भारतीय प्रौद्योगिकी संस्थान मुंबई

Indian Institute of Technology Bombay

Artificial Intelligence and Machine Learning

Week 1: Optimization Foundations and Linear Regression

Swaprava Nath

Department of Computer Science and Engineering, IIT Bombay

ज्ञानम् परमम् ध्येयम्

Knowledge is the supreme goal

Course Overview

What this course covers (6 weeks):

1. Optimization + Linear Regression
2. Classification + SVM
3. Neural Networks I (MLP + Backprop)
4. Neural Networks II (CNN, RNN, Transformers)
5. Classical AI Search (A* and friends)
6. Multi-Agent AI (Game Theory + Minimax)

Evaluation:

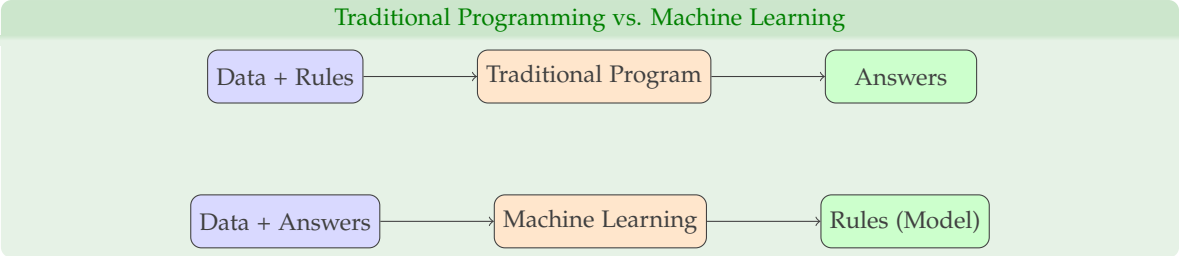
- **Programming Assignment** – end of Week 4 (in-person, proctored)
- **Written Endsem Exam** – end of Week 6 (in-person, proctored)

Prerequisites:

- Probability and statistics
- Linear algebra (matrices, eigenvalues)
- Calculus (partial derivatives)
- Basic Python programming

Contents

What is Machine Learning?



- In ML, we **learn a function** $f : \mathcal{X} \to \mathcal{Y}$ from **examples** $(x_1, y_1), \ldots, (x_n, y_n)$
- $\mathcal{X}$ is the **input space** (e.g., images, text, sensor readings)
- $\mathcal{Y}$ is the **output space** (e.g., a price, a label, a probability)

Types of Machine Learning

Supervised Learning

- Given: labeled pairs (x_i, y_i)
- Goal: learn $f(x) \approx y$
- Examples: house price prediction, spam detection

Unsupervised Learning

- Given: unlabeled data $\{x_i\}$
- Goal: discover structure
- Examples: clustering customers, topic modelling

Reinforcement Learning

- Given: environment + reward signal
- Goal: learn a policy via interaction
- Examples: game playing, robotics

This Course Focus

We cover **supervised learning** in depth (Weeks 1–4). Classical AI search (Week 5) and multi-agent game theory (Week 6) complete the picture.

Contents

Why Optimization?

- Almost every ML algorithm reduces to: **minimize a loss function** $L(\theta)$ over parameters θ
- Example: for linear regression, $L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^\top x_i)^2$
- We need to find $\theta^* = \arg\min_\theta L(\theta)$

Key Concept: Convexity

A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** if for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$:

$$f(\lambda x + (1-\lambda)y) \leqslant \lambda f(x) + (1-\lambda) f(y)$$

- Convex functions have **no local minima other than the global minimum**
- The squared loss $L(\theta) = \|\mathbf{y} - X\theta\|^2$ is convex in θ – so gradient descent will find the global minimum
- **Intuition:** the function bowl never has false valleys

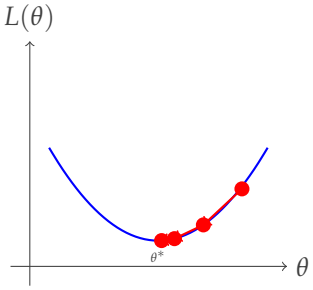
Gradient Descent

Algorithm: Gradient Descent

1. Initialize $\theta^{(0)}$ (e.g., randomly or at $\mathbf{0}$)
2. Repeat until convergence:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha \nabla_\theta L(\theta^{(t)})$$

- $\alpha > 0$ is the **learning rate** (step size)
- $\nabla_\theta L$ points in the direction of steepest **ascent**; we subtract it to descend
- Each step moves θ toward lower loss



