Introduction to ML

Foundations, Models, and Applications

Summer school on Machine Learning for Space 2026

Federica Bragone

bragone@kth.se

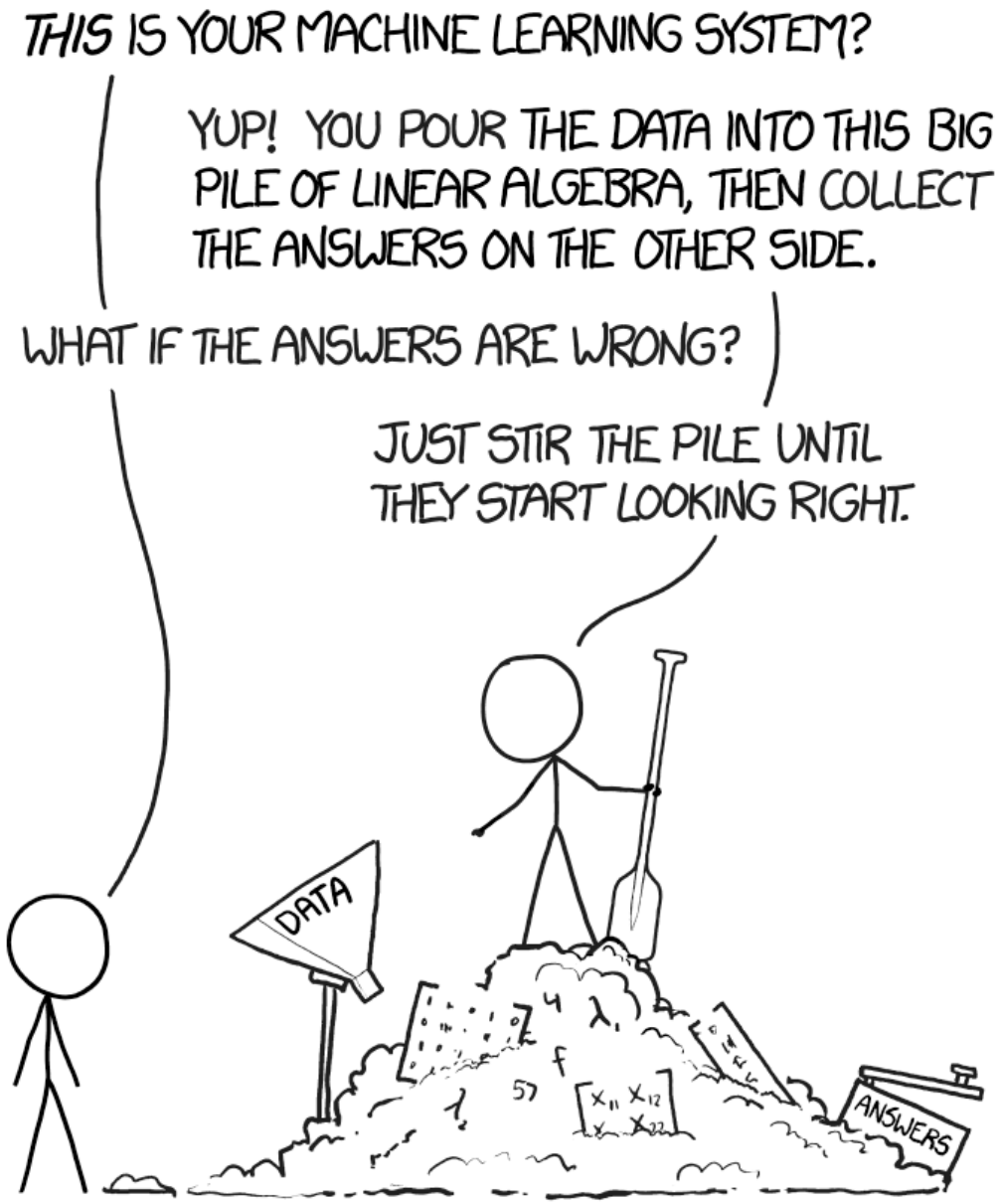
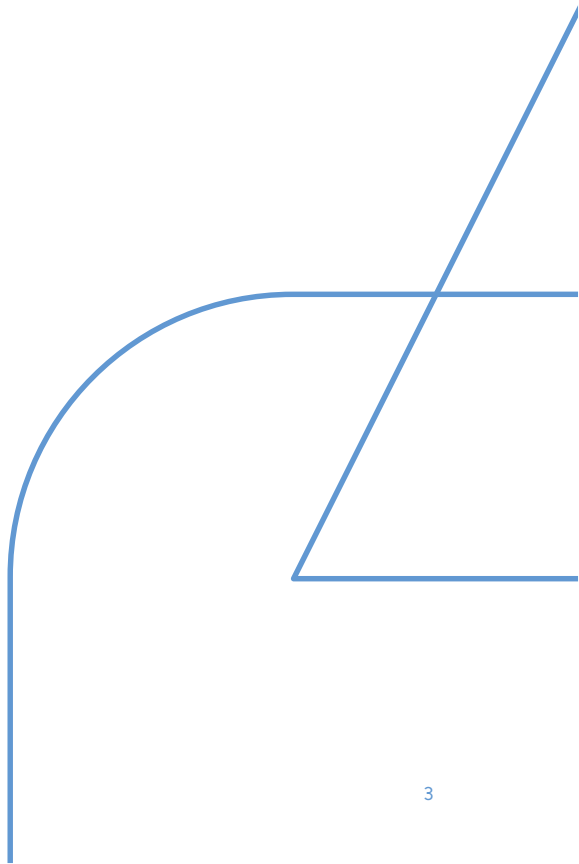


Image source: Munroe, R. (2017). *Machine Learning* [webcomic]. xkcd. https://xkcd.com/1838/

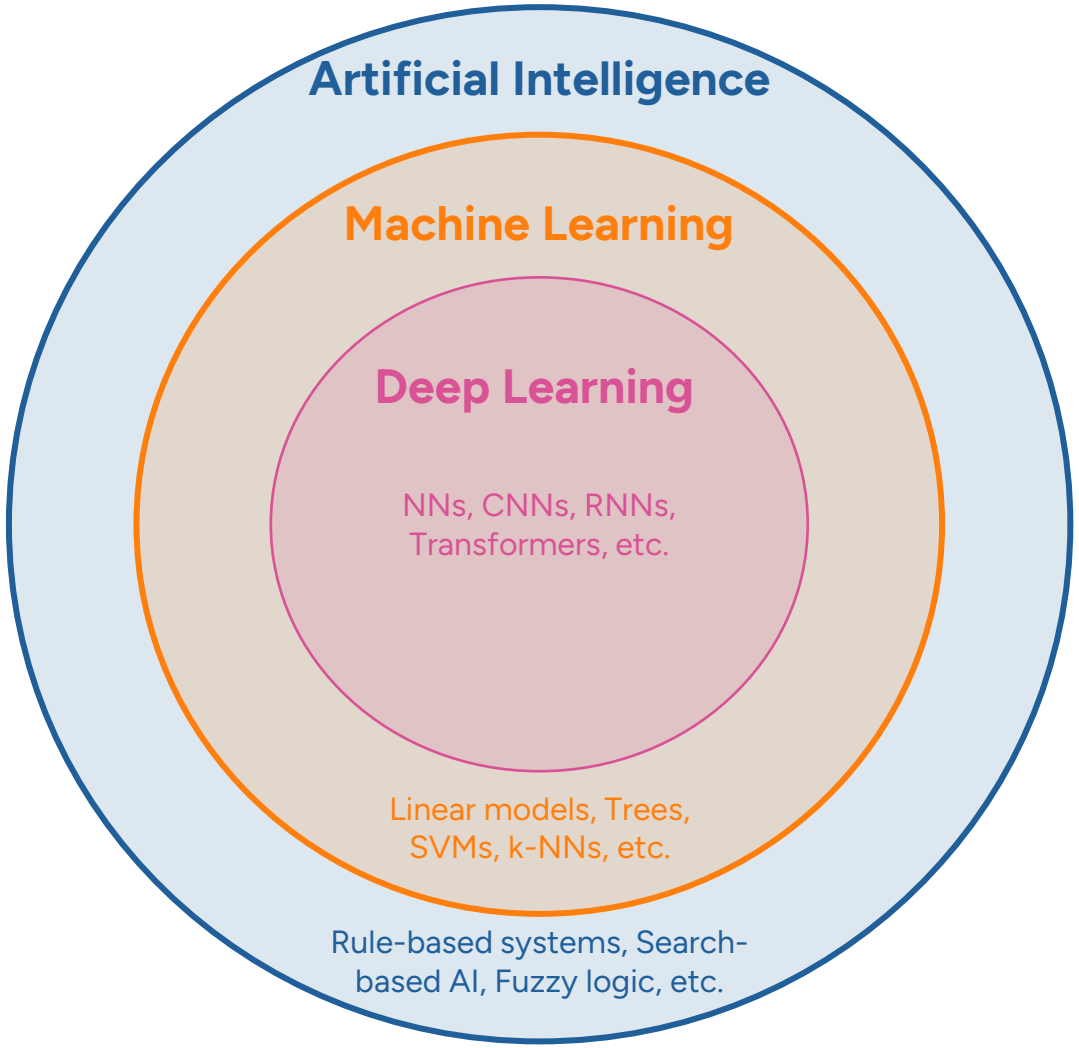
Agenda

- **What is ML?**
  - *Definition: AI, ML, DL*
  - *History of AI*
  - *ML Applications*
  - *Why ML Matters Today*
  - *Main Components*
- **Types of ML**
  - *Supervised Learning*
  - *Unsupervised Learning*
  - *Reinforcement Learning*
- **The ML Workflow**
  - *Key Stages*
  - *Loss Functions*
  - *Optimization Algorithms: Definition*
  - *Optimization Algorithms: Examples*
  - *Model Evaluation*
  - *Overfitting vs Underfitting*
- **ML Models Examples**
  - Supervised Learning: k-NN
  - Supervise Learning: SVM
  - Unsupervised Learning: k-Means
- **DL Models Examples**
  - Perceptron
  - Neural Networks
  - Depth and Width
  - Activation Functions
  - Backpropagation
  - Universal Approximation Theorem
  - Other architectures
  - Scientific ML - Generative AI – LLMs

What is ML?



Definition: AI, ML, DL



"A field of study that gives computers the ability to learn without being explicitly programmed".
-Arthur Samuel, 1959

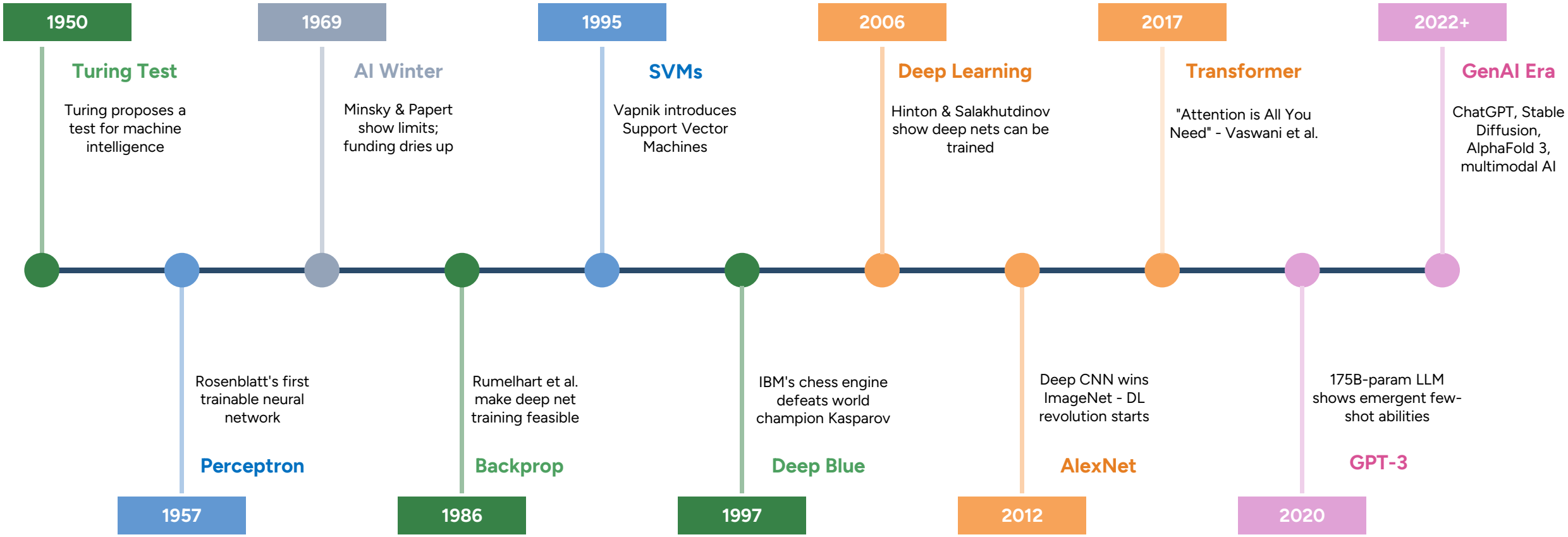
Core Idea
Instead of programming explicit rules, we give the algorithm examples and let it discover the rules itself, by optimizing a mathematical objective over data.

AI – Techniques enabling machines to mimic human intelligence.

ML – Systems improving automatically through experience and data.

DL – Multi-layer neural networks to learn hierarchical features.

History of AI



ML Applications

Search

Ranks billions of pages in milliseconds to find what people meant, and not just what they typed.

Spam filter

Classifies hundreds of millions of emails per day.

Recommendations

Spotify, Netflix, YouTube: the "up next" queue is an ML model.

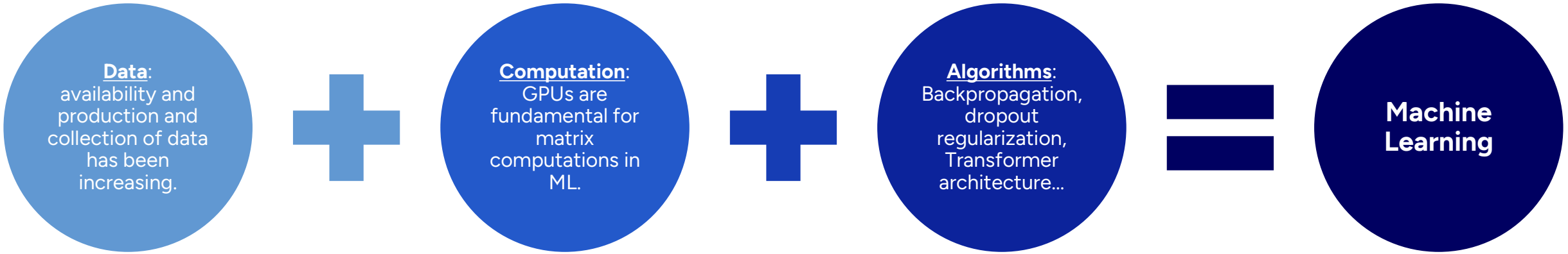
Medical imaging

Detects tumors in radiology scans, often matching specialist accuracy.

Space

Satellite imagery using object detection; telescope data for exoplanet detection; rover navigation; signal processing.

Why ML Matters Today



Main Components of an ML System

1 Raw Data

Collected observation, like images, sensor readings, labels, text, etc. Quality of data is fundamental.

2 Feature Engineering

Transforming raw inputs into meaningful numerical representations the model can use.

3 Model

A parametrized function $f(x;\ \theta)$ mapping inputs to outputs. Architecture encodes inductive bias.

4 Loss Function

Quantifies how wrong the model's predictions are. It guides the learning process, and choosing the right one is fundamental.

5 Optimization

Iteratively adjusts model parameters θ to minimize the loss. An example is gradient descent.

