Neural Networks III

Unsupervised & Reinforcement Learning

Recap

Aspect	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Goal	Learn a mapping from x to y	Discover hidden structure/patterns in x	Learn a policy that maximizes reward
Data	Labeled data	Unlabeled data	State-action pