your calendar for focused work. In order to succeed, we recommend having experience with intermediate Python programming (including experience with basic algorithms, common data structures, and Object Oriented Programming), and intermediate statistics & linear algebra (including discrete & continuous distributions, vector spaces & matrices).

If you'd like to prepare for this Nanodegree program, check out our Programming for Artificial Intelligence and at least one advanced Nanodegree program like the Machine Learning or Deep Learning programs.

Contact Info

While going through the program, if you have questions about anything, you can reach us at aind-support@udacity.com.

Nanodegree Program Info

This program will teach you how to become a better Artificial Intelligence or Machine Learning Engineer by teaching you classical AI algorithms applied to common problem types. You will complete projects and exercises incorporating search, optimization, planning, and probabilistic graphical models which have been used in Artificial Intelligence applications for automation, logistics, operations research, and more. These concepts form the foundation for many of the most exciting advances in AI in recent years. Each project you build will be an opportunity to demonstrate what you've learned in your lessons, and become part of a career portfolio that will demonstrate your mastery of these skills to potential employers.

This is a term-based program that requires students to keep pace with their peers. The program is delivered in 1 term spread over 3 months. On average, students will need to spend about 12-15 hours per week in order to complete all required coursework, including lecture and project time.

Length of Program: 150 Hours*
Frequency of Classes: Term-based
Textbooks required: Although there is no required textbook, the content closely follows the recommended textbook Artificial Intelligence: A Modern Approach by Stuart Russell & Peter Norvig (link) - required readings are provided in the program.
Instructional Tools Available: Video lectures, Personalized project reviews, Text instructions, Quizzes, Forum support, In-classroom mentorship

* This is an estimation of total hours the average student may take to complete all required coursework, including lecture and project time. Actual hours may vary.

Projects

This program covers classical AI techniques that you will need to master to become a better AI practitioner. Specifically, we will focus on intermediate to advanced programming skills, linear algebra, and algorithms that appear in a variety of AI applications.

One of our main goals at Udacity is to help you create a job-ready portfolio. Building a project is one of the best ways both to test the skills you've acquired and to demonstrate your newfound abilities to future employers. Throughout this Nanodegree program, you'll have the opportunity to prove your skills by building the following projects:

- Build a Sudoku Solver
- Build a Forward Planning Agent
- Build an Adversarial Game Playing Agent
- Part of Speech Tagging

In the sections below, you'll find a detailed description of each project along with the course material that presents the skills required to complete the project.

Lesson Content: Intro to Artificial Intelligence

Lesson	Learning Outcomes
Welcome to the Program	➔ Meet the course instructors and Udacity team ➔ Learn about the resources available to help you succeed
Intro to Artificial Intelligence	➔ Consider the meaning of “artificial intelligence” ➔ Be able to define core concepts from AI including “agents”, “environments”, and “states” ➔ Learn the concept of “rational” behavior for AI agents
Setting Up your Environment with Anaconda	➔ Install the software and complete necessary system configuration you’ll need for the projects

Project: Build a Sudoku Solver

Humans use reason to solve problems by decomposing the problem statement and incorporating domain knowledge to limit the possible solution space. In this project you’ll use a technique called constraint propagation together with backtracking search to make an agent that only considers reasonable solution candidates and efficiently solves any Sudoku puzzle. This approach appears in many classical AI problems, and the solution techniques have been extended and applied to diverse problems in bioinformatics, logistics, and operations research.

In this project you will demonstrate some basic algorithms knowledge, and learn to use constraint satisfaction to solve general problems.

Supporting Lesson Content: Constraint Satisfaction Problems

Lesson	Learning Outcomes
Solving Sudoku With AI	➔ Express logical constraints as Python functions ➔ Use constraint propagation & search to solve all Sudoku puzzles
Constraint Satisfaction Problems	➔ Learn to represent problems in terms of logical constraints ➔ Use constraint propagation to limit the potential solution space ➔ Incorporate backtracking search to find a solution when the set of constraints is incomplete
Additional Topics in CSP	➔ List of external resources for you to continue learning about CSPs

Project: Build a Forward Planning Agent

Intelligent agents are expected to act in complex domains where their goals and objectives may not be immediately achievable. They must reason about their goals and make rational choices of actions to achieve them. In this project you will build a system using symbolic logic to represent general problem domains and use classical search to find optimal plans for achieving your agent's goals. Planning & scheduling systems power modern automation & logistics operations, and aerospace applications like the Hubble telescope & NASA Mars rovers.