6 Evaluation

Testing generalization on unseen data. Common metrics to use are accuracy, MSE, F1, AUC, etc.

Types of ML

Supervised Learning

The algorithm learns to map input data to a specific output using examples of input-output pairs, defined as **labeled data**. Given a dataset $\{(x_i, y_i)\}$, learns a function f such that $f(x) \approx y$ for new inputs. The model is trained to minimize prediction error over known labels.

Classification

The goal is to predict discrete categories or labels from a fixed set of options.

Examples:

- Is the image a *cat*, *dog* or *bird*?
- Is the galaxy *spiral* or *elliptical*?
- Is the patient *healthy* or *diseased*?

Models: Logistic Regression, Decision Tree, Support Vector Machine (SVM), etc.

Regression

The goal is to predict continuous numerical values.

Examples:

- Predict *temperature* from sensor data.
- *Electricity* consumption in a building.
- Estimate *orbital decay time*.

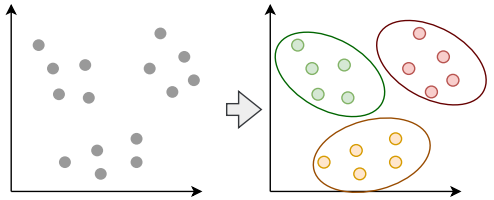
Models: Linear Regression, Ridge/Lasso, Random Forest, Gradient Boosting, etc.

Unsupervised Learning

The algorithm learns patterns, clusters, or compressed representation from **unlabeled data**, entirely from the structure of the input data. By exploring these patterns, it learns the output.

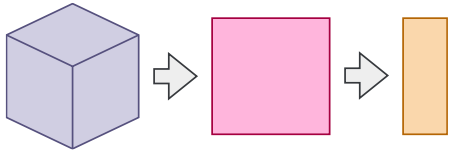
Clustering

It groups data points into clusters based on their similarities or differences.
Models: k-means, Gaussian mixture models, hierarchical, DBSCAN, etc.



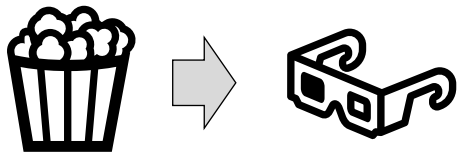
Dimensionality Reduction

It is used to compress high-dimensional dataset reducing the number of features while retaining most of the variance.
Models: Principal component analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE), etc.



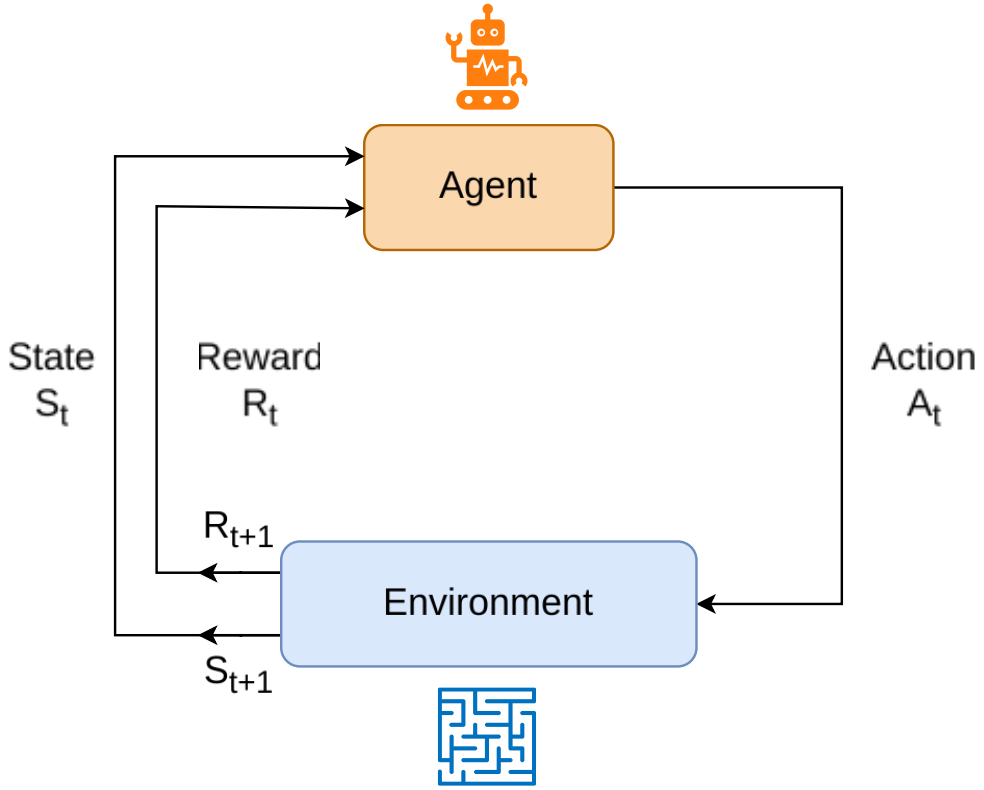
Association Rule

It finds patterns between items in large datasets.
Models: Apriori algorithm, Frequent Pattern-Growth (FP-Growth), etc.



Reinforcement Learning

It focuses on training an **agent** using **interactions** within a dynamical **environment** to maximize the **reward** signal. The goal is to learn a **policy**, a mapping from state to actions, that maximizes cumulative reward over time.



Examples

Game & Simulation

AlphaGo, Atari DQN. Mastering complex strategy games.

Robotics & Autonomy

Rover path planning, robotic arm control, docking.

Spacecraft Control

Orbit transfer optimization, onboard scheduling.

Material Design

Molecule generation optimized for target properties.

The ML Workflow

Key Stages

Framing

1. Problem Definition
What decision will the model support?

Define clearly:
What is the input?
What is the output?

Data work

2. Data Collection
Gather data related to the problem.

3. Feature Engineering
Transform raw data into model-ready inputs.

Data preprocessing is one of the most important parts of the ML workflow.

Modeling

4. Model Selection
Pick an architecture suited to the problem.

5. Training & Optimization
Minimize loss over the training set.

Evaluation

6. Evaluation & Tuning
Measure on test set, adjust hyperparameters.

Production

7. Deployment & Monitoring
Moving to a live production system.

Key metrics:
Accuracy, Precision,
Recall, F1, MSE, RMSE

Loss Functions

A loss function $L(y, \hat{y})$ quantifies how far the model's predictions $\hat{y}$ are from the true values y.
Training a model means minimizing L over the training set.

Mean Squared Error (MSE)

Regression

$$L = \frac{1}{n}\Sigma_i \, (y_i \, - \, \hat{y}_i)^2$$

Penalizes large errors quadratically. Sensitive to outliers.

Mean Absolute Error (MAE)

Regression

$$L = \frac{1}{n}\Sigma_i \, |y_i \, - \, \hat{y}_i|$$

More robust to outliers than MSE.

Binary Cross-Entropy

Classification

$$L = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}))$$

Measures divergence between two distributions.

Categorical Cross-Entropy

Classification

$$L = -\Sigma_k y_k \cdot \log(\hat{y}_k)$$

Generalizes Binary Cross-Entropy to K classes.

Hinge Loss

Classification

$$L = \max(0, 1 \, - \, y \cdot \hat{y})$$

Encourages a margin between classes.

