CS-470: Machine Learning

Week 1 — Introduction to Machine Learning

Instructor: Dr. Sajjad Hussain

Department of Electrical and Computer Engineering
SEECS, NUST

September 11, 2025

Outline

1. Course Logistics
2. Introduction to Machine Learning
3. Types of Machine Learning
4. Challenges of Machine Learning
5. Summary

Course Logistics

- Weekly lectures accompanied by a programming tutorial session
- Assessments include assignments, quizzes, a mid-semester exam, and a final exam
- Strongly encouraged to complete the Coursera *Machine Learning Specialization* by Andrew Ng

Week-01 Python Tutorials

- A Python tutorial notebook has been provided on the course LMS
- Covers basic Python concepts and syntax
- Intended for hands-on practice alongside lectures
- Please download the notebook and complete the exercises
- Bring any questions to the next class for discussion

Evaluation Criteria

Tentative

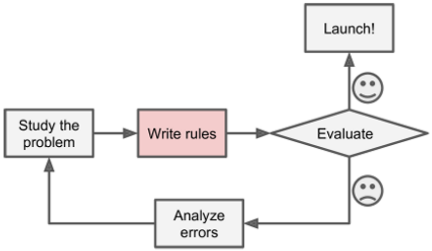
- Assignments: 20%
- Quizzes: 10%
- Midterm: 30%
- Final: 40%

What is Machine Learning?

- Field of study that gives computers the ability to learn without being explicitly programmed. —Arthur Samuel, 1959.
- Focuses on building models that generalize from data
- A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. —Tom Mitchell, 1997.

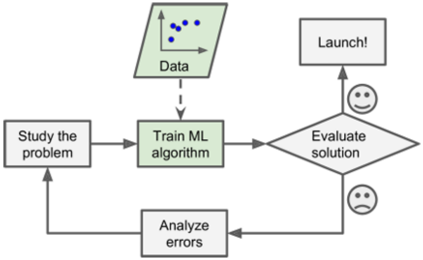
Traditional vs Machine Learning Approach

Traditional Programming



- Hand-coded logic
- Works well for deterministic tasks

Machine Learning



- Learns patterns automatically
- Works well for complex tasks

Source: A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (O'Reilly).

When to Use Machine Learning?

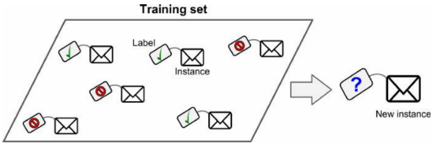
- Tasks where traditional solutions need extensive hand-tuning or rule lists (e.g., spam email filtering)
- Complex problems with no effective traditional solution (e.g., speech recognition)
- Dynamic or changing environments where systems must adapt to new data
- Extracting insights from complex problems or large datasets

Real-World Applications

- Email spam filtering
- Recommendation systems (Netflix, YouTube)
- Medical diagnosis and drug discovery
- Self-driving cars

Supervised Learning

- In supervised learning, training data includes the correct answers, called *labels*.
- Types of supervised algorithms include
    - k-Nearest Neighbors
    - Linear Regression
    - Logistic Regression
    - Support Vector Machines (SVMs)
    - Decision Trees and Random Forests
    - Neural networks



Classification vs Regression

Classification

- Predicts **discrete categories/classes**
- Output is a **label**
- Examples:
  - Spam vs Not Spam
  - Cat vs Dog image
  - Disease: Positive vs Negative
- Evaluation metrics: Accuracy, Precision, Recall, F1-score

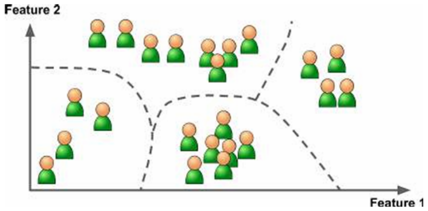
Regression

- Predicts **continuous numeric values**
- Output is a **real number**
- Examples:
  - Predicting house prices
  - Forecasting temperature
  - Predicting stock prices
- Evaluation metrics: MSE, RMSE, MAE, R^2

Both are forms of **Supervised Learning** (learn from labeled data).

