Introduction to Machine Learning and the Prediction Problem

Deep Learning course - SKEMA 2025

Mastère Spécialisé® Chef de Projet Intelligence Artificielle

Salem Lahlou

Some figures adapted from 3Blue1Brown (YouTube)

Let's start with an example

Among the digits $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$, can you identify what the images above correspond to?

Easy, right?

Can you explain why they all correspond to the digit 3?

Traditional programming is not enough

- Identifying digits = Very easy for you and me
- Describing the process = Very hard
- $\rightarrow$ We can't really write a traditional computer program, with a bunch of if-else statements and for loops, to make a computer solve the problem
- What about the following problem?
  - Is the sentence "himlen er blå min ven" contain words not in the Swedish language?

The Swedish example


```
# Assume we have a file on our computer called swedish_dictionary.txt, that contains one word per line
list_of_swedish_words = []
with open('swedish_dictionary.txt', 'r') as f:
    for line in f.readlines():
        list_of_swedish_words.append(line.strip())
sentence = "himlen er blå min ven"
for word in sentence.split():
    if word not in list_of_swedish_words:
        return True  # As soon as we find a non swedish word, the program stops
return False  # If we have gone through all words and all of them are valid
```

What is Machine Learning?

Machine learning teaches computers to **discover patterns in *training* data** and use them to make predictions or decisions, **without explicit programming** for each task.

Prototype theory is a theory of categorization in cognitive science, particularly in psychology and cognitive linguistics, where any given concept in any given language has a real world example that best represents this concept.

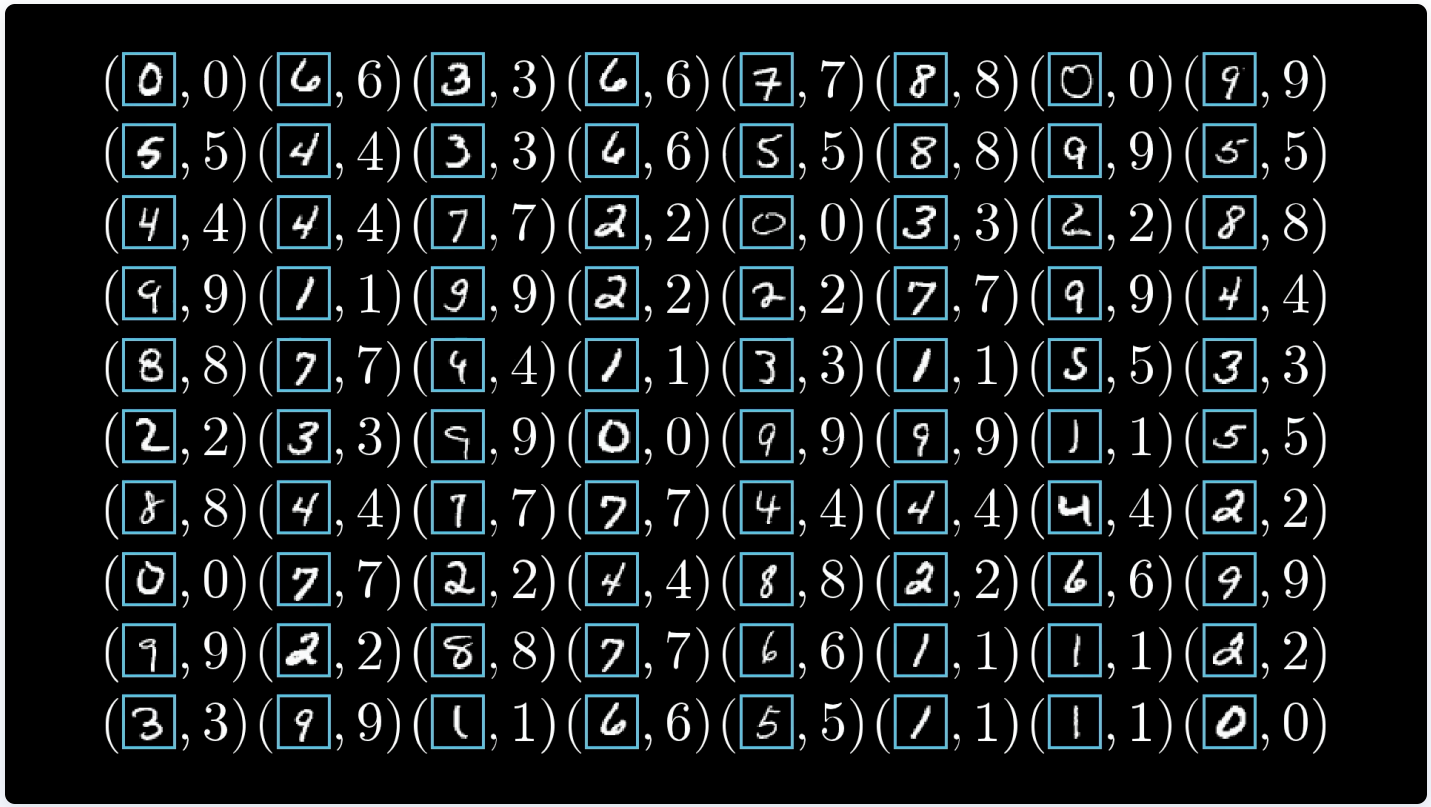
Traditional Programming:

- Input: Data + Rules
- Processing: Computer executes rules
- Output: Answers

Machine Learning:

- Input: Data + Answers (Examples)
- Processing: Computer learns rules
- Output: New Rules -> Predictions

Machine Learning = "Training" a system to learn from data



First quiz! Let's get to know each other

Go to the course homepage: https://la7.lu/sk25

Click on "Quiz 0"

Don't worry. It's anonymous!

This class

Content:

- Basics of Machine Learning
- Why linear regression is not enough
- Neural networks to the rescue
- Backpropagation: how to train a neural network
- Tips and Tricks
- Convolutional neural networks for image processing
- What's behind the hype today: language models and transformers
- A few jupyter notebooks we will go through together, to get exposed to code!

Schedule:

- Monday, May 5th: 9:00 to 12:00
- Tuesday, May 6th: 9:00 to 13:00
- Tuesday, May 6th: 14:00 to 17:00
- Wednesday, May 7th: 9:00 to 13:00
- **Your grade: ~30 minute multiple choice questions quiz, on **Wednesday, May 7th**, that will start some time between 9:00 and 12:29

++ In-class participation will count for your final grade
++ Some small homework

A few anonymous quizzes here and there (Passive learning doesn't work):

- Gives me a signal of how many are following, how fast/slow I am
- I can use the responses to adjust my pace, and content
- You have zero incentive to cheat (look online for answers, ask your favorite LLM assistant, look over your classmate's shoulder) in order to get answers, as it is anonymous
- For the benefit of your classmates, don't troll answer!

You can and *should* interrupt me for questions/clarifications whenever needed

The Prediction Problem: Why We Use Machine Learning

The Core Idea: Can we predict an unknown outcome based on known information?

It's Central Because: Most real-world value from ML comes from accurate predictions about the future or unseen situations.

Examples:

- *Business:* What **sales** will we achieve next quarter?
- *Finance:* What will this **stock price** be? Is this transaction **fraudulent**?
- *Everyday:* Is this email **spam**? What **movie** should I watch? Will it **rain** tomorrow?
- *LLMs*: What **token** is next?
- *Healthcare*: Does this scan show **cancer**? When will I **die**? What **molecule** would solve this disease?
- *Robotics*: What sequence of **actions** should I take to empty the dishwasher safely?
- *Imaging*: What **pixel values** would make this old corrupted photo better?

The Goal of Machine Learning

Primary Goal: Build models that **Generalize** well.

- **Generalization:** The model performs accurately on *new, unseen data*, not just the data it was trained on.
- It means the model learned the true underlying patterns, not just memorized the training examples (including noise).
- **The Challenge:** Finding the sweet spot between fitting the training data well and maintaining simplicity to generalize.

Why it matters: A model that only works on training data is practically useless.

Data is the Fuel for Machine Learning

Ubiquitous Data Sources:

- *Business:* Transactions, CRM data, supply chain logs
- *Sensors:* IoT devices, wearables, industrial sensors
- *Media:* Images, videos, audio recordings
- *Text:* Emails, documents, social media posts, web pages
- *Scientific:* Experiment results, simulation data

The Opportunity & Challenge: Transforming raw data into actionable insights and predictions.

Representing Data: Features and Vectors

What is a feature? An individual measurable property or characteristic of a phenomenon being observed.

What is a vector?

- An ordered list of numerical feature values for a *single data point*.
- The language computers understand for data.

Example: House Data Point (x_1)

- x_1 = [1500, 3, 2, 1985, 1] represents:
  - Feature 1: Square footage = 1500
  - Feature 2: Bedrooms = 3
  - Feature 3: Bathrooms = 2
  - Feature 4: Year Built = 1985
  - Feature 5: Has Garage = 1 (Yes)

From Vectors to Matrices: Structuring Datasets

What is a matrix?

- A 2D array (table) of numbers.
- Standard way to represent an entire dataset.
- **Rows:** Typically represent individual data points (samples, examples).
- **Columns:** Typically represent different features.

