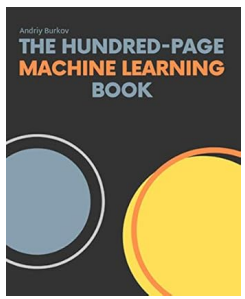
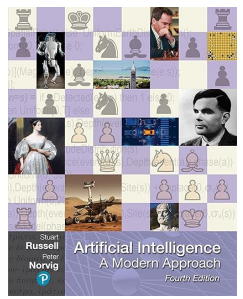
Lesson 02.2: Machine Learning Paradigms

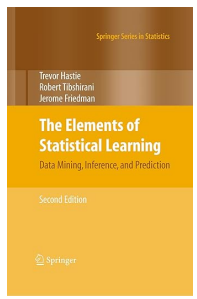
Instructor: Dr. GP Saggese - gsaggese@umd.edu

References:

- Burkov: *"The Hundred-Page Machine Learning Book"* (2019)
- Russell et al.: *"Artificial Intelligence: A Modern Approach"* (4th ed, 2020)
- Hastie et al.: *"The Elements of Statistical Learning"* (2nd ed, 2009)







- ***Machine Learning Paradigms***
  - Major Paradigms
  - Some Examples
  - Machine Learning in Practice
  - Pipeline Organization

- Machine Learning Paradigms
  - ***Major Paradigms***
  - Some Examples
  - Machine Learning in Practice
  - Pipeline Organization

Machine Learning Paradigms with Examples (1/3)

- **Supervised Learning**
  - Learn from labeled data to predict labels for new inputs
  - E.g., image classification using ResNet on ImageNet
- **Unsupervised Learning**
  - Discover hidden patterns or structure in unlabeled data
  - E.g., K-means clustering for customer segmentation
- **Reinforcement Learning**
  - Learn through interaction with an environment, receiving rewards/punishments
  - E.g., deep Q-Learning for playing Atari games
- **Self-Supervised Learning**
  - Generate pseudo-labels from unlabeled data to pre-train models
  - E.g., BERT (Masked Language Modeling)
- **Semi-Supervised Learning**
  - Combine small labeled data with large unlabeled data to improve performance
  - E.g., named entity recognition (NER) using annotated sentences with entity tags combined with many raw text documents

Machine Learning Paradigms with Examples (2/3)

- **Online Learning**
  - Learn incrementally from a stream of data in real time
  - E.g., online logistic regression for click-through rate prediction
- **Multi-Task Learning**
  - Train simultaneously a model to perform multiple related tasks
  - E.g., learn sentiment analysis and question answering
- **Meta-Learning**
  - "Learning to learn": adapt quickly to new tasks using prior experience
  - E.g., a model can be fine-tuned quickly on a new task using just a few gradient steps
- **Zero-Shot / Few-Shot Learning**
  - Generalize to new tasks with no or few labeled examples
  - E.g., GPT-4 solving tasks with zero-shot prompting
- **Active Learning**
  - The model selects the most informative samples to be labeled by an oracle (e.g., a human)
  - E.g., pick samples where the model is least confident to get more examples

Machine Learning Paradigms with Examples (3/3)

- **Federated Learning**
  - Train models across decentralized devices without sharing raw data
  - E.g., fraud detection or credit scoring across banks
- **Evolutionary Learning**
  - Optimize model structures or parameters using evolutionary algorithms inspired by natural selection and genetics
  - Gradient free, global search, variable length inputs
  - E.g., genetic algorithms
- **Curriculum Learning**
  - Train models on easier tasks first, gradually increasing difficulty
  - E.g., training in robotic control simulations
- **Multi-Agent Learning**
  - Multiple agents learn and interact in shared environments, often in game-theoretic settings (e.g., competition, collaboration)
  - E.g., AlphaStar to play StarCraft II

- Machine Learning Paradigms
  - Major Paradigms
  - ***Some Examples***
  - Machine Learning in Practice
  - Pipeline Organization

Supervised Learning

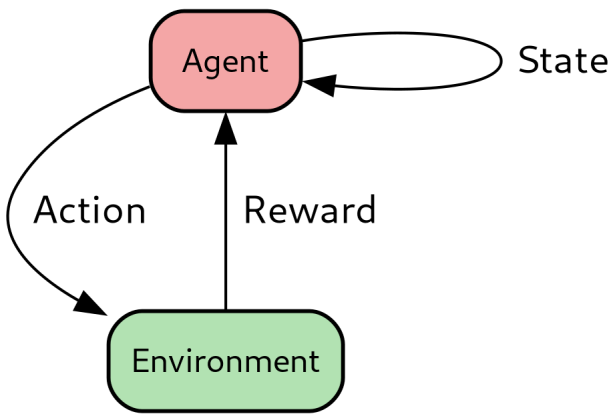
- Learn a function $f : X \to Y$ that maps inputs to correct outputs
  - Training examples $(\underline{\boldsymbol{x}}, y)$ with pairs of inputs and correct outputs
  - Requires labeled data for training
  - Measure performance with error on a separate test set
- **Classification**: output is a discrete label, e.g.,
  - `Spam` vs `Not Spam`
  - Digit recognition `0`, `1`, ...
  - Sentiment analysis `Pos`, `Neg`, `Neutral`
- **Regression**: output is a continuous value, e.g.,
  - House prices given features like size and location
  - House demand
  - Stock prices
- **Common algorithms**:
  - Linear Regression
  - Decision Trees
  - K-nearest neighbors
  - Neural Networks
  - ...

Unsupervised Learning

- Learn from data **without** labeled outputs
  - Goal: discover patterns or structure in the data
  - No explicit feedback signal
  - Evaluation can be qualitative
- **Main techniques**:
  - **Clustering**: Group similar examples, e.g.,
    - Customer segmentation
    - Grouping news articles by topic without knowing the topics
  - **Dimensionality Reduction**: Reduce number of variables with PCA while preserving structure
    - E.g., visualize high-dimensional data in 2D
  - **Density Estimation**: Estimate probability distribution of data
    - E.g., anomaly detection in server logs
  - **Association Rule Learning**: Discover interesting relations between variables
    - E.g., market basket analysis (e.g., "people who buy X also buy Y")
- **Common algorithms:**
  - K-means
  - PCA
  - Autoencoders

Reinforcement Learning

- Learn by **interacting with an environment** to **maximize cumulative reward**
  - Learn policy $\pi(s) \rightarrow a$ that maximizes expected reward
  - Trade-off between exploration (trying new actions) and exploitation (using known good actions)
  - Environments provide clear rules and feedback (win/loss/reward)
  - Often involve physical simulation or real-world interaction
- **Core elements:**
  - Agent: Learner and decision maker
  - Environment: Everything the agent interacts with
  - State s
  - Action a
  - Reward r
- **Algorithms:**
  - Q-learning
  - Policy Gradient methods



Reinforcement Learning: Examples

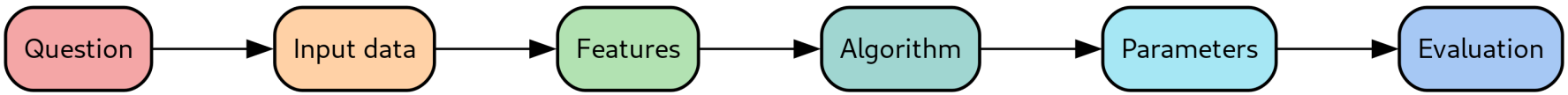
- In **game playing**, learn strategies through trial and error
  - E.g., AlphaGo mastering the game of Go
- In **robotics**, learn control policies for movement and manipulation
- In **autonomous driving**, learn safe and efficient driving behaviors
- In **resource management**, optimize allocation of limited resources over time
  - E.g., data center cooling or CPU job scheduling
- In **personalized recommendations**, adapt suggestions based on user interaction
  - E.g., newsfeed ranking adjusting based on user clicks
- In **healthcare**, optimize treatment plans over time

- Machine Learning Paradigms
  - Major Paradigms
  - Some Examples
  - ***Machine Learning in Practice***
  - Pipeline Organization

Machine Learning Flow (1/2)

- **Question**
  - E.g., "How can we predict house prices?"
- **Input data**
  - E.g., historical data of house sales
- **Features**
  - E.g., number of bedrooms, location, square footage
- **Algorithm**
  - E.g., linear regression, decision trees
- **Parameters**
  - E.g., learning rate, number of trees in a random forest
- **Evaluation**
  - E.g., accuracy, precision, recall

Machine Learning Flow (2/2)



- **Not all phases are equally important!**
  - Question > Data > Features > Algorithm
- Clarity of the **question** impacts project success
- Quality and relevance of **data** are crucial for performance
- Proper **feature** selection simplifies the model and improves accuracy
- **Algorithm** is often less important (contrary to popular belief!)

Question

- **Make the question concrete and precise**
  - Define the problem clearly
  - Specify inputs and expected outputs
  - Align question with business or research objectives
  - E.g.,:
    - **Bad**: *"How can we improve sales?"*
    - **Good**: *"What factors most significantly impact sales of product X in region Y during season Z?"*
- Formulating question is **the most important part** of the machine learning problem
  - Misunderstanding leads to:
    - Solving the wrong problem
    - Collecting wrong data
    - ...
- *"If I were given one hour to save the planet, I would spend 59 minutes defining the problem and one minute resolving it"* (Einstein)

