Introduction to Machine Learning

Fundamentals of Machine Learning

Xufeng Zhang

Inria
`xufeng.zhang@inria.fr`

December 11, 2025

Course context

- This lecture is part of the course **Fundamentals of Machine Learning**.
- Goal of this first lecture:
  - Build intuition for what machine learning (ML) is.
  - Motivate why ML is useful across many domains.
  - Introduce the basic ML paradigm and terminology.
  - Discuss feature representation and distance metrics.
- We deliberately avoid complex algorithms and heavy mathematics.

Plan of the lectures

1. Introduction to Machine Learning + Practical Work (4h)
2. Supervised Learning: Classification, Regression + PW (4h)
3. Supervised Learning: More Methods, Model Evaluation + PW (4h)
4. Unsupervised Learning: Clustering + PW (4h)
5. Machine Learning Project with LLM (3h)
6. Competition session (3h)

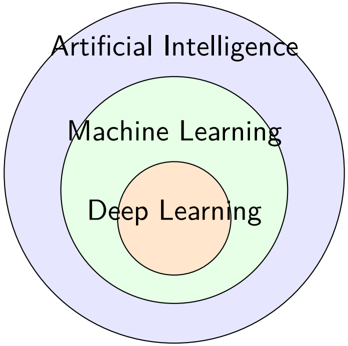
What you will learn in this course

- How to translate real-world problems into ML tasks.
- The main families of ML methods and when to use them.
- How to represent data as **feature vectors**.
- How to train, validate and evaluate ML models.
- How to reason about generalization, overfitting and underfitting.
- Hands-on ML practice in Python / common ML libraries.

Where is Machine Learning?

- **Digital services**
  - Recommendation systems (movies, music, products).
  - Search ranking and query suggestion.
  - Online advertisement targeting.
- **Perception**
  - Image and face recognition.
  - Speech recognition and voice assistants.
  - Photo filters and style transfer.
- **Decision and control**
  - Self-driving cars and robotics.
  - Fraud detection and credit scoring.
  - Medical diagnosis support systems.

AI, Machine Learning and Deep Learning



- **AI**: broad field of making machines perform tasks requiring human intelligence.
- **ML**: subset of AI that learns patterns from data instead of using hand-crafted rules.
- **DL**: subset of ML based on deep neural networks, powerful for unstructured data.

What is Machine Learning? — definitions

- Informal idea: build programs that **improve their performance with experience**.
- Common textbook-style definition: ML studies computer algorithms that **learn from data** and **generalize** to unseen data, instead of following hard-coded instructions.
- In practice:
    - We observe example pairs of input and output.
    - We construct a model that captures regularities in the data.
    - We use the model to make predictions or decisions for new inputs.

Machine Learning vs. Traditional Programming

Traditional programming

- Developer writes explicit rules.
- Program + input data $\Rightarrow$ output.
- Works well when rules are clear and stable.

Machine learning

- Developer provides data and a learning algorithm.
- Algorithm learns rules (model) from data.
- Useful when rules are complex or unknown.

One program (the learning algorithm) can be reused to solve many problems by changing only the data.

A motivating example: game retention (1/3)

- Imagine you launched a mobile game.
- Some players:
    - Play for a few hours and never return.
    - Others come back every day and even pay for in-game items.
- Business question:

How can we learn the behaviour of players and design strategies to keep them playing?

A motivating example: game retention (2/3)

- Step 1: **Collect data**
  - For each player, record features:
    - Total play time.
    - Levels completed.
    - Number of friends invited.
    - In-game purchases, etc.
  - Record outcome:
    - 1 if the player is still active after 7 days.
    - 0 otherwise.
- Each player becomes one data point in our dataset.

A motivating example: game retention (3/3)

- Step 2: **Train an ML model**
  - Input: player features.
  - Output: probability that the player will remain active.
- Step 3: **Use the model**
  - Identify at-risk players (low predicted probability).
  - Offer them a special item or bonus (e.g., powerful sword).
  - Measure whether the intervention improves retention.
- This is a typical **supervised classification** task.