Gradient Descent: Variants

Batch GD

- Use **all** n samples per update
- Stable, but slow for large n
- $\nabla L = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_i$

Stochastic GD (SGD)

- Use **one** random sample per update
- Fast but noisy updates
- $\nabla L \approx \nabla \ell_i$

Mini-batch GD

- Use a **batch** of B samples
- Best of both worlds
- Typical: $B \in \{32, 64, 128\}$

Learning Rate Sensitivity

- Too large α: oscillates or diverges
- Too small α: converges very slowly
- In practice: use learning rate schedules or adaptive methods (Adam, covered in Week 3)

Contents

The Supervised Learning Setup

Formal Setup

- **Training data**: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$
- **Hypothesis class**: a family of functions $\mathcal{H} = \{f_\theta : \theta \in \Theta\}$
- **Loss function**: $\ell(f_\theta(x), y)$ measures prediction error
- **Goal**: find $\theta^* = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i)$

- **Example:** Predict a student's exam score y from hours studied x_1 and assignments completed x_2
- Features: $x = (x_1, x_2) \in \mathbb{R}^2$, label: $y \in \mathbb{R}$
- We want to find a function that generalizes to **new, unseen** students

Linear Regression: Model

- **Linear regression** assumes $f_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d = \theta^\top \tilde{x}$
- where $\tilde{x} = (1, x_1, \ldots, x_d)^\top \in \mathbb{R}^{d+1}$ includes the bias term, $\theta \in \mathbb{R}^{d+1}$

Squared Error Loss (Ordinary Least Squares)

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^\top \tilde{x}_i)^2 = \frac{1}{n} \|y - X\theta\|^2$$

where $X \in \mathbb{R}^{n \times (d+1)}$ is the **design matrix** with rows $\tilde{x}_i^\top$, and $y \in \mathbb{R}^n$ is the label vector.

- This is a convex quadratic in θ
- **Why squared error?** Penalizes large mistakes more than small ones. Also arises naturally from a probabilistic model (next slide).

Closed-Form Solution: The Normal Equation

- Since $L(\theta) = \frac{1}{n}\|y - X\theta\|^2$ is convex and differentiable, set gradient to zero:

$$\nabla_\theta L = \frac{2}{n} X^\top (X\theta - y) = 0$$

- Solving: $X^\top X\theta = X^\top y$

Normal Equation

$$\hat{\theta} = (X^\top X)^{-1} X^\top y$$

provided $X^\top X$ is invertible (requires $n \geqslant d + 1$ and no redundant features).

- **Geometric interpretation:** $X\hat{\theta}$ is the **orthogonal projection** of y onto the column space of X
- $y - X\hat{\theta}$ (the **residual**) is perpendicular to every column of X
- **Limitation:** Computing $(X^\top X)^{-1}$ costs $O(d^3)$ – impractical when d is very large. Use gradient descent instead.

Probabilistic Interpretation: MLE

- Suppose the true relationship is $y_i = \theta^\top x_i + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ i.i.d.
- Then $y_i \mid x_i \sim \mathcal{N}(\theta^\top x_i, \sigma^2)$, so:

$$p(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^\top x_i)^2}{2\sigma^2} \right)$$

Maximum Likelihood Estimation (MLE)

Maximize $\prod_{i=1}^n p(y_i \mid x_i; \theta)$ over θ. Taking log and simplifying, this is **equivalent to minimizing** $\sum_{i=1}^n (y_i - \theta^\top x_i)^2$.

- **Conclusion:** Ordinary least squares is the MLE under Gaussian noise. Minimizing squared error is not just convenient – it has a principled probabilistic justification.

Gradient Descent for Linear Regression

For $L(\theta) = \frac{1}{n}\|y - X\theta\|^2$, the gradient is:

$$\nabla_\theta L = -\frac{2}{n} X^\top (y - X\theta)$$

Update Rule

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \frac{2\alpha}{n} X^\top \left(y - X\theta^{(t)}\right)$$

Convergence:

- For a fixed step size $\alpha < \frac{1}{\lambda_{\max}(X^\top X/n)}$, batch gradient descent converges to $\hat{\theta}$
- When n and d are large, mini-batch SGD is preferred over the normal equation

Gradient Descent for Linear Regression

For $L(\theta) = \frac{1}{n}\|y - X\theta\|^2$, the gradient is:

$$\nabla_\theta L = -\frac{2}{n} X^\top (y - X\theta)$$

Update Rule

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \frac{2\alpha}{n} X^\top \left(y - X\theta^{(t)}\right)$$

Convergence:

- For a fixed step size $\alpha < \frac{1}{\lambda_{\max}(X^\top X/n)}$, batch gradient descent converges to $\hat{\theta}$
- When n and d are large, mini-batch SGD is preferred over the normal equation

Question

When should you use the normal equation vs. gradient descent?

Gradient Descent for Linear Regression

For $L(\theta) = \frac{1}{n}\|y - X\theta\|^2$, the gradient is:

$$\nabla_\theta L = -\frac{2}{n} X^\top (y - X\theta)$$

Update Rule

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \frac{2\alpha}{n} X^\top \left(y - X\theta^{(t)}\right)$$

Convergence:

- For a fixed step size $\alpha < \frac{1}{\lambda_{\max}(X^\top X/n)}$, batch gradient descent converges to $\hat{\theta}$
- When n and d are large, mini-batch SGD is preferred over the normal equation

Question

When should you use the normal equation vs. gradient descent?

Answer

Normal equation: small d (e.g., $d < 10^4$). Gradient descent: large d or large n where matrix inversion is too slow.

Basis Functions: Going Beyond Linear

- Real data is often not linearly related to raw inputs
- **Basis functions** $\phi : \mathbb{R}^d \to \mathbb{R}^p$ transform the input before fitting a linear model:

$$f_\theta(x) = \theta^\top \phi(x)$$

- The model is **linear in** θ but **non-linear in** x

Polynomial basis (for $d = 1$):

$$\phi(x) = (1, x, x^2, \ldots, x^p)^\top$$

Fits degree-p polynomial curves.

Radial Basis Functions (RBF):

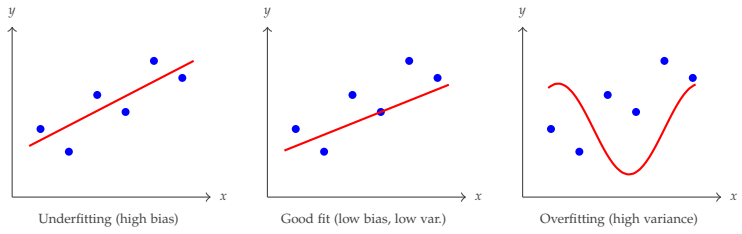
$$\phi_j(x) = \exp\left(-\frac{\|x - \mu_j\|^2}{2s^2}\right)$$

Basis function is a Gaussian with mean μ_j.

Warning: Overfitting

Using too many basis functions fits the training data exactly but generalizes poorly. This is the **bias-variance tradeoff** – a central theme in ML.

Overfitting and Model Selection



- **Bias:** error from wrong model assumptions (too simple)
- **Variance:** sensitivity to small fluctuations in training data (too complex)
- **Remedy:** use a **validation set** to choose model complexity; use **regularization**

Regularization: Ridge Regression

- Add a penalty on the magnitude of θ to the loss:

$$L_{\text{ridge}}(\theta) = \frac{1}{n}\|y - X\theta\|^2 + \lambda\|\theta\|^2$$

- $\lambda > 0$ is the **regularization strength** (a hyperparameter)
- Larger λ pulls θ toward zero, reducing variance at the cost of some bias

Closed-Form Ridge Solution

$$\hat{\theta}_{\text{ridge}} = (X^\top X + \lambda n I)^{-1} X^\top y$$

The added $\lambda n I$ ensures the matrix is always invertible (even if $n < d$).

- **Lasso** (ℓ_1 penalty): $\lambda\|\theta\|_1$ encourages **sparse solutions** – many $\theta_j = 0$
- **Elastic Net:** combines ℓ_1 and ℓ_2 penalties

Week 1 Summary

What We Learned

1. ML is about learning a function from data. Supervised learning uses labeled pairs.
2. Optimization underpins all of ML. Convex functions have global minima; gradient descent finds them.
3. Linear regression models $y = \theta^\top x$. The normal equation $\hat{\theta} = (X^\top X)^{-1} X^\top y$ gives the exact solution.
4. The squared error loss has a probabilistic justification: it is the MLE under Gaussian noise.
5. Basis functions let us fit non-linear patterns while keeping the optimization linear.
6. Overfitting arises when the model is too complex. Regularization (ridge, lasso) controls it.

Looking ahead (Week 2): When y is a discrete class label rather than a real number, we need **classification** methods – logistic regression and support vector machines.

भारतीय प्रौद्योगिकी संस्थान मुंबई

Indian Institute of Technology Bombay