Module 1: Introduction and Framing

Overview

This module establishes the foundational framework for understanding machine learning. At its core, machine learning is about learning functions from data—given examples of inputs and outputs, we want to discover a mapping that can predict outputs for new, unseen inputs. This module introduces the key vocabulary and mental models you will use throughout the course.

---

1. Functions and Models

What is a Function?

In mathematics, a function maps inputs to outputs: f(x) = y. In machine learning, we rarely know the true function. Instead, we observe input-output pairs and try to learn an approximation.

Example: - Input: A photo of an animal - Output: "cat" or "dog" - Function: The unknown rule that maps pixels to labels

What is a Model?

A model is our hypothesis for what the true function might be. It has: - **Structure**: The form of the function (linear, polynomial, neural network) - **Parameters**: Adjustable values learned from data (weights, biases)

A simple model: `y = w * x + b` where w and b are parameters we learn.

The Learning Process

1. Start with a model structure (hypothesis class)
2. Initialize parameters randomly
3. Measure how wrong the model is (loss)
4. Adjust parameters to reduce loss
5. Repeat until satisfied

---

2. Generalization: The Central Goal

Why Generalization Matters

The purpose of machine learning is not to memorize training examples—it's to perform well on **new, unseen data**. This ability is called generalization.

A model that only works on training data is useless in practice. We need models that have learned the underlying pattern, not the specific noise or quirks of our training set.

Training vs. Test Performance

- **Training performance**: How well the model does on data it learned from
- **Test performance**: How well the model does on held-out data

A good model has similar training and test performance. A large gap indicates problems.

---

3. Overfitting and Underfitting

Overfitting

Overfitting occurs when a model learns the training data too well, capturing noise rather than signal. Signs of overfitting: - Very low training error - Much higher test error - Model is overly complex for the data

Analogy: A student who memorizes answers without understanding concepts fails new exam questions.

Underfitting

Underfitting occurs when a model is too simple to capture the underlying pattern. Signs: - High training error - High test error - Model cannot represent the true relationship

Analogy: Using a straight line to fit data that follows a curve.

The Bias-Variance Tradeoff

- **High bias (underfitting)**: Model is too simple, makes strong assumptions
- **High variance (overfitting)**: Model is too flexible, fits noise

The goal is to find the right complexity—enough to capture the pattern, not so much that we fit noise.

---

4. Train/Test Split

Why Split the Data?

To estimate how well our model generalizes, we need data it hasn't seen during training. We split our data into:

- **Training set (70-80%)**: Used to learn model parameters
- **Test set (20-30%)**: Held out for final evaluation

Important Rules

1. **Never train on test data**: The test set must remain unseen during training
2. **Split before preprocessing**: Avoid data leakage
3. **Randomize the split**: Unless data has temporal ordering

4. **Sufficient test size**: Need enough examples for reliable estimates

Validation Sets

For model selection (choosing hyperparameters), we often use a three-way split: - Training set: Fit models - Validation set: Choose best model - Test set: Final evaluation

---

5. Evaluation Metrics

For Regression (Continuous Outputs)

Mean Squared Error (MSE):

$$MSE = (1/n) * \Sigma(prediction - actual)^2$$

- Always positive
- Penalizes large errors heavily (due to squaring)
- Same units as the target, squared

Mean Absolute Error (MAE):

$$MAE = (1/n) * \Sigma|prediction - actual|$$

- More robust to outliers than MSE
- Same units as the target

For Classification (Discrete Outputs)

Accuracy:

$$Accuracy = (correct\ predictions) / (total\ predictions)$$

- Simple and intuitive
- Can be misleading for imbalanced classes

---

6. Baselines

What is a Baseline?

A baseline is a simple, often naive model that sets a minimum performance bar. If your sophisticated model can't beat a baseline, something is wrong.

Common Baselines

For Regression: - Predict the mean of training targets - Predict the previous value (time series)

For Classification: - Predict the most common class - Random prediction according to class frequencies

Why Baselines Matter

- Sanity check your pipeline
- Establish what “good” performance means
- Sometimes simple is good enough

---

7. Framing ML Problems

Key Questions to Answer

1. **What are the inputs?** What information is available for prediction?
2. **What is the output?** What are we trying to predict?
3. **What is the evaluation metric?** How do we measure success?
4. **What is the baseline?** What should we beat?
5. **What data is available?** How much? How clean?

Example: House Price Prediction

- **Input**: Square footage, bedrooms, location, age
- **Output**: Sale price (continuous regression)
- **Metric**: Mean Absolute Error in dollars
- **Baseline**: Predict mean price ($350,000)
- **Data**: 10,000 historical sales

Key Takeaways

1. **ML is function learning**: We learn f(x) = y from examples
2. **Models have structure and parameters**: Structure is chosen; parameters are learned
3. **Generalization is the goal**: Performance on new data matters most
4. **Overfitting is the enemy**: Monitor training vs. test gap
5. **Train/test split is essential**: Never evaluate on training data
6. **Baselines set expectations**: Always compare to simple alternatives
7. **Framing matters**: Clearly define inputs, outputs, and metrics

---

Connections to Future Modules

- **Module 2**: How do we actually learn parameters? Gradient descent
- **Module 3**: How do we prepare inputs? Feature engineering
- **Module 4**: What if outputs are classes? Logistic regression
- **Module 5**: What about multiple classes? Multiclass classification

---

Further Reading

- [Link to foundational papers will be added]
- Introduction to Statistical Learning, Chapter 2
- Pattern Recognition and Machine Learning, Chapter 1 (Bishop)