How are things learned? Memorization

- **Memorization**
  - Accumulation of individual facts:
    - “Player A stayed 10 days.”
    - “Player B left after 2 days.”
  - Limited by:
    - Time to observe all possible situations.
    - Memory to store all examples.
- Pure memorization does **not** help us make good predictions for new players.

How are things learned? Generalization

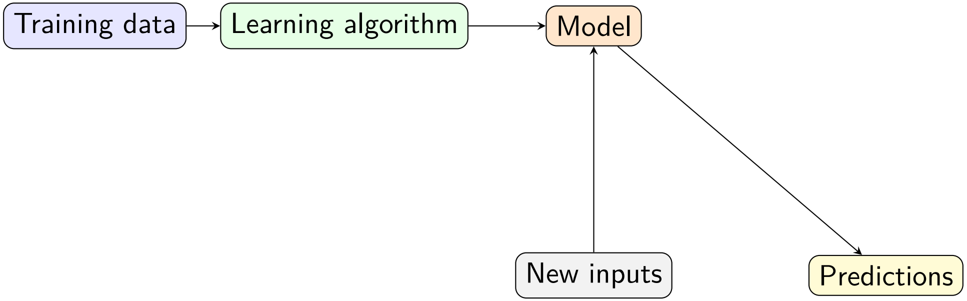
- **Generalization**
  - Deduce new facts from old facts.
  - Look for regularities, patterns and structure in the data.
  - Essentially a **predictive activity**.
- Assumption: **the past is informative about the future**.
- Goal of ML: build programs that generalize well from training data to unseen data.

Declarative vs. Imperative knowledge

- **Declarative knowledge**
    - Facts: “this email is spam”, “this image contains a cat”.
    - Easy to store, hard to use directly for new cases.
- **Imperative knowledge**
    - Procedures: how to decide if an email is spam.
    - ML aims to infer such procedures from many labelled examples.
- We are interested in programs that infer **useful procedures** from data.

Basic paradigm of Machine Learning

1. Observe a set of examples: **training data**.
2. Infer an internal model of the process that generated the data.
3. Use this model to make predictions on **test data** (unseen examples).



Two main variations on the paradigm

Supervised learning

- Training data: feature/label pairs (x_i, y_i).
- Goal: learn a function that predicts y for a new x.
- Tasks: classification, regression, sequence labelling.

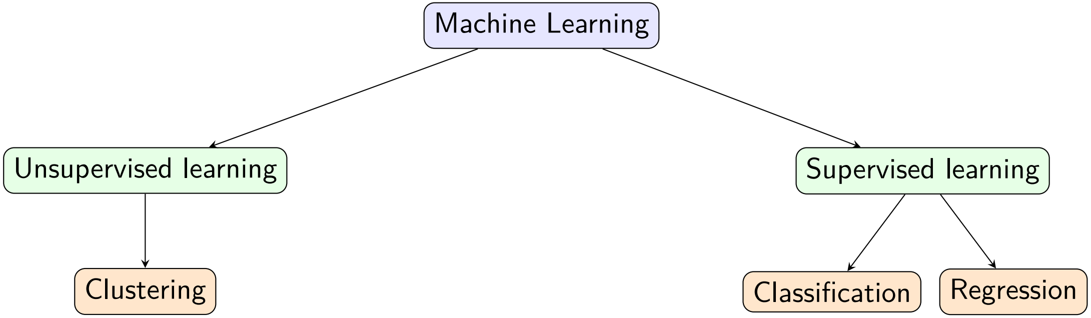
Unsupervised learning

- Training data: feature vectors x_i without labels.
- Goal: discover structure, e.g., clusters, latent factors.
- Tasks: clustering, dimensionality reduction, density estimation.

Other learning paradigms (high-level view)

- **Semi-supervised learning**
  - Use both labelled and unlabelled data to improve performance.
- **Reinforcement learning**
  - Agent interacts with an environment.
  - Receives rewards and learns a policy to maximize long-term reward.
- **Online learning**
  - Data arrives sequentially; the model is updated continuously.
- In this lecture we focus on basic supervised and unsupervised intuition.