KL Divergence

Classification

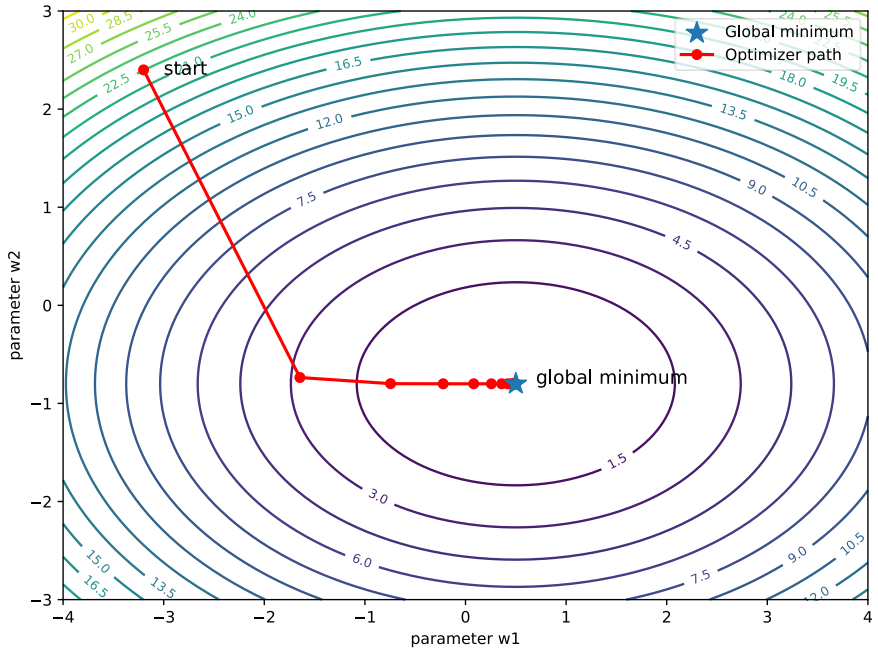
$$L = \Sigma \, p(x) \cdot \log(p(x)/q(x))$$

Measures how one distribution differs from another.

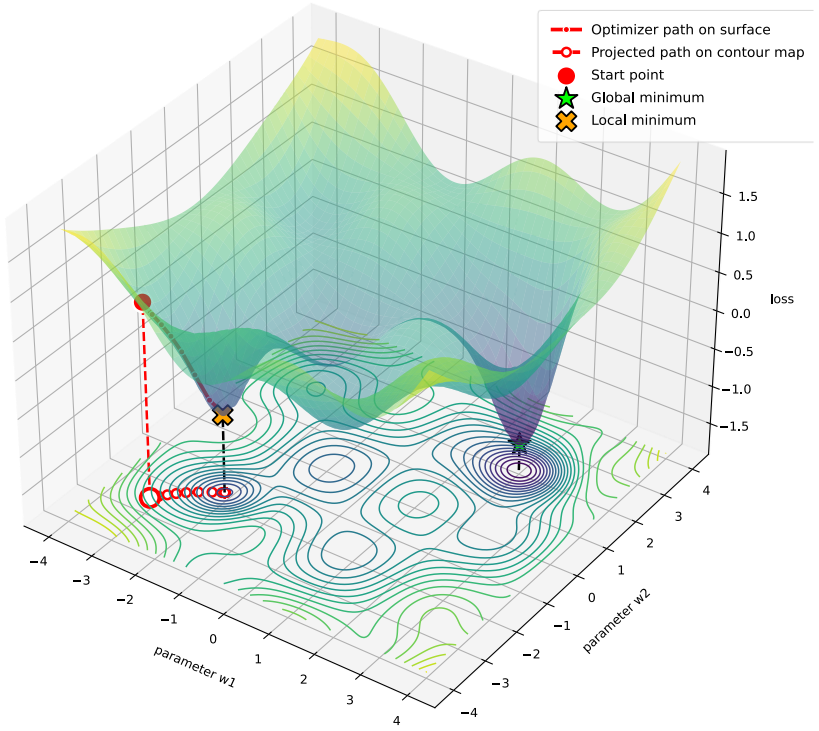
Optimization Algorithms: Definition

Optimization is the process of adjusting the model's parameters to minimize some objective function, i.e. the loss function, while training on data. The goal is to find the best parameters to minimize $L(\theta)$. Optimization algorithms determine how the model move through the parameter space.

Loss Landscape



Loss Landscape



Optimization Algorithms: Examples

First-order uses gradient only: GD, SGD, Momentum, Adam, etc.
Second-order uses gradient + curvature (Hessian): Newton's Method, L-BFGS, etc.

Gradient Descent (GD)

It computes gradient on entire dataset.

Pros: - Easy to analyze
- Converges for convex problems

Cons: - Slow
- Sensitive to learning rate
- Can get stuck in local minima

Update rule (one step):

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L\ (\theta_t)$$

where

- θ_t are model parameters at step t
- η is the learning rate
- $\nabla_\theta L(\theta_t)$ is the gradient of the loss with respect to the parameters

Convergence condition (when to stop):

$$\| \nabla_\theta L(\theta_t) \| < \epsilon$$

Adam

Adaptive per-parameter learning rates.

Pros: - Handles sparse gradients
- Most common default optimizer

Cons: - Memory consuming
- Could struggle to generalize
- More hyperparameters

Step 1 – compute gradient:

$$g_t = \nabla_\theta L(\theta_t)$$

Step 2 – update first moment:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

Step 3 – update second moment:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Step 4 – bias correction ($m_0 = v_0 = 0$):

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Step 5 – parameter update:

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Stochastic Gradient Descent (SGD)

It uses one sample (or mini-batch) at a time.

Pros: - Scales to big datasets
- Fast iterations
- Noise can help to avoid local minima

Cons: - Noisy updates
- Requires tuning learning rate schedule

Newton's Method

Uses curvature information (Hessian).

Pros: - Fast near optimum
- Fewer iterations

Cons: - Expensive in complexity and memory

L-BFGS

Approximates Hessian efficiently.

Pros: - Strong performance on smaller problems
- Less memory

Cons: - Expensive
- Does not work in stochastic

Model Evaluation

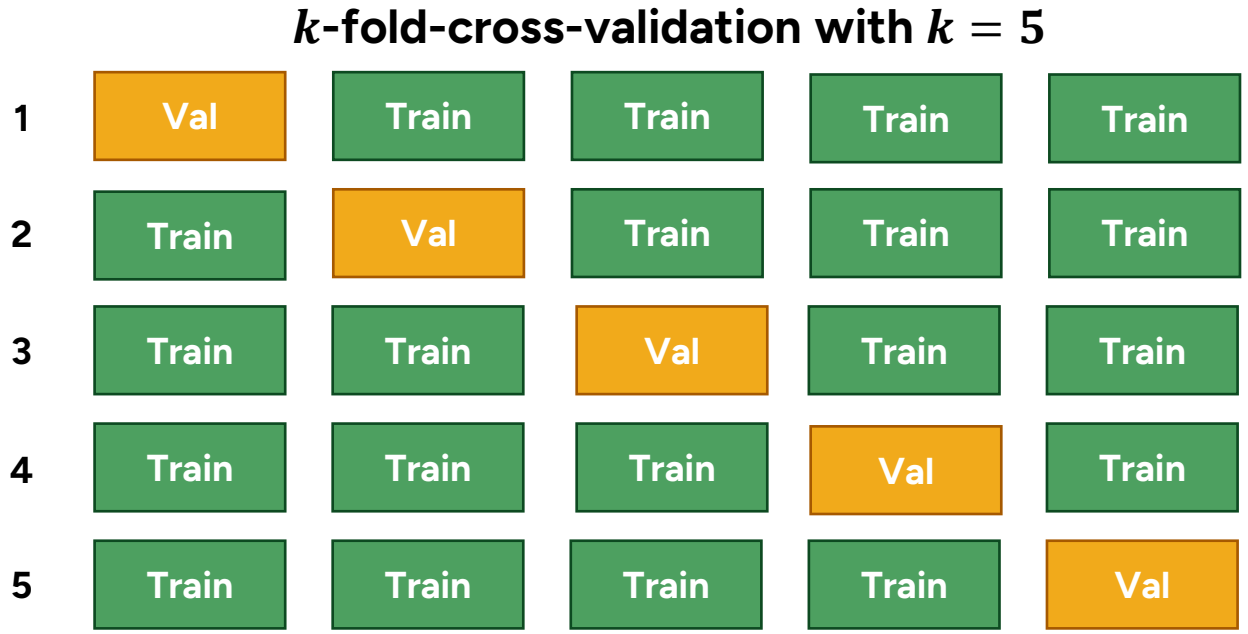
It is fundamental to measure how well the chosen model actually performs.