Input Data

- Ensure **data is specific to prediction** goal
  - E.g., use known movie ratings to predict unseen movie ratings from the same population
  - Training set $\approx$ test set
- Relationship between data and prediction goal is **not always direct**
  - E.g., interested in prices but predict supply and demand instead
- Poor-quality data leads to inaccurate predictions
  - *"Garbage in - garbage out"*
- Recognize **when data is insufficient** for valid answers
  - *"Combination of data and desire for an answer does not ensure a reasonable answer can be extracted"* (John Tukey)
- **More data vs better models**
  - Meta-studies show difference between generic and best model is like 5%
  - *"It's not who has the best algorithm that wins. It's who has the most data"* (Google researcher)
  - *"Every time I fire a linguist, the performance of the speech recognizer goes up"* (IBM researcher in speech recognition)

Features

- **Features** provide high-level information about inputs
  - E.g., use intensity and symmetry for scanned numbers instead of raw bit maps
- **Characteristics of good features**:
  1. Perform data compression
  2. Retain relevant information
  3. Often created with expert knowledge
- **Common mistakes in feature building**:
  1. Automating feature selection may lead to overfitting
     - Black box predictions can be accurate but stop working anytime
     - E.g., Google Flu's unclear feature-model link
  2. Ignoring data-specific quirks
     - E.g., mislabeling outliers
  3. Unnecessarily discarding information

Models

- Best models are:
  - **Interpretable**
    - Allow users to understand and trust the model's decisions
    - E.g., decision trees are appropriate in medical studies since they produce a "reasoning"
  - **Simple**
    - Easier to implement and maintain
    - Reduces the risk of overfitting
  - **Accurate**
    - Often accuracy is traded off for remaining characteristics
    - E.g., accuracy vs interpretability, accuracy vs speed
  - **Fast**
    - Reduce computational costs to train and test
    - E.g., real-time applications
  - **Scalable**
    - Can handle large datasets efficiently (growing data and user bases)
    - E.g., in the Netflix prize, Netflix didn't end up implementing the best algorithm since it wasn't scalable enough

- Machine Learning Paradigms
  - Major Paradigms
  - Some Examples
  - Machine Learning in Practice
  - ***Pipeline Organization***

How Are Machine Learning Systems Organized?

- Machine learning systems organized in a **pipeline**
  1. **Break down** problem into sub-problems
  2. **Solve** problems one at a time
  3. **Combine** solutions to solve initial problem
- Performance p of a pipeline:

$$p_{system} = \sum_i p_i \cdot \alpha_i$$

where:

- p_i: Performance of each stage
- α_i: Importance of each stage

Example of Photo OCR System

- **Goal**: build systems to read text in a picture
  - OCR = "Optical Character Recognition"
- **Stages of ML pipeline for OCR:**
  - Text detection: find text areas in the picture
  - Character segmentation: split text into letter boxes
    - E.g., `h e l l o`
  - Character classification: classify characters individually
  - Spelling correction: fix text errors using context
    - E.g., `hell0` corrected to `hello`
- **Sliding window approach**
  - **Problem**: Unknown text location and size
  - Text detection
    - Train classifier to recognize letters vs non-letters
    - Scan image in two directions with different sizes for text
    - Evaluate classifier cheaper than training
    - Create text likelihood map (e.g., heatmap) using classifier probabilities
    - Enclose text areas in boxes
    - Discard boxes not fitting aspect ratio (valid text width $>$ height)
  - Character segmentation
    - Use sliding window classifiers to find "breaks" between characters

Getting More Data

- The ideal recipe for ML is: "low-bias algorithm + train on massive amount of data"
- Use learning curves to make sure we are taking advantage of more data
- The **problem** is getting large amount of data
- Always ask yourself: *"how much work is to get 10x more data than we currently have?"*
- Often it is not that difficult:
  1. Artificial data
     - E.g., synthesize or amplify data set
  2. Collect and label by hand
     - E.g., crowd sourcing like Amazon Mechanical Turk

OCR Pipeline: Example of Artificial Data Synthesis

- How can we increase data set size?
  1. Synthesize data set
     - Use font libraries to generate large training sets
     - Paste characters against random backgrounds
     - Apply scaling, distortion, adding noise, etc
  2. Amplify a data set
     - Add examples by warping/distorting existing examples
- Transformations and noise should be specific to the application domain
  - E.g., Gaussian noise is not always appropriate

Ceiling Analysis for ML Pipeline

- The most valuable resource is time
  - You work on optimization for months
  - Find out that the optimization doesn't make much difference
- **Problem**: On which part of the pipeline should time/resource be spent?
- **Solution**: Ceiling analysis to Analyze pipeline performance
  - Single number for system performance
    - E.g., accuracy for OCR
  - For each component:
    - Mock component with "oracle" (always give correct output)
    - Leave others untouched
    - Compute pipeline performance
  - Identify critical components by estimating performance improvement with 10% component improvement
  - Measure, don't guess!