07-280 Learning Objectives

v1.1

Spring 2026

Course Level Learning Outcomes

1. Implement and analyze existing algorithms, including well-studied methods for heuristic-based search, constraint satisfaction problems, classification, regression, and feature learning
2. Integrate multiple facets of practical AI and ML in a single system: problem formulation, data preprocessing, learning, regularization, and algorithm/model selection
3. Describe the formal properties of algorithms and models in AI and ML, and explain the practical implications of those results
4. Describe the trade-offs between computationally cheap, less efficient/accurate methods and computationally expensive, but more accurate methods, especially as they apply to heuristic implementation and sampling techniques
5. Compare and contrast different paradigms for learning (supervised, unsupervised, etc.)
6. Design experiments to evaluate and compare different machine learning techniques on real-world problems
7. Employ probability, statistics, calculus, linear algebra, and optimization in order to develop new predictive models or learning methods
8. Given a description of an AI/ML technique, analyze it to identify (1) the expressive power of the formalism; (2) the size and complexity of the search space; (3) the computational properties of the algorithm; (4) any guarantees (or lack thereof) regarding termination, convergence, correctness, accuracy, or generalization power
9. Select and apply an appropriate supervised learning algorithm for classification and regression problems (e.g., decision trees, linear regression, logistic regression, neural networks)
10. Describe specific challenges related to the ethics of AI systems and approaches that should be used to better align AI systems with human objectives and values.

Search Fundamentals

1. Heuristic Search
    a. Understand how to formulate and define problems as search problems.
    b. Describe and implement various search algorithms.
    c. Understand how search algorithms run on a given problem.
    d. Analyze the time and space complexity of a variety of search algorithms.
    e. Verify or create admissible/consistent heuristics for a given search problem
    f. Describe the trade-offs between computationally cheap, inaccurate heuristics and computationally expensive heuristics

    g. Understand both the algorithmic differences and the distinct use cases for tree search vs graph search
    h. Discuss the ethical implications of a given real-world application involving search algorithms
2. Adversarial Search
    a. Formulate a real-world problem as a search problem over an adversarial search tree
    b. Trace and implement algorithms for game trees with a mixture of node types (min/max/expectation) given a start state, player list, actions, and utilities/rewards
    c. Describe the bounded lookahead algorithm
    d. Define and create evaluation functions
    e. Analyze a general game tree to determine which nodes can be pruned without causing an impact on the optimal solution
3. Constraint Satisfaction Problems
    a. Describe the definition of a CSP problem and its connection with general search problems
    b. Formulate a real-world problem as a CSP
    c. Describe and implement the backtracking algorithm
    d. Define arc consistency
    e. Describe and implement forward checking and AC-3
    f. Explain the differences between MRV and LCV heuristics

ML Fundamentals

4. ML Overview
    a. Formulate a well-posed learning problem for a real-world task by identifying the task, performance measure, and training experience
    b. Reason about the decision boundary for a classification algorithm (e.g., compare decision trees and logistic regression)
5. Decision Trees
    a. Formalize a learning problem by identifying the input space, output space, hypothesis, and hypothesis space
    b. Implement decision tree training and prediction
    c. Use effective splitting criteria for decision trees and be able to define entropy, conditional entropy, and mutual information
    d. Judge whether a decision tree is underfitting or overfitting
    e. Explain how early stopping method affects overfitting in decision tree learning
    f. Describe the greedy nature of decision tree algorithms and the implications of achieving an optimal solution
    g. Explain how decision trees could be used for regression tasks as well as classification tasks
6. Optimization for ML
    a. Identify the relationships between the concepts of loss, risk, empirical risk, and empirical risk minimization as they relate to finding a hypothesis function / model in machine learning
    b. Given a loss function and hypothesis function structure (e.g., linear regression structure), formulate the machine learning optimization problem

    c. Explain why there could be a need for more general loss functions than just zero-one loss for classification tasks
    d. Demonstrate how squared error loss and mean squared error are related to empirical risk minimization for regression
    e. Apply (batch) gradient descent, stochastic gradient descent, and mini-batch gradient descent to optimize an objective function
    f. Apply knowledge of zero derivatives to identify a closed-form solution (if one exists) to an optimization problem
    g. Distinguish between convex and nonconvex functions and how they are related to various machine learning models
    h. Explain the issues with optimizing non-convex functions
2. Calculus
    a. Derive the gradient of a differentiable, multi-variable objective function by applying knowledge of multivariable calculus, including:
        i. Partial derivatives involving scalars, vectors, and matrices
        ii. Denominator layout of partial derivatives
        iii. Multivariable chain rule
    b. Derive matrix calculus properties by expanding linear algebra into scalar form as needed
3. Linear Regression
    a. Implement learning for linear regression using four optimization techniques: (1) closed form, (2) (batch) gradient descent, (3) stochastic gradient descent, and mini-batch stochastic gradient descent
    b. Analyze the tradeoffs of computational complexity vs. convergence speed for linear regression optimization techniques on a particular dataset
    c. Identify properties of datasets that affect the number of unique solutions to linear regression
4. Logistic Regression
    a. Describe the differences between linear and logistic regression, including modeling, optimization, and application
    b. Implement logistic regression for binary or multi-class classification
    c. Explain the differences and similarities between binary and multi-class logistic regression models and loss functions
5. Model Selection
    a. Plan an experiment that uses training, validation, and test datasets to predict the performance of a classifier on unseen data (without cheating)
    b. Explain the difference between (1) training error, (2) cross-validation error, and (3) test error
    c. For a given learning technique, identify the model, learning algorithm, parameters, and hyperparameters
    d. Describe trade-offs between different options for cross-validation
6. Feature Engineering and Regularization
    a. Engineer appropriate features for a new task
    b. Convert a non-linear dataset (or linearly inseparable dataset) to a linear dataset (or linearly separable dataset) in higher dimensions
    c. Explain why linear models are still considered linear despite their ability to be applied to tasks with highly non-linear datasets
    d. Identify symptoms that indicate that a model is overfitting

    e. Add a regularizer to an existing objective in order to combat overfitting
7. Neural Networks
    a. Explain the biological motivations for a neural network
    b. Combine simpler models (e.g., linear regression, binary logistic regression, multi-class logistic regression) as components to build up feed-forward neural network architectures
    c. Explain the reasons why a neural network can model nonlinear decision boundaries for classification
    d. Compare and contrast feature engineering with learning features
    e. Identify (some of) the options available when designing the architecture of a neural network
    f. Implement and train a feed-forward neural network from scratch, including training with the backpropagation algorithm
    g. Instantiate an optimization method (e.g., SGD) when the parameters of a model are comprised of several matrices corresponding to different layers of a neural network

AI Alignment

1. Describe the goal of AI alignment and distinguish between different types of alignment (e.g., user goals vs. broader human values).
2. Identify and explain key challenges in aligning AI systems (e.g., reward hacking, distribution shift, difficulty of specification and evaluation).
3. Compare general approaches to improving alignment (e.g., post-training with feedback, guardrails, interpretability, improved reward design), including their strengths and limitations.

Building AlexNet

1. Neural Network Applications
    a. Combine various layers and loss functions to create, train, and deploy task-specific neural network architectures, such as convolutional neural networks and autoencoders
    b. Identify (some of) the options available when designing the architecture of a neural network
    c. Utilize pre-trained parameters of a neural network to fine-tune training a new network such that it can be applied to a different but related task
    d. Leverage functionality within a deep learning toolkit, such as PyTorch, to train and deploy a neural network for a specific task
2. Computer Vision
    a. Describe the potential benefits of using convolutional layers rather than fully connected layers
    b. Describe the parameters used in convolutional neural networks
    c. Explain how pooling and kernel stride length can reduce the number of values as the network gets deeper
    d. Understand the reasons for using additional techniques in deep learning, including residual connections, normalization, and dropout
    e. Discuss the impact of AlexNet on AI
3. Parallel Processing

    a. Describe why GPUs can outperform CPUs for neural network training and inference
    b. Discuss how training data and neural network models can be split across multiple GPUs.