Example: Dataset of Houses (Shape: 3 samples x 5 features)

```
       Feat 1 Feat 2 Feat 3 Feat 4 Feat 5
       (SqFt) (Beds) (Baths)(Year) (Garage)
X = [[1500,    3,     2,    1985,    1],  # House 1 (Sample 1)
     [1200,    2,     1,    1995,    0],  # House 2 (Sample 2)
     [1800,    4,     3,    2005,    2]]  # House 3 (Sample 3)
```

Linear Algebra provides the tools to manipulate these matrices efficiently.

Mathematical Formulation of Learning

Let's formalize the goal with n training examples:

- X: Input features (matrix of vectors x_i, for $i \in \{1, \ldots, n\}$)
- y: Output target values (vector or matrix y_i, for $i \in \{1, \ldots, n\}$)
- f: The **unknown true underlying function** such that $y = f(X)$ (plus noise).
- $\hat{f}$: Our learned model, an **approximation** of f.

The Learning Problem:
Given examples (X, y), find a model $\hat{f}$ such that $\hat{f}(X_{new}) \approx y_{new}$ for new, unseen data X_{new}.
The challenge is estimating f effectively using only the training data.

Major Categories of Machine Learning

Supervised Learning: (Learn from Labeled Examples)

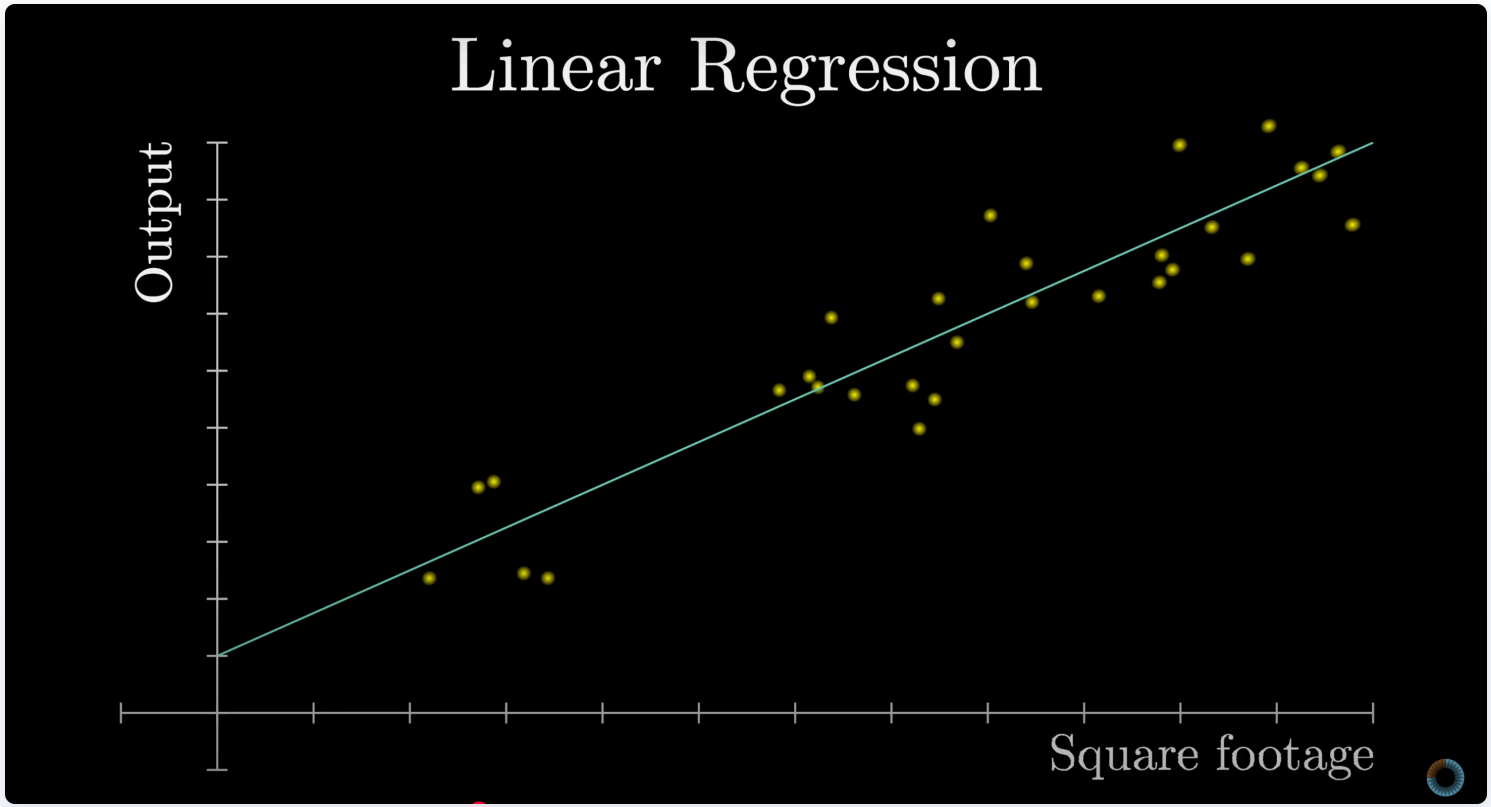
- **Input:** Data points (x_i, y_i) where y_i is the known answer/label.
- **Goal:** Learn a mapping $\hat{f}$ such that $\hat{f}(x) \approx y$.
- **Tasks:**
  - **Classification:** Predict discrete category (Spam/Not Spam, Cat/Dog).
  - **Regression:** Predict continuous value (House Price, Temperature).