Machine learning methods



Typical ML workflow

1. Formulate the problem and define the goal.
2. Collect and clean data.
3. Represent objects with feature vectors.
4. Choose a model family and objective function.
5. Train the model using an optimization algorithm.
6. Validate and tune hyperparameters.
7. Evaluate on test data.
8. Deploy and monitor the model in production.

What all ML methods require

- A suitable **training dataset** and evaluation method.
- A **representation** of the features.
- A **distance** or similarity measure between feature vectors (for many methods).
- An **objective function** (loss) and possibly constraints.
- An **optimization method** to learn model parameters.

What is a feature?

- A **feature** is a measurable property of an object.
- Examples for an email:
  - Number of exclamation marks.
  - Presence of certain keywords.
  - Sender domain, time of day.
- We group features into a **feature vector**:

$$x = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d.$$

- Good features make patterns easier to learn and generalize.

Feature engineering

- The art of constructing informative features from raw data.
- Goals:
    - Facilitate generalization.
    - Avoid overfitting by reducing noise and irrelevant dimensions.
    - Encode prior knowledge about the problem.
- Strategies:
    - Transform variables (log, normalization, binning).
    - Create new features (ratios, interactions, counts).
    - Select a subset of useful features.

Reptile classification example — setting

- Suppose we want to classify animals into:
    - **Reptile** vs. **Not reptile**.
- We start with candidate features:
    - Egg-laying (yes/no).
    - Has scales (yes/no).
    - Poisonous (yes/no).
    - Cold-blooded (yes/no).
    - Number of legs (integer).
- Our labels come from a biology expert.

Reptile classification — initial model

- First attempt:

Rule 1
An animal is a reptile if it:

- lays eggs,
- has scales,
- is poisonous,
- is cold-blooded,
- has no legs.

- Obviously this is too strict: many reptiles will not satisfy all conditions.
- We need to refine our feature set and rules.

Reptile classification — refining the rule (1/3)

- Consider cobra, rattlesnake and boa constrictor.
- We observe:
  - All have scales, are cold-blooded and have no legs.
  - Egg-laying and poisonous may vary.
- Updated model:

Rule 2
Reptile if:

- has scales,
- is cold-blooded,
- has no legs.

- This rule correctly classifies these snakes as reptiles.

Reptile classification — refining the rule (2/3)

- Now we add chicken, which:
  - lays eggs,
  - is warm-blooded,
  - has two legs,
  - has no scales.
- Rule 2 correctly identifies chicken as **not** a reptile.
- Our model is improving but remains incomplete.

Reptile classification — refining the rule (3/3)

- Add alligator and dart frog.
- Revised idea:

Rule 3
Reptile if:

- has scales,
- is cold-blooded,
- has 0 or 4 legs.

- Problem:
  - Some non-reptiles (e.g., certain fish) may also satisfy these features.
  - Difficult to find a simple exact rule with our limited features.

Imperfect but useful models

- With current features, we may obtain a rule that:
  - rarely misses true reptiles (**few false negatives**),
  - but may mistakenly label some animals as reptiles (**false positives**).
- In many applications this trade-off is acceptable.
- Feature engineering often aims at:
  - reducing false positives,
  - without increasing false negatives too much.

Feature engineering lessons from reptiles

- Features must be **informative** for the task.
- Too few features $\Rightarrow$ cannot separate classes.
- Too many irrelevant features $\Rightarrow$ noise and overfitting.
- Sometimes we must change representation:
    - continuous vs. binary,
    - absolute counts vs. ratios,
    - domain-specific transformations.

From animals to feature vectors

- Encode animals as binary / integer feature vectors.
- Example:

$$\begin{aligned}\text{rattlesnake} &= (1,1,1,1,0)\\ \text{boa constrictor} &= (0,1,0,1,0)\\ \text{dart frog} &= (1,0,1,0,4)\end{aligned}$$

- We now need a way to measure how similar two animals are.