Data Splitting Strategy

Training Set ~70%	Validation Set ~15%	Test Set ~15%
The model sees this data and adjusts weights.	Used for hyperparameter tuning and early stopping.	Held out data, used once at the end for final testing.

Cross-validation

A single train/validation split could lead to bad results. For example, validation may be unusually easy or difficult.
A performance estimate can then have high variance. With cross-validation, we can repeatedly change which samples are used for validation.
The most common form is ***k*-fold cross-validation.**

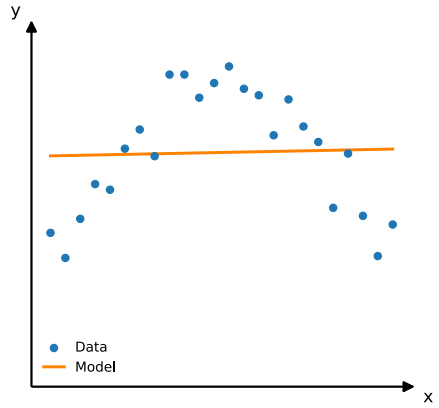
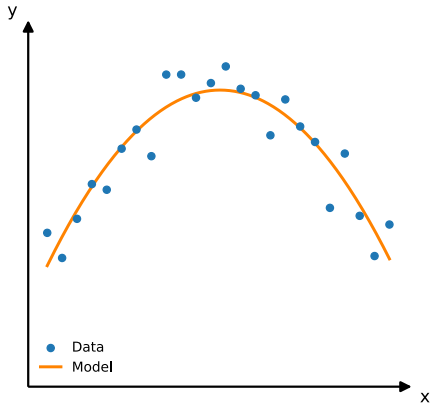
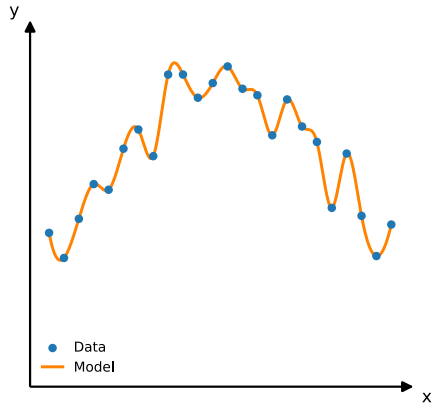
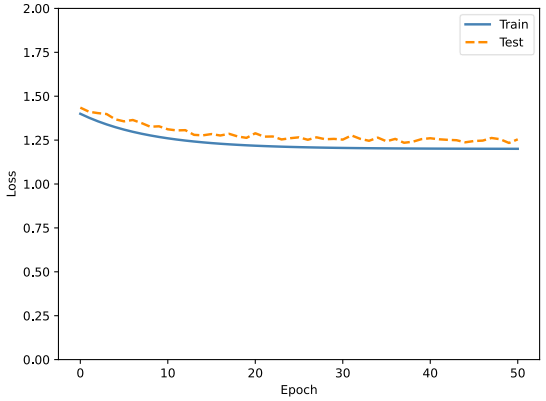
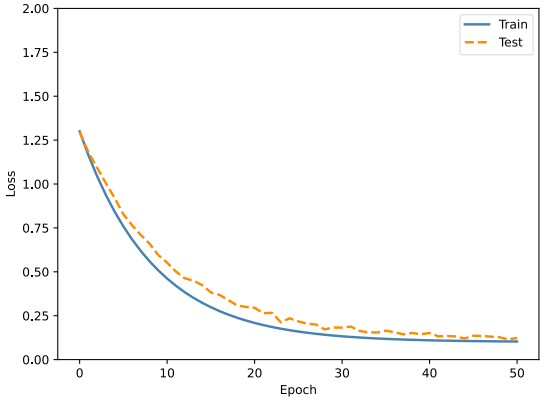
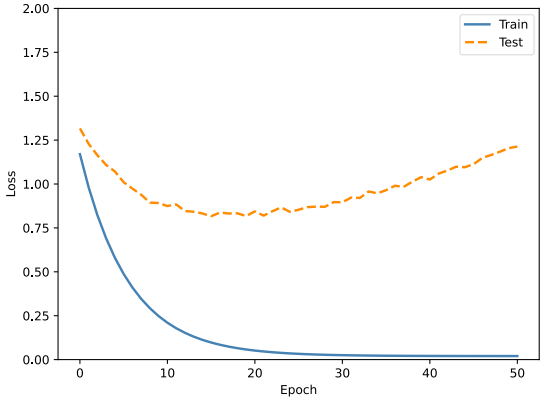


Key Metrics

TP = True Positive FP = False Positive
TN = True Negative FN = False Negative

Accuracy	Acc = (TP + TN) / (TP + TN + FP + FN)	Proportion of all classifications that were correct, whether positive or negative
Precision	Pr = TP / (TP + FP)	Proportion of all the model's positive classifications that are actually positive.
Recall	Rec = TP / (TP + FN)	Proportion of all actual positives that were classified correctly as positives.
F1 Score	F1 = 2 x (Pr x Rec) / (Pr + Rec)	Harmonic mean of precision and recall; balances both metrics.
MSE	$1/n \sum (y - \hat{y})^2$	Average squared difference between predictions and true values.
RMSE	$\sqrt{1/n \sum (y - \hat{y})^2}$	Square root of MSE; average prediction error in the original units.

Overfitting vs Underfitting

	Underfitting	Good Fit	Overfitting
	High Bias, Low Variance	Low Bias, Low Variance	Low Bias, High Variance
Model Fitting			
Loss Function			
	Use a more complex model; add more or better features; reduce regularization strength	Well-tuned regularization; appropriate model complexity; sufficient data	Regularization; more training data; early stopping; simpler model

ML Models Examples

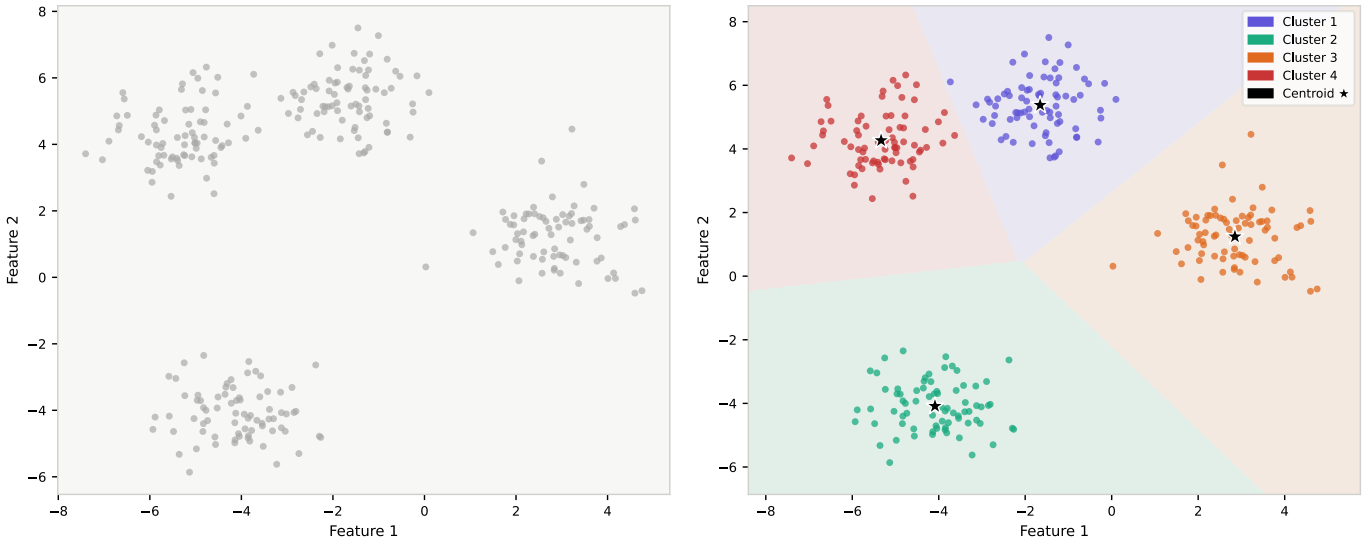
Unsupervised Learning: k-Means

k-means is an unsupervised algorithm that partitions n points into k clusters by minimizing the within-cluster sum of squared distances to the centroid.