Unsupervised Learning: (Find Structure in Unlabeled Data)

- **Input:** Data points x_i without known labels.
- **Goal:** Discover hidden patterns, groupings, or representations.
- **Tasks:**
  - **Clustering:** Group similar data points (Customer Segmentation).
  - **Dimensionality Reduction:** Simplify data while preserving structure (Visualization, Feature Engineering).
  - **Anomaly Detection:** Identify unusual data points (Fraud Detection).

(Self-Supervised & Reinforcement Learning exist too!)

Linear regression: a simple model



Goal: Find the line that best fits the existing data (pairs of (square foots/meters, house price)), that would best predict future house prices

Can you find the best fit?

Reminder: If the input is one-dimensional (only one number, rather than a vector of multiple numbers each representing a feature), a linear function is defined by:

$$f(x) = a \cdot x + b,$$

where a, b are the **parameters** of the function.

Can you find the parameter pair (a, b) that best fits the training data?
https://la7.lu/ml-viz-tool-skema2025/

(Accessible from the course website: https://la7.lu/sk25)

Supervised Learning: A Closer Look

Given a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$:

Regression: Predict a continuous value.

- Target $y_i \in \mathbb{R}$ (any real number).
- **Example:** Predicting house price (y_i) based on features (x_i).
- **Goal:** Learn $\hat{f}$ to minimize prediction error, e.g., $|y_i - \hat{f}(x_i)|$.

Classification: Predict a discrete category label.

- Target $y_i \in \{C_1, C_2, \ldots, C_K\}$ (one of K predefined classes).
- **Example:** Classifying email (x_i) as Spam (C_1) or Not Spam (C_2).
- **Goal:** Learn $\hat{f}$ to maximize the number of correct classifications $y_i = \hat{f}(x_i)$.

Why Probability is Essential in ML

Machine Learning operates under uncertainty:

- **Noisy Data:** Measurements are often imperfect.
- **Incomplete Information:** We rarely have all possible features.
- **Stochasticity:** The underlying processes might be inherently random.

Probability Theory allows us to:

- **Quantify Uncertainty:** Express confidence in predictions (e.g., 90% probability of rain).
- **Build Robust Models:** Account for noise and randomness.
- **Make Principled Decisions:** Choose actions based on likelihoods.
- **Understand Model Outputs:** Many models output probabilities (e.g., classification).

Limitations of linear models

Can you find a non-linear function that better fits the training data?
https://la7.lu/ml-viz-tool-skema2025/

(Accessible from the course website: https://la7.lu/sk25)

The Typical Machine Learning Workflow

1. **Problem Definition:** Clearly state the goal (e.g., predict Y from X).
2. **Data Collection & Preparation:** Gather, clean, and preprocess data. *Crucial & time-consuming!*
3. **Feature Engineering:** Select, transform, and create relevant features from raw data.
4. **Model Selection:** Choose a suitable algorithm class (e.g., Linear Regression, Decision Tree, Neural Network).
5. **Training:** Fit the model to the training data (optimize parameters using a loss function).
6. **Evaluation:** Assess model performance on unseen validation/test data using metrics.
7. **Hyperparameter Tuning:** Adjust model settings (not learned from data) based on validation performance.
8. **Deployment & Monitoring:** Use the model in production and track its performance over time.

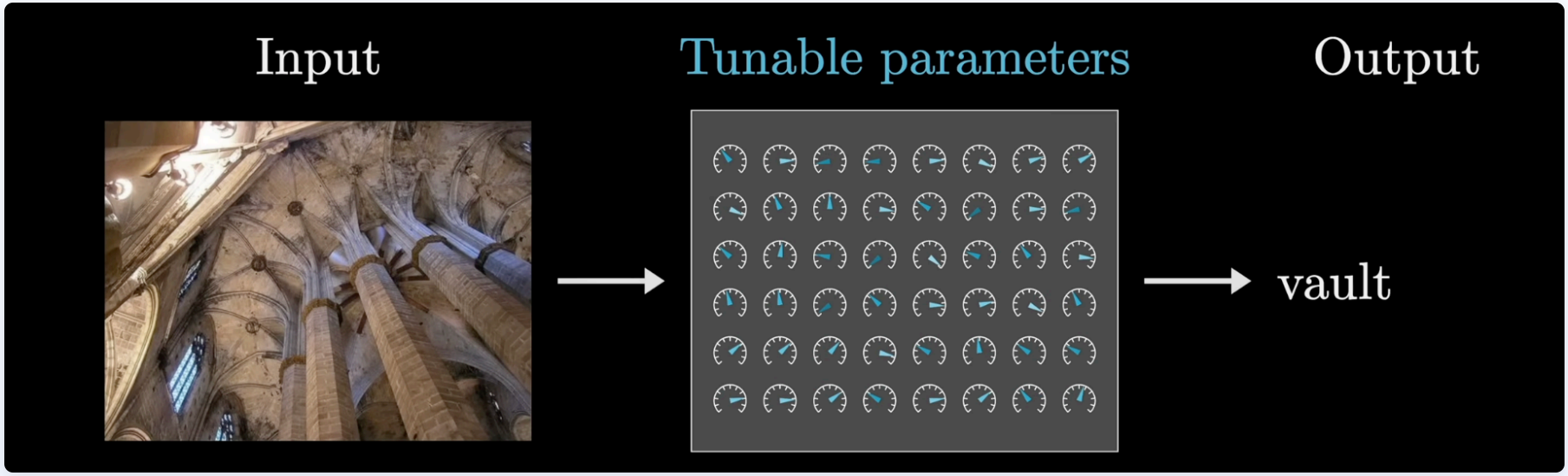
(Often iterative)

Parametric models

Most models are parametric. Different models have different numbers of parameters: a linear function in 1D has two parameters, a quadratic function in 1D has three parameters, a linear function in 2D has 3 parameters ($f(x_1, x_2) = a_1 x_1 + a_2 x_2 + b$), modern transformers have *hundreds of billions* of parameters.

We usually abstract this away by encoding **all parameters** in one **big vector** θ.

The goal of machine learning is to find the best parameters θ.



Understanding Loss Functions: Measuring Error

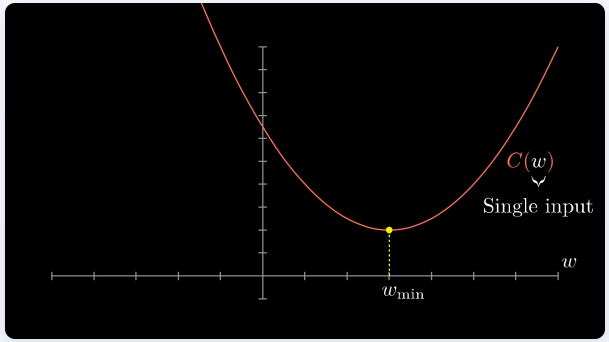
What is a Loss Function $J(\theta)$?

- A function that quantifies how "bad" a model's predictions ($\hat{y}$) are compared to the true values (y) for a given set of parameters θ.
- Measures the "cost" or "penalty" for prediction errors.
- **Goal of Training:** Find the parameters θ that **minimize** the loss function on the training data.

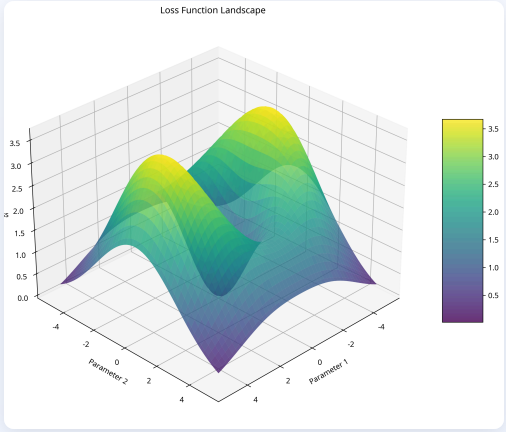
Why different loss functions?

- Different tasks (regression vs. classification) have different objectives.
- Choice depends on statistical assumptions and desired model behavior (e.g., sensitivity to outliers).

One-parameter function



Two-parameter function



Common Loss Functions & Their Intuition

Mean Squared Error (MSE) for Regression:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Intuition:** Penalizes larger errors much more heavily due to the squaring. Sensitive to outliers. Assumes errors are normally distributed.