The Minkowski distance

Definition
For two feature vectors $x_1, x_2 \in \mathbb{R}^d$ and $p \geq 1$,

$$\operatorname{dist}_p(x_1, x_2) = \left(\sum_{k=1}^{d} |x_{1k} - x_{2k}|^p \right)^{1/p} .$$

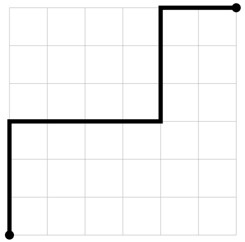
- $p = 1$: **Manhattan distance**.
- $p = 2$: **Euclidean distance**.
- Larger p emphasize larger coordinate differences.

Manhattan distance: taxicab geometry

- Imagine a city with a grid of streets.
- Distance is the number of blocks you must walk:

$$d_1(x, y) = \sum_k |x_k - y_k|.$$

- Many shortest paths may exist.
- Often appropriate when different dimensions are not directly comparable.



Euclidean distance

- Standard geometric distance in $\mathbb{R}^d$:

$$d_2(x, y) = \sqrt{\sum_k (x_k - y_k)^2}.$$

- Corresponds to the length of the straight line segment between two points.
- Often the default in many ML algorithms (e.g., k-means, k-NN).
- Sensitive to scale of each dimension.

Which distance says what?

- Consider a simple 2D grid with three points: a circle, a star and a cross.
- Question: is the circle closer to the star or to the cross?
- Using Euclidean distance:
  - cross ≈ 2.8
  - star ≈ 3.0
- Using Manhattan distance:
  - cross $= 4$
  - star $= 3$
- Different metrics can lead to different nearest neighbours.

Scaling issues: adding the alligator

- Recall:

$$\begin{aligned} \text{dart frog} &= (1,0,1,0,4),\\ \text{boa} &= (0,1,0,1,0),\\ \text{alligator} &= (1,1,0,1,4). \end{aligned}$$

- Legs feature ranges from 0 to 4, others from 0 to 1.
- The “legs” dimension dominates Euclidean distance.
- Alligator may incorrectly appear closer to dart frog than to boa.

Feature scaling and normalization

- To avoid dominated dimensions we often:
  - rescale each feature to comparable ranges,
  - or standardize to zero mean and unit variance.
- After scaling, distances better reflect meaningful similarity.
- In the reptile example, we might:
  - convert “legs” to a binary feature (has legs / no legs),
  - or divide by maximum number of legs.

Using binary features

- Encode presence/absence rather than counts:

$$\begin{aligned} \text{rattlesnake} &= (1,1,1,1,0) \\ \text{boa} &= (0,1,0,1,0) \\ \text{dart frog} &= (1,0,1,0,1) \\ \text{alligator} &= (1,1,0,1,1) \end{aligned}$$

- Now alligator is closer to the snakes than to the frog:
  - Distances reflect the intended biological similarity.
- **Feature engineering matters!**

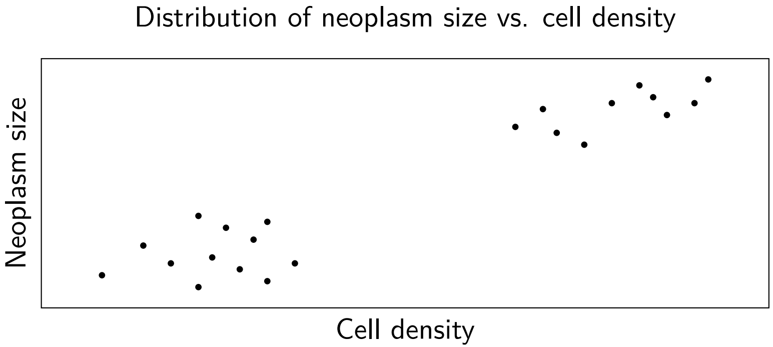
Summary: similarity measures

- Many ML methods rely on a notion of distance or similarity.
- Choice of metric interacts with feature representation.
- Key points:
  - Use appropriate scaling/normalization.
  - Consider Manhattan vs. Euclidean for different data types.
  - Binary / categorical features often require specialized measures.
- Poor choices here can severely hurt model performance.