How it works:

- Initialize k centroids randomly
- Assign each point to its nearest centroid
- Recompute centroids as the mean of their assigned points
- Repeat until assignments stop changing

Example:



Left plot: Raw unlabeled data.

Right plot: Data after k-means, where color is the assigned cluster, the stars are the learned centroids, and the shaded regions are the decision boundaries.

Key limitation: k must be chosen in advance; sensitive to initialization and outliers.

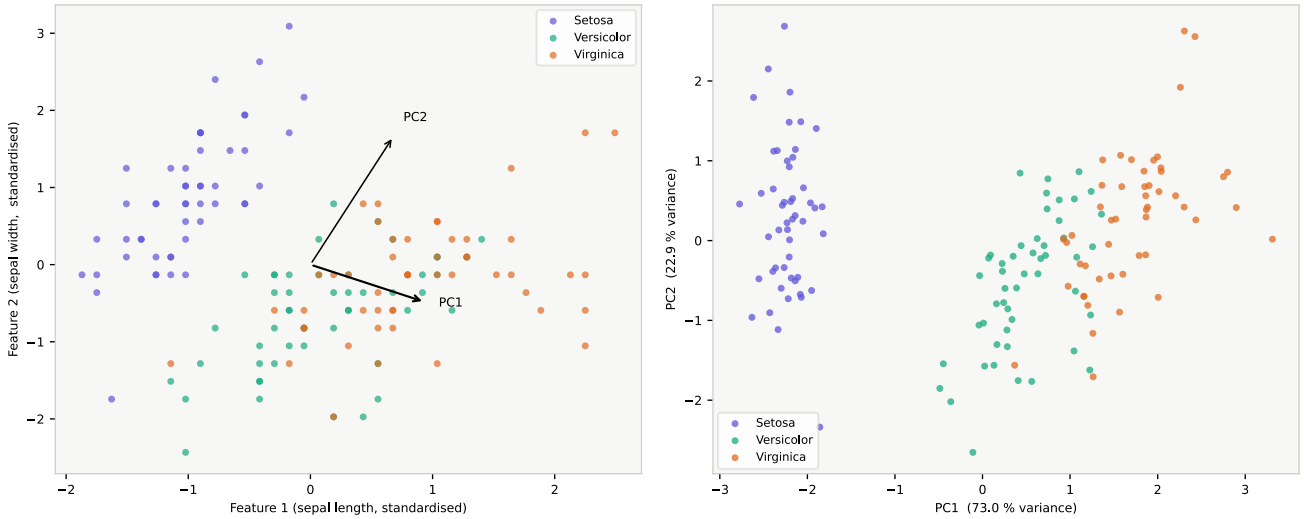
Unsupervised Learning: PCA

PCA is a dimensionality reduction technique that finds the directions of maximum variance in high-dimensional data and projects it onto a lower-dimensional subspace.

How it works:

- Standardize the data (zero mean, unit variance)
- Compute the covariance matrix
- Extract eigenvectors (principal components, PCs) ordered by eigenvalue
- Project data into the top d components

Example:

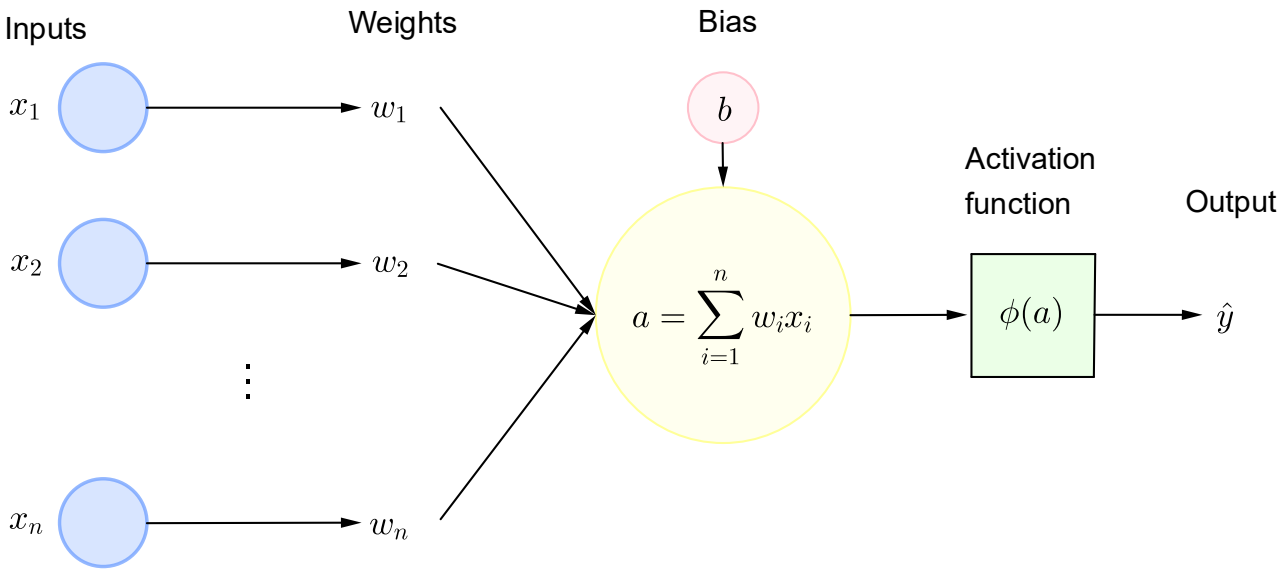


Dataset: Iris dataset (150 flowers x 4 features: sepal length/width, petal length/width.
Left plot: Raw data (only 2 of 4 features are visible), arrows show the first two PCs directions.
Right plot: After PCA, the 4 features are compressed to 2, retaining 95.9% of the total variance.
Key insight: The axes on the right are no longer original features but linear combinations of all 4, ranked by the quantity of variance they explain.

DL Models Examples

Perceptron

Frank Rosenblatt, 1957 – The simplest trainable neural unit



Formula: $\hat{y} = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \phi(\mathbf{w}^\top \mathbf{x} + b)$

For the classic perceptron the activation function is a step function.

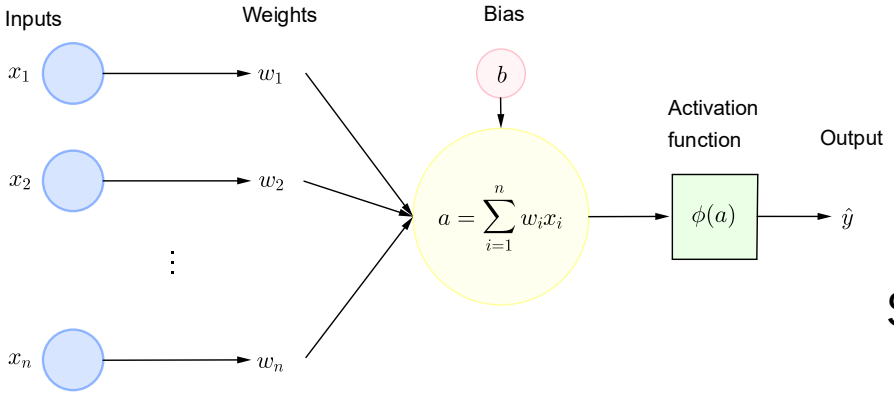
Inputs - x_i
Feature values fed into the neuron.

Weights - w_i
Learnable parameters that encode the importance of inputs.