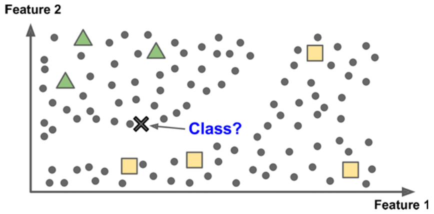
Unsupervised Learning

- In unsupervised learning, the training data is *unlabeled*, and the system learns patterns without guidance.
- Common unsupervised learning algorithms:
  - Clustering
  - Anomaly detection and novelty detection
  - Visualization and dimensionality reduction (PCA, t-SNE)
  - Association rule learning



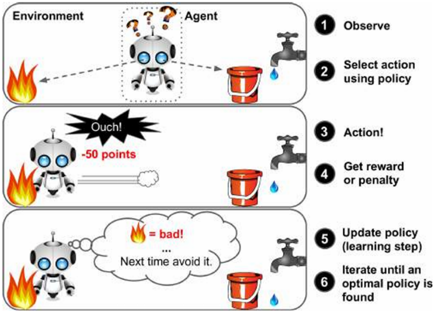
Semi-supervised Learning

- Semi-supervised learning combines aspects of supervised and unsupervised learning.
- A small portion of the data is *labeled*, and the system uses it to automatically label the much larger *unlabeled* portion.
- Common applications include Google Photos and news article categorization.



Reinforcement Learning

- In reinforcement learning, an *agent* interacts with an *environment*, takes actions, and receives *rewards* or *penalties* as feedback.
- The agent learns an optimal *policy*, which is a strategy for choosing actions that maximizes the total reward over time.


Environment
Agent
Ouch!
-50 points
= bad!
...
Next time avoid it.
1 Observe
2 Select action using policy
3 Action!
4 Get reward or penalty
5 Update policy (learning step)
6 Iterate until an optimal policy is found

Example Datasets

- Supervised: Housing prices, image classification
- Unsupervised: Customer segmentation, clustering news articles
- Reinforcement: Game playing, robotic control

Insufficient Quantity of Training Data

- Machine learning models need large amounts of data for effective training.
- Simple tasks may require thousands of examples, while complex tasks (e.g., speech recognition) may need millions.
- A well-known Microsoft study showed that even simple models can achieve strong performance on complex tasks when trained on very large datasets.

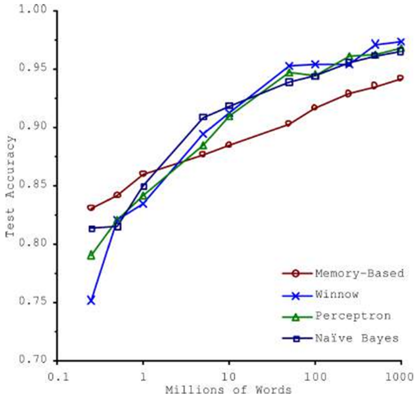


Figure: The importance of Data vs Algorithms

Non-representative Training Data

- To generalize well, training data must reflect the real-world cases we want to predict.
- Non-representative data leads to models that perform poorly on unseen data.
- **Sampling Noise:** Small training sets may miss important patterns and fail to generalize.
- **Sampling Bias:** Even large datasets can be misleading if collected using flawed methods.
- *Example:* In the 1936 U.S. election, a survey by *Literary Digest* predicted a Republican win, but it over-sampled wealthy individuals. Roosevelt (Democrat) won decisively.

Poor-Quality Data

- If training data contains errors, outliers, or noise (e.g., due to poor measurements), the model will struggle to detect underlying patterns and perform well.
- Cleaning and preprocessing data is often time-consuming but crucial. Most data scientists spend a large part of their time on this step.
- **Handling poor-quality data:**
    - Remove or correct obvious outliers.
    - Handle missing values:
        - Ignore the affected feature altogether.
        - Drop instances with missing values.
        - Fill missing values (e.g., with median or mean).
        - Train separate models with and without the feature.