Cross-Entropy Loss for Classification (Binary Example):

$$\text{Cross-Entropy} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)]$$

- Where y_i is 0 or 1, $\hat{p}_i$ is the predicted probability of class 1.
- **Intuition:** Measures the difference between two probability distributions (true labels vs. predicted probabilities). Penalizes confident wrong predictions heavily.

Why Calculus? The Engine of Optimization

The Core Problem: We have a loss function $J(\theta)$ that depends on model parameters θ. We want to find θ that minimizes $J(\theta)$.

How do we find the minimum?

- We need to know how the loss changes when we change the parameters.
- We need to know which direction to move θ to *decrease* the loss.

Calculus provides the tools:

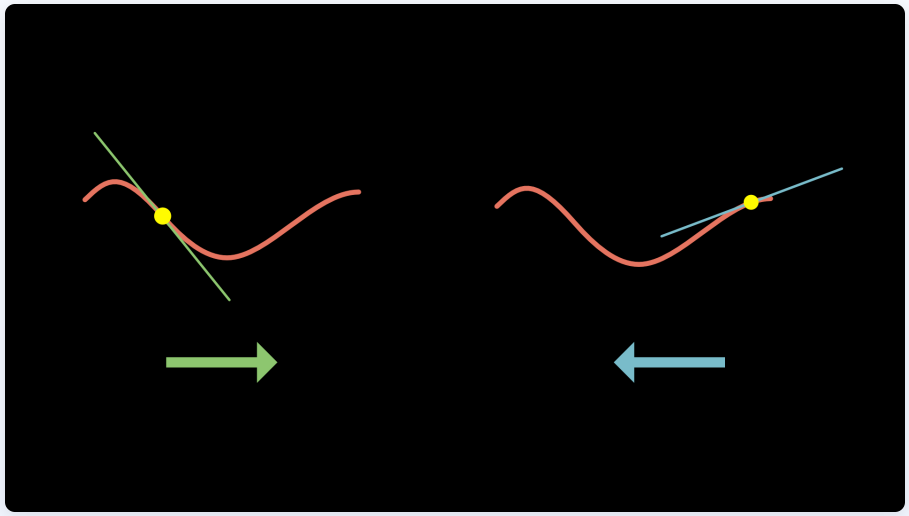
- **Derivatives ($\frac{dJ}{d\theta}$):** Tell us the rate of change (slope) of the loss wrt each parameter.
- **Gradients ($\nabla_\theta J(\theta)$):** A vector of partial derivatives; points in the direction of the *steepest ascent* of the loss function.
- **Optimization Algorithms (like Gradient Descent):** Use gradients to iteratively update parameters and find a minimum.

Loss Minimization: The Valley Analogy

Imagine finding the lowest point in a foggy valley:

- Your **position** (θ) = Current model parameter values.
- The **altitude** ($J(\theta)$) = The loss value for your current position.
- **Goal:** Reach the lowest point in the valley (minimum loss).
- **The Fog:** You can only see the slope immediately around you.

How? Check the slope (gradient) where you are and take a step downhill. Repeat. The gradient of the function $\theta \mapsto J(\theta)$ at some value θ_0 is a vector, denoted $\nabla_\theta J(\theta_0)$, that contains the **partial derivatives** of the function J.



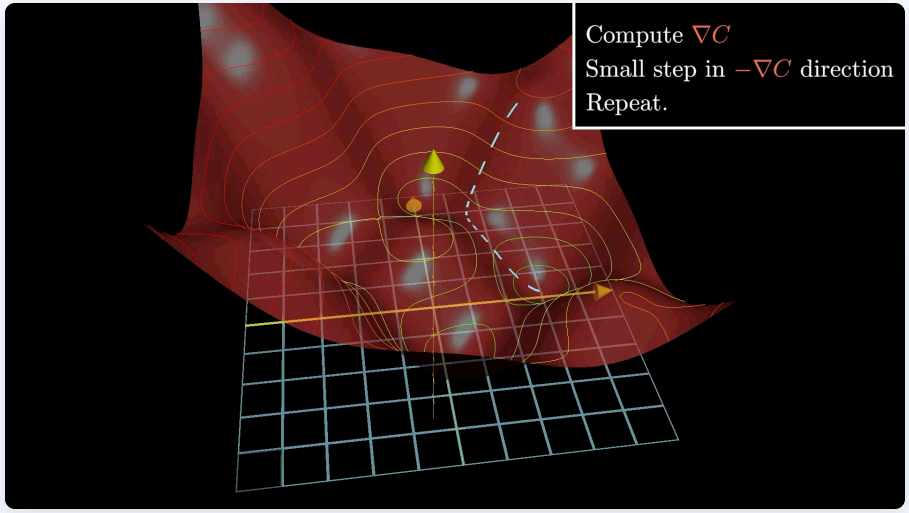
Optimization Algorithm: Gradient Descent

The Problem: How do we systematically find the minimum of the loss function $J(\theta)$?