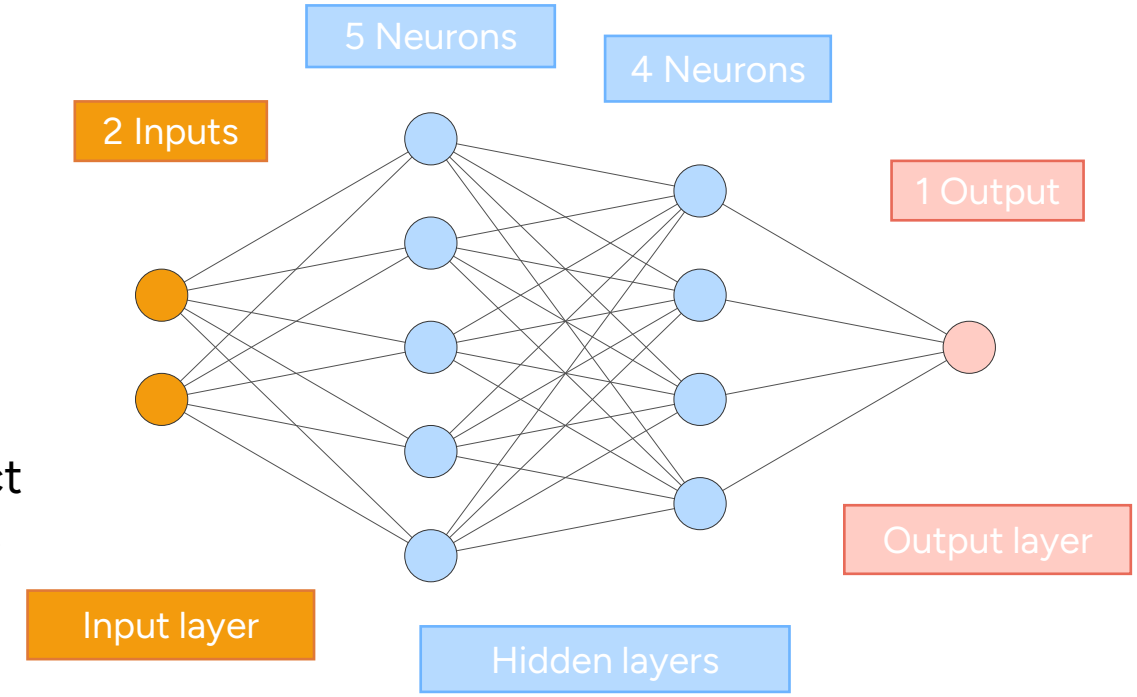
Bias - b
It shifts the decision boundary independently of the inputs.

Activation function - $\phi(a)$
A non-linear function applied to the weighted sum a.

Neural Networks



Stack and connect neurons in layers



Single Layer Perceptron

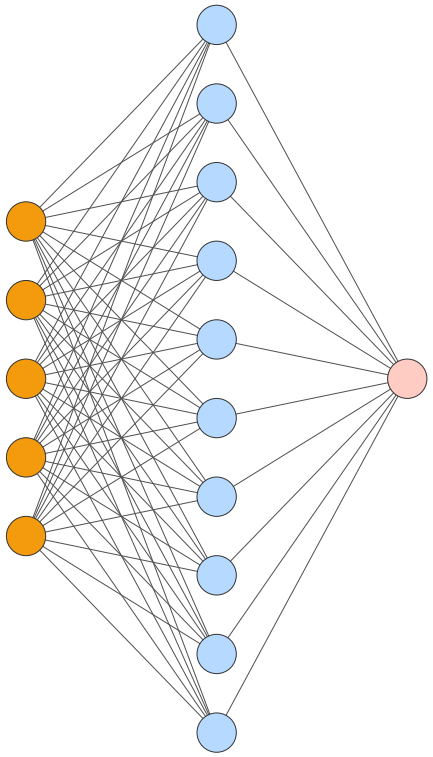
Neural Network

Multiple neurons in layers, non-linear boundaries, Universal function approximator

Neural Networks: Depth and Width

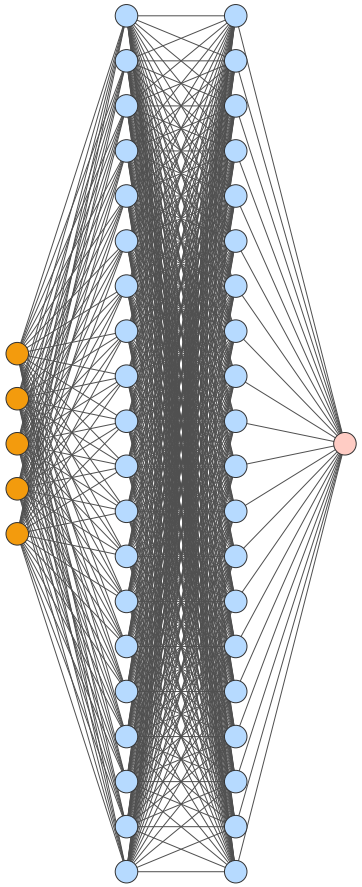
Depth = Number of layers
Width = Neurons per layer

Shallow & Narrow



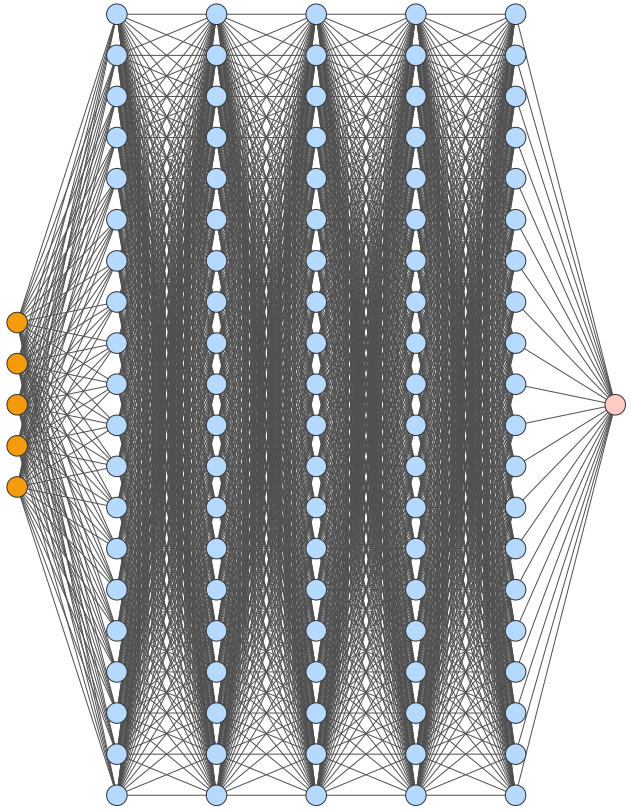
[5, 10, 1]

Moderate Depth & Width



[5, 20, 20, 1]

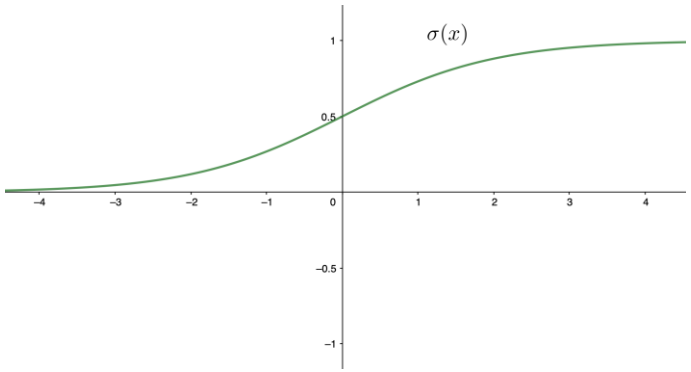
Deep & Wide



[5, 20, 20, 20, 20, 20, 1]

Neural Networks: Activation Functions

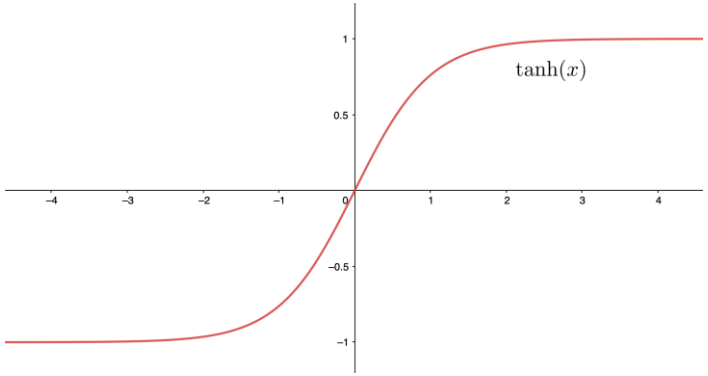
Sigmoid



$$f(x) = \frac{1}{1 + e^{-x}}$$

It outputs values in the range (0,1). Vanishing gradient in deep nets.

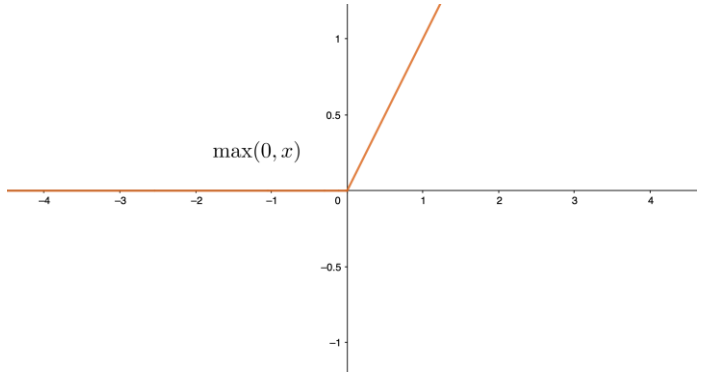
Hyperbolic Tangent (tanh)



$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Zero-centered, range (-1,1). Better gradient flow than sigmoid.

Rectified Linear Unit (ReLU)



$$f(x) = \max(0, x)$$

Fast and sparse. Default for hidden layers. Dead neuron risk at $x < 0$.

Neural Networks: Backpropagation

1. **Forward Pass:** Feed input x through all layers. Compute each neuron's output: $a = f(Wx + b)$.

2. **Compute Loss:** Compare prediction $\hat{y}$ to true label y using a loss function $L(y, \hat{y})$.

3. **Backward Pass:** Apply chain rule to compute $\frac{\partial L}{\partial w}$ for every weight w in every layer. For a neuron:

$$z = wx + b$$
$$a = f(z)$$
$$L = Loss$$
$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w}$$

4. **Weight Update:** Using the optimizer chosen. For example, gradient descent: $w \leftarrow w - \eta \cdot \frac{\partial L}{\partial w}$, where η is the learning rate.

Universal Approximation Theorem

Theorem

Let $f: K \to \mathbb{R}$ be continuous on a compact set $K \subset \mathbb{R}^n$. Then, for every $\epsilon > 0$, there exists a one-hidden layer neural network

$$N(x) = \sum_{i=1}^{m} a_i \sigma(w_i^T x + b\_i)$$

Such that

$$\max_{x \in K} |f(x) - N(x)| < \epsilon .$$

Historical Development:

- **1989 – George Cybenko:** He proved the theorem for single hidden layer networks with sigmoidal activation functions on a compact domain.
- **1991 – Kurt Hornik:** He generalized the result and showed that the approximation capability is a property of feedforward networks and is not restricted to the sigmoid activation function.
- **1993-1995 – Chen & Chen**: They showed strong approximation properties for networks with more general activation functions and established results on approximation rates and multivariate functions.

Neural Networks Architectures

Convolutional Neural Network (CNN)

- Designed for **spatially structured data** (images, video).
- Uses small **filters** across the input to detect local patterns: edges, textures, shapes, objects.
- Key operations: convolution, ReLU, pooling, fully connected.
- Parameters are shared across the space: far fewer weights than a dense network.

Used in: image classification, object detection, medical imaging, autonomous driving, satellite analysis

Recurrent Neural Network (RNN)

- Designed for **sequential data** where order matters.
- Maintains a **hidden state**, a memory of previous steps, updated at each time step.
- Processes inputs one step at a time: $h_t = f(h_{t-1}, x_t)$
- Core weakness: **vanishing gradients** over long sequences, which led to variants like LSTM and GRU.

Used in: time series forecasting, speech recognition, sensor data, early NLP

Transformer

- Replaces recurrence with **self-attention**; every token attends to every other token directly.
- Captures long-range dependencies and is fully **parallelizable** (unlike RNN).
- Built from two blocks: multi-head self-attention + position-wise feed-forward.

Used in: language models, machine translation, vision (ViT), protein folding (AlphaFold2), multimodal AI

Graph Neural Network (GNN)

- Designed for **graph-structured data**: nodes + edges.
- Each node aggregates messages from its neighbors, updates its own embedding, and repeats it across layers.
- **Permutation invariant**: output does not depend on how nodes are ordered.

Used in: drug discovery, molecular property prediction, social networks, recommendation systems, fraud detection

Scientific ML - Generative AI – LLMs

Scientific ML

- Embeds **known physics** (PDEs, conservation laws) into the model via the loss function.
- Models must fit data and satisfy governing equations simultaneously.
- They can be used for forward and inverse problems.

Common approaches: Physics-Informed Neural Network (PINN), Neural Operators, Neural ODE, Symbolic Regression, Surrogate Models

Generative AI

- Models that learn a data distribution and sample new, plausible examples from it.
- Modern generative models are **multimoda**l, i.e., a single model can take text as input and output images, video, or code.

Common approaches: Generative Adversarial Network (GAN), Diffusion Models, Variational Autoencoder (VAE)

Large Language Models (LLMs)

- Transformer trained at scale to predict the next token.
- Two-phase training: pre-training and fine-tuning.
- Key limitations: no persistent memory, hallucination, and no guaranteed factual grounding.

Common approaches: GPT-4o, Claude, Gemini, Llama

Conclusion

Conclusion

1

Knowing your data is fundamental

2

Simpler does not mean irrelevant

3

Make fair evaluations of your model

Explore and find the methods, evaluations, and metrics that work best for your problem and data!

References and Useful Links

ML Foundations

- Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition
- Google, Machine Learning Crash Course
- Kaggle Learn
- Christopher M. Bishop, Pattern Recognition and Machine Learning
- Hugging Face, Papers with Code

Deep Learning & Neural Networks

- Goodfellow, Bengio & Courville, Deep Learning
- DeepLearning.AI, Coursera, Deep Learning Specialization
- 3Blue1Brown, YouTube, Neural Networks
- PyTorch Tutorials

References and Useful Links

Optimization Algorithms

- Diederik P. Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimization
- Distill, Why Momentum Really Works
- CS231n Deep Learning for Computer Vision
- Jorge Nocedal, Stephen J. Wright, Numerical Optimization

Architectures: CNNs, RNNs, Transformers, GNNs

- Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks
- Vaswani, Shazeer, Parmar, Uszkoreit, Jones, N. Gomez, Kaiser, Polosukhin, Attention Is All You Need
- Jay Alammar, The Illustrated Transformer
- Sepp Hochreiter, Jürgen Schmidhuber, Long Short-Term Memory

References and Useful Links

- Colah's Blog, Understanding LSTM networks
- Thomas N. Kipf, Max Welling, Semi-Supervised Classification with Graph Convolutional Networks
- Stanford CS224W: Machine Learning with Graphs

Scientific ML

- Raissi, Perdikaris, Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations
- Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, Anandkumar, Fourier Neural Operator for Parametric Partial Differential Equations
- Bragone, Scientific Machine Learning for Forward and Inverse Problems: Physics-Informed Neural Networks and Machine Learning Algorithms with Applications to Dynamical Systems
- MIT 18.337, Parallel Computing, Scientific Machine Learning, and Modern Agentic Modelling

References and Useful Links

Generative AI & LLMs

- Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio, Generative Adversarial Networks
- Jonathan Ho, Ajay Jain, Pieter Abbeel, Denoising Diffusion Probabilistic Models
- Brown et al., Language Models are Few-Shot Learners
- Hugging Face docs