In this project you will demonstrate an understanding of classical optimization & search algorithms, symbolic logic, and domain-independent planning.

Lesson Content: Classical Search

Lesson	Learning Outcomes
Introduction	➔ Learn about the significance of search in AI
Uninformed Search	➔ Learn uninformed search techniques including depth-first order, breadth-first order, and Uniform Cost Search
Informed Search	➔ Learn informed search techniques (using heuristics) including A* ➔ Understand admissibility and consistency conditions for heuristics
Additional Topics: Search	➔ List of external resources for you to continue learning about search

Classroom Exercise: Search	➔ Implement informed & uninformed search for Pacman

Lesson Content: Optimization Problems

Lesson	Learning Outcomes
Introduction	➔ Introduce iterative improvement problems that can be solved with optimization
Hill Climbing	➔ Learn Random Hill Climbing for local search optimization problems
Simulated Annealing	➔ Learn to use Simulated Annealing for global optimization problems
Genetic Algorithms	➔ Explore and implement Genetic Algorithms that keep a pool of candidates to solve optimization problems
Additional Optimization Topics	➔ Learn about improvements & optimizations to optimization search including Late Acceptance Hill Climbing, Basin Hopping, & Differential Evolution
Classroom Exercise: Optimization Problems	➔ Compare optimization techniques on a variety of problems

Supporting Lesson Content: Automated Planning

Lesson	Learning Outcomes
Symbolic Logic & Reasoning	➔ Learn Propositional logic (propositions & statements) ➔ Learn First-Order logic (quantifiers, variables, & objects) ➔ Encode problems with symbolic constraints using first-order logic
Introduction to Automated Planning	➔ Learn to define planning problems
Classical Planning	➔ Learn high-level features of automated planning techniques using search & symbolic logic including forward planning, backwards planning, & hierarchical planning ➔ Explore planning heuristics & planning graphs
Additional Topics in Planning	➔ List of external resources for you to continue learning about planning

Project: Build an Adversarial Game Playing Agent

AI agents acting in the real world have to “hope for the best, but prepare for the worst.” In this project you will write an agent that uses that idea to make rational choices to achieve super-human performance in games competing against adversarial agents. The principles of adversarial search provide a foundation for autonomous agents acting in the real world, and for understanding modern advances in AI like DeepMind’s AlphaGo Zero.

In this project you will demonstrate advanced algorithms knowledge, including minimax with alpha-beta pruning for adversarial search.

Supporting Lesson Content: Adversarial Search

Lesson	Learning Outcomes
Search in Multi-Agent Domains	➔ Understand “adversarial” problems & applications (e.g., multi-agent environments) ➔ Extend state space search techniques to domains your agents do not fully control ➔ Learn the minimax search technique
Optimizing Minimax Search	➔ Learn techniques used to overcome limitations in basic minimax search like depth-limiting and alpha-beta pruning,
Extending Minimax Search	➔ Extend adversarial search to non-deterministic domains and domains with more than two players
Additional Adversarial Search Topics	➔ List of external resources for you to continue learning about adversarial search

Project: Part of Speech Tagging

Probabilistic models allow your agents to better handle the uncertainty of the real world by explicitly modeling their belief state as a distribution over all possible states. In this project you’ll use a Hidden Markov Model (HMM) to perform part of speech tagging, a common pre-processing step in Natural Language Processing. HMMs have been used extensively in NLP, speech recognition, bioinformatics, and computer vision tasks.

Supporting Lesson Content: Probabilistic Models & Pattern Recognition

Lesson	Learning Outcomes
Probability	➔ Review key concepts in probability including discrete distributions,

	joint probabilities, and conditional probabilities
Bayes Networks	➔ Efficiently encode joint probabilities in Bayes networks
Inference in Bayes Nets	➔ Learn about inference in Bayes networks through exact enumeration with optimizations ➔ Learn techniques for approximate inference in more complex Bayes networks
Hidden Markov Models	➔ Learn parameters to maximize the likelihood of model parameters to training data ➔ Determine the likelihood of observing test data given a fixed model ➔ Learn an algorithm to Identify the most likely sequence of states in a model given some data
Dynamic Time Warping	➔ Learn the dynamic time warping algorithm for time-series analysis
Additional Topics in PGMs	➔ List of external resources for you to continue learning about probabilistic graphical models