The Strategy: Iteratively move in the opposite direction of the gradient.

1. Initialize parameters θ (randomly or smartly).
2. Calculate the gradient $\nabla_\theta J(\theta)$ at the current θ. (Which way is uphill?)
3. Update parameters by taking a small step *downhill*:
   $\theta_{new} = \theta_{old} - \alpha \nabla_\theta J(\theta)$
4. Repeat steps 2-3 until the loss converges (stops decreasing significantly).

- α is the **learning rate**: controls the step size.*



Gradient Descent: The Math

The core update rule for a parameter θ_j:

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

In vector notation for all parameters θ:

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$$

Where:

- θ: Vector of model parameters.
- $J(\theta)$: The loss function we want to minimize.
- $\nabla_\theta J(\theta)$: The gradient of the loss function with respect to the parameters θ. It's a vector where each element is $\frac{\partial J}{\partial \theta_j}$.
- α: The learning rate (a small positive scalar). Controls how big a step we take.

Gradient explained

$$-\nabla C(\vec{\mathbf{W}}) = \begin{bmatrix} 0.31 \\ 0.03 \\ -1.25 \\ \vdots \\ 0.78 \\ -0.37 \\ 0.16 \end{bmatrix}$$

w_0 should increase somewhat
w_1 should increase a little
w_2 should decrease a lot

$w_{13,000}$ should increase a lot
$w_{13,001}$ should decrease somewhat
$w_{13,002}$ should increase a little

Optimization: Ideal Metric vs. Surrogate Loss

Ideal Scenario:

- We ultimately care about a specific evaluation metric (e.g., Accuracy, F1-Score, Precision, Recall, ROI).
- We would ideally optimize the model parameters θ to directly maximize/minimize this target metric.

The Reality:

- Most standard evaluation metrics are **non-differentiable** or computationally hard to optimize directly (e.g., accuracy involves counting, which has zero gradient almost everywhere).
- Gradient Descent requires smooth, differentiable functions.

The Solution:

- Use a **surrogate loss function** (like MSE or Cross-Entropy) that *is* differentiable and closely related to the target metric.
- **Assumption:** Minimizing the surrogate loss will likely lead to good performance on the actual target metric.

Data Splitting: Training, Validation, and Test Sets

Why split data? To simulate generalization and avoid overly optimistic performance estimates.

Training Set (e.g., 60-80%):

- **Purpose:** Used to **train** the model.
- The model learns patterns and optimizes its parameters based *only* on this data.

Validation Set (e.g., 10-20%):

- **Purpose:** Used for **hyperparameter tuning** and **model selection**.
- *Not* used for training parameters directly.
- Helps choose the best model architecture, learning rate, regularization strength, etc.
- Provides an unbiased estimate of performance *during* development.

Test Set (e.g., 10-20%):

- **Purpose:** Provides a **final, unbiased evaluation** of the *chosen* model's performance after training and tuning are complete.
- **Crucial:** Must be held out and used *only once* at the very end.
- Represents how the model is expected to perform on completely new, unseen data in the real world.

Generalization: The Holy Grail of Machine Learning

What is Generalization?

- The ability of a model to perform well on **new, unseen data** after learning patterns from the training data.
- It's about *learning* the underlying structure, not just *memorizing* the examples it was trained on.

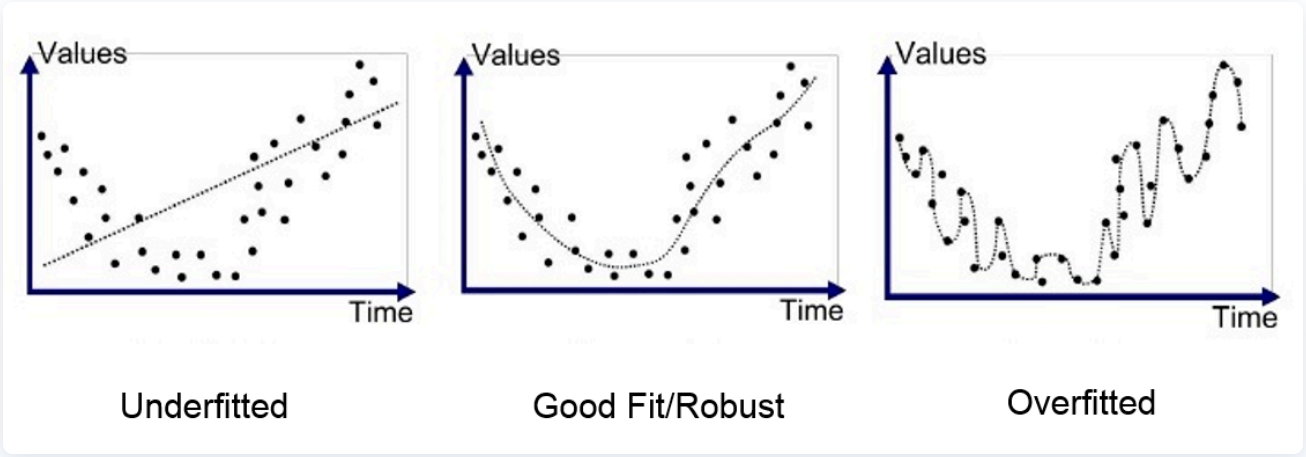
Why is it the "Holy Grail"?