Breast cancer example — data

- Each example: breast tumor (neoplasm).
- Features:
  - Neoplasm size.
  - Cell density.
- Two underlying types:
  - **Benign**.
  - **Malignant**.
- Initially assume we do **not** know which point belongs to which type.

Unlabelled data: scatter plot



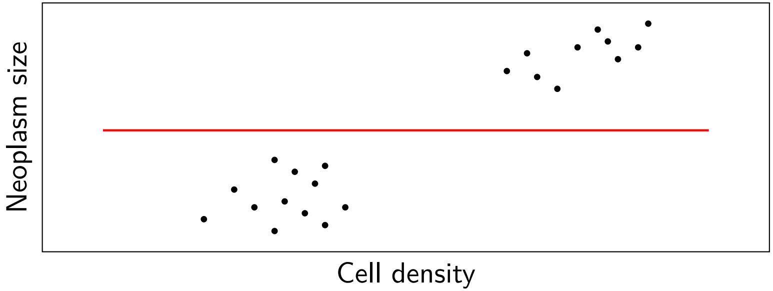
Two visually separable clusters suggest two neoplasm types.

Task: clustering examples into groups

- **Clustering**:
  - Group examples such that points in the same cluster are similar.
  - Points in different clusters are dissimilar.
- We need:
  - A distance measure (e.g., Euclidean).
  - A number of clusters K (here, assume $K = 2$).
  - An objective, e.g., minimize distance within clusters.
- Algorithms: k-means, hierarchical clustering, etc. (not covered in detail here).

Similarity based on neoplasm size only

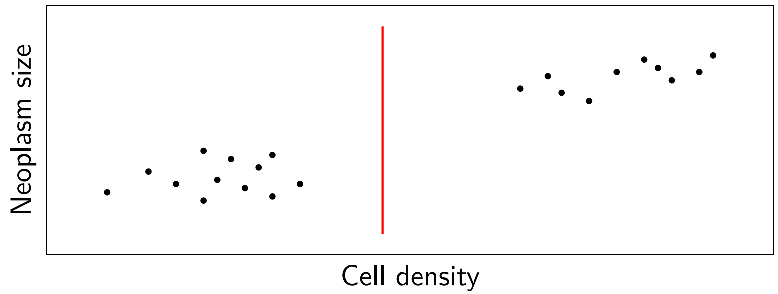
Vertical split using size only



Horizontal line splits data purely by size.

Similarity based on cell density only

Vertical split using density only



Vertical line splits data purely by cell density.

Using both attributes

- A better similarity notion uses both size and cell density.
- Clusters correspond to two dense regions in 2D space.
- Many clustering algorithms implicitly do this by:
  - representing each point as $(\text{density}, \text{size})$,
  - minimizing within-cluster distances in this 2D space.
- This illustrates again the importance of feature representation.

If the data were labelled

- Now suppose each point is labelled:
  - blue: benign,
  - red: malignant.
- Task becomes **supervised classification**.
- Goal: find a **decision boundary** that separates the two classes.
- In 2D this is a line or curve; in higher dimensions, a surface.

Finding classifier surfaces

- Given labelled training data, we want a function $f(x)$ such that:
  - $f(x) = 0$ for benign,
  - $f(x) = 1$ for malignant (for example).
- The **classifier surface** is where the model switches prediction.

  $$\{x : f(x) = 0.5\}$$

  (for probabilistic models).
- Constraints:
  - Surface should not be too complex (avoid overfitting).
  - Some trade-off between false positives and false negatives.

Adding new data points

- After training a classifier on past tumors, we receive a new patient.
- We measure:
  - neoplasm size,
  - cell density.
- We feed the feature vector into the model:
  - obtain a predicted probability of malignancy,
  - possibly use a threshold (e.g., 0.5) to decide the label.
- This is the standard way ML models support decision making.