MLE and Markov Chains

1. MLE
    a. Recall probability basics, including but not limited to: discrete and continuous random variables, probability mass functions, probability density functions, events vs. random variables, expectation and variance, joint probability distributions, marginal probabilities, conditional probabilities, independence, and conditional independence
    b. Describe common probability distributions such as Bernoulli, Categorical, Gaussian, etc.
    c. State the principle of maximum likelihood estimation and explain what it tries to accomplish
    d. State the principle of maximum a posteriori estimation and explain why we use it
    e. Derive the MLE parameters of a simple model in closed form
    f. Define maximum conditional likelihood estimation (MCLE) and describe how it differs from MLE
    g. Provide a probabilistic interpretation of linear regression
    h. For linear regression, show that the parameters that minimize squared error are equivalent to those that maximize conditional likelihood
    i. Explain the practical reasons why we work with the log of the likelihood
2. Markov Chains and N-grams
    a. Express an n-gram language model as a markov chain probability distribution
    b. Connect the frequency counts of N-grams with the solution to maximizing the log likelihood of the corpus

Building GPT2

1. Language Models
    a. Build a language model using word embeddings by formulating the objective using token encoding and decoding linear layers, deriving the gradient update equations, and training the model in a self-supervised manner
    b. Adjust basic language models (n-grams and word embeddings) to use longer context than just a single token and discuss the impact of increased context size on model size, training data, and model performance.
    c. Describe byte pair encoding tokenization as it compares to character- and word-level tokenization
    d. Implement positional encoding to help language models differentiate between different orders of tokens in the input context.
    e. Given an input token context, generate the next token by sampling from the probabilities generated by a trained language model using additional techniques, including top-k and temperature sampling.
    f. Compare different techniques for basic language models, specifically n-grams, word embeddings, and attention

    g. Describe the motivation for using attention mechanisms rather than a uniform combination of input content embeddings
    h. Describe the motivation for using multiple attention heads rather than a single attention head
    i. Give various examples of the role of linear operations in GPT language models
    j. Identify the size of and difference between parameters and computed values (activations) in various components within transformer models, especially as the context size changes.
    k. Implement and train a transformer language model with multiple attention heads and multiple transformer blocks.

Markov Decision Processes

1. Describe the definition of Markov Decision Process
2. Compute the utility of a reward sequence given a discount factor
3. Define policy and optimal policy of an MDP
4. Define the state-values of an MDP
5. Define the Q-values of an MDP
6. Write the Bellman Equation for state-value and Q-value for optimal policy and a given policy
7. Describe and implement the value iteration algorithm (through Bellman update) for solving MDPs
8. Describe and implement policy iteration algorithm (through policy evaluation and policy improvement) for solving MDPs
9. Understand convergence for value iteration and policy iteration

Reinforcement Learning

1. Understand the concept of exploration, exploitation, and regret
2. Describe the relationships and differences between:
    a. Markov Decision Processes (MDP) vs Reinforcement Learning (RL)
    b. Model-based vs Model-free RL
    c. Temporal-Difference Value Learning (TD Value Learning) vs Q-Learning
    d. Passive vs Active RL
    e. Off-policy vs On-policy Learning
    f. Exploration vs Exploitation
3. Describe and implement:
    a. Temporal difference learning
    b. Q-Learning
    c. ϵ-Greedy algorithm
    d. Approximate Q-learning (Feature-based)
4. Derive the weight update for Approximate Q-learning

Building AlphaGo

1. Explain how a neural network can be trained and used in approximate Q-learning

2. Describe the practical changes that need to be made to stabilize the training of Deep Q Networks (DQN), specifically, experience replay, and having a target network copy.
3. Explain the trade-offs along the spectrum of only using Monte Carlo rollouts to exhaustive game tree search, and show how Monte Carlo tree search (MCTS) fits into this spectrum.
4. Implement the MCTS algorithm and integrate it into an autonomous game agent
5. Convert a MCTS game agent implementation into an AlphaZero game agent, and train the AlphaZero network after collecting experience samples from self-play.

LLM Post Training

1. Explain the limitations of a pre-trained large language model when applied to interacting with humans.
2. Describe how the supervised fine-tuning (SFT) including instruction tuning, RL with human feedback (RLHF), and RL with Verifiable Rewards (RLVR) techniques can help improve the pre-trained (base) model to follow human instructions and have better alignment and better performance.

References

Built from 10-301/601 Learning objectives