Irrelevant Features

- As the saying goes: *garbage in, garbage out.*
- A model can only learn well if the training data includes enough **relevant** features and avoids too many **irrelevant** ones.
- Designing a good set of features (**feature engineering**) is critical for ML success.
- **Feature engineering involves:**
  - **Feature selection:** Choosing the most useful existing features.
  - **Feature extraction:** Combining or transforming features (e.g., using dimensionality reduction).
  - **Creating new features:** Collecting or deriving additional data.

Overfitting the Training Data

- **Overfitting:** When a model performs well on training data but fails to generalize to unseen data.
- Happens when the model is **too complex** relative to the size or noisiness of the training set.
- Complex models can pick up on **random noise or irrelevant patterns**, mistaking them for real patterns.
- Root cause: Overgeneralizing from limited or noisy data.

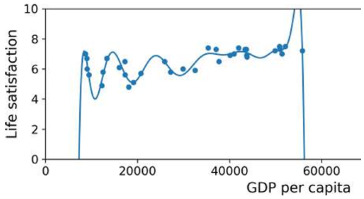


Figure: Overfitting the training data

Preventing Overfitting

- **Simplify the model:**
  - Use fewer parameters (e.g., linear vs high-degree polynomial model)
  - Reduce number of features
  - Apply constraints (regularization)
- **Improve training data:**
  - Collect more data
  - Clean data to remove noise/outliers
- **Regularization:** Adds constraints to keep model weights small
  - Balances fit vs. simplicity
  - Controlled by a **hyperparameter** (set before training)

Underfitting the Training Data

- **Underfitting:** When a model is **too simple** to capture the underlying structure of the data.
- Performs poorly even on the training data (and thus also on new data).
- Example: A simple linear model may underfit if the real relationship is more complex.
- **How to address underfitting:**
  - Use a more powerful model (with more parameters)
  - Provide better features (*feature engineering*)
  - Reduce constraints (e.g., lower regularization)

Testing and Validating Models

- The only way to know how well a model generalizes is to test it on **new data**.
- Deploying directly to production is risky because poor models will impact users.
- Better approach: **split data** into
  - **Training set**: used to train the model
  - **Test set**: used to estimate generalization error (out-of-sample error)
- If training error is low but test error is high, the model is **overfitting**.
- Typical split is **80% training and 20% testing** (depends on dataset size)

Hyperparameter Tuning and Model Selection

- Directly using the test set to pick models can lead to overfitting on the test set
- **Holdout validation:**
    - Split data into training, validation, and test sets
    - Train models with different hyperparameters on training set
    - Select the model that performs best on validation set
    - Retrain the best model on full training data, then evaluate on test set
- **Cross-validation:**
    - Split data into multiple folds
    - Train and evaluate on different folds, average the scores
    - Gives a more reliable performance estimate when data is limited
- Helps choose model type and hyperparameter values while reducing overfitting risk

No Free Lunch Theorem

- Models simplify observations by discarding details unlikely to generalize
- Choosing what to keep/discard requires assumptions about the data
- Example: Linear models assume data is fundamentally linear, and deviations are noise
- **No Free Lunch (NFL) Theorem — David Wolpert (1996):**
  - Without assumptions, no model is better than any other
  - The best model depends on the dataset
  - No single model works best for all problems
- In practice:
  - Make reasonable assumptions about the data
  - Evaluate a few suitable models (e.g., linear models for simple tasks, neural networks for complex tasks)

Summary of Week 1

- Defined Machine Learning and its real-world importance
- Discussed main types of ML: supervised, unsupervised, and reinforcement learning
- Reviewed key ML challenges: data quality, irrelevant features, underfitting, overfitting
- Covered model evaluation: train/validation/test sets, generalization error
- Introduced hyperparameter tuning, model selection, and the No Free Lunch theorem

Takeaway: Strong data, thoughtful model design, and rigorous evaluation are essential for ML success.