Clustering vs. classification

Clustering (unsupervised)

- Input: unlabelled data.
- Goal: discover groups based on similarity.
- Output: cluster assignments.
- No direct notion of “correct” label.

Classification (supervised)

- Input: labelled data.
- Goal: learn mapping from features to labels.
- Output: predicted labels or probabilities.
- Performance measured against ground truth labels.

Sources of error

- **Overlapping classes**
  - Benign and malignant regions not perfectly separable.
- **Noisy measurements**
  - Size and density may be measured with error.
- **Limited features**
  - Important biological factors not captured.
- **Limited data**
  - Model may not see enough examples of rare cases.

Statistical view of machine learning

- Assume input X and output Y follow an unknown joint distribution $P(X, Y)$.
- Training and test data are drawn i.i.d. from this distribution.
- Goal of learning:
    - Approximate either the conditional probability $P(Y \mid X)$,
    - or a decision function $f(X)$ that predicts Y.
- This motivates the term **statistical learning**.

Hypothesis space and model

- We restrict ourselves to a set of candidate functions:

$$\mathcal{H} = \{f_\theta : \theta \in \Theta\},$$

  called the **hypothesis space**.
- Examples:
  - Linear models.
  - Decision trees.
  - Neural networks of a given architecture.
- Learning $\approx$ choosing a good θ based on data.

Loss, risk and empirical risk

- **Loss function** $\ell(f(x), y)$ measures how bad a prediction is.
  - Classification: 0–1 loss, cross-entropy.
  - Regression: squared loss, absolute loss.
- **Risk** (expected loss):

$$R(f) = \mathbb{E}_{(X,Y)\sim P}[\ell(f(X), Y)].$$

- We cannot compute $R(f)$ directly (because P is unknown).
- Instead we minimize the **empirical risk** on training data.

Overfitting and model complexity

- Minimizing empirical risk alone may cause **overfitting**.
- Overfitting:
  - Model fits training data extremely well,
  - but performs poorly on unseen test data.
- Typically happens when:
  - Model is too complex relative to the amount of data.
  - Features contain too much noise.
- We need **regularization** and good model selection.

Regularization idea (informal)

- Add a penalty for model complexity:

$$\text{objective} = \text{empirical risk} + \lambda \cdot \text{complexity}(f).$$

- Examples:
  - Penalize large weights in linear models.
  - Penalize deep or unpruned decision trees.
- Balances fit to training data and ability to generalize.

Model evaluation and selection

- Split data into:
  - Training set.
  - Validation set (for model/hyperparameter selection).
  - Test set (for final evaluation).
- Use techniques such as:
  - k-fold cross-validation.
  - Stratified sampling for imbalanced classes.
- Always report performance on data not used for training.

Key takeaways (1/2)

- Machine learning builds models that **learn from data** and generalize to new cases.
- ML is a subset of AI, and deep learning is a subset of ML.
- Basic paradigm:
  - collect data,
  - choose representation and model,
  - train and evaluate.
- Feature engineering and distance metrics strongly influence results.

Key takeaways (2/2)

- Supervised vs. unsupervised learning:
  - **Classification**: predict discrete labels.
  - **Regression**: predict continuous values.
  - **Clustering**: group similar examples without labels.
- Statistical learning view:
  - loss, risk, empirical risk,
  - overfitting and regularization.
- Always keep in mind:

The quality of data and features often matters more than the choice of algorithm.

Next lectures

- **Supervised Learning I**
  - k-nearest neighbours.
  - Linear regression, logistic regression.
- **Supervised Learning II**
  - Decision trees and ensembles.
  - Model evaluation metrics.
- **Unsupervised Learning**
  - k-means and other clustering methods.
  - Dimensionality reduction.

Suggested reading

- T. Mitchell, *Machine Learning*.
- K. Murphy, *Probabilistic Machine Learning*.
- A. Ng, *Machine Learning* (online course).
- Articles:
    - “Machine learning, explained” (MIT Sloan).
    - “What is Machine Learning?” (IBM).

Questions?

Thank you for your attention.

Questions and discussion.