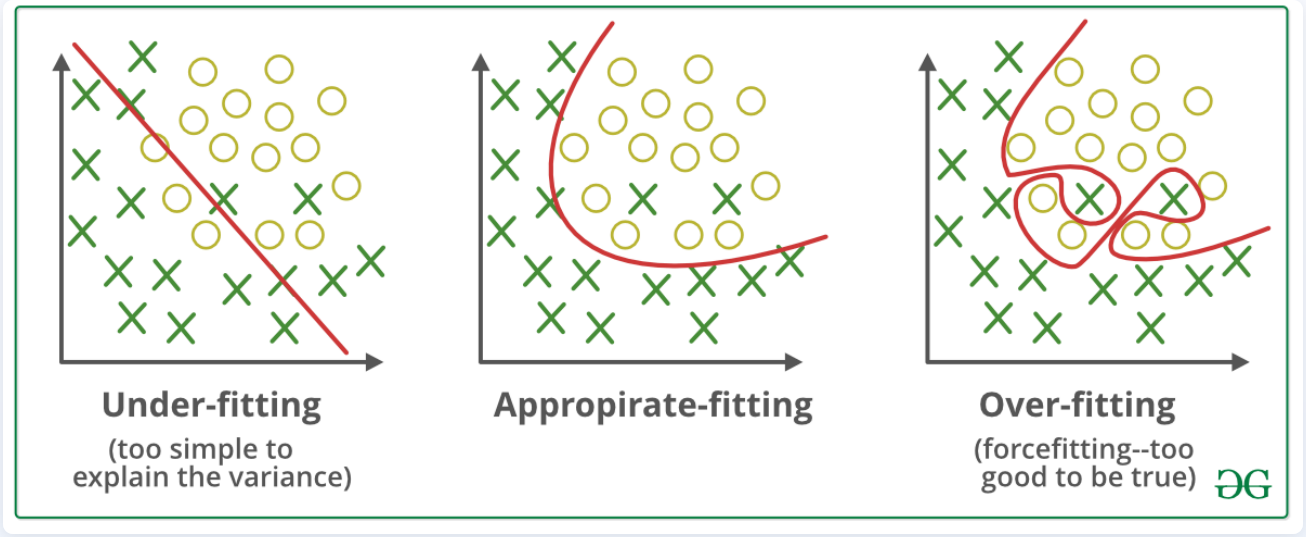
- The true value of a model lies in its ability to make accurate predictions in real-world scenarios (i.e., on data it hasn't seen before).
- A model that only performs well on training data is useless.
- Generalization means the model has captured the true signal, not just the noise.

Finding the Sweet Spot:

- **Underfitting:** Model too simple. High error on Training Set, High error on Test Set. (High Bias)
- **Good Fit:** Model captures underlying pattern. Low error on Training Set, Low error on Test Set. (Goal!)
- **Overfitting:** Model too complex/memorized noise. Very Low error on Training Set, High error on Test Set. (High Variance)

Underfitting, Overfitting visualized





Sources: geeksforgeeks, Anup Bhande

Session 1: Key Takeaways

1. **ML Goal:** Learn patterns from data to make predictions on *new* data (**Generalization**).
2. **Data Representation:** Features are organized into **vectors** (samples) and **matrices** (datasets). Linear algebra is key.
3. **Learning Process:** Models learn by minimizing a **Loss Function** that measures prediction error.
4. **Optimization: Calculus** (gradients) drives algorithms like **Gradient Descent** to find parameters that minimize loss.
5. **Evaluation:** Splitting data into **Train/Validation/Test** sets is crucial for assessing true performance and avoiding overfitting.
6. **Core Challenge:** Balancing model complexity to achieve good generalization (avoiding under/overfitting).

Quiz time

Go to the course homepage: https://la7.lu/sk25

Click on "Quiz 1"

Don't worry. It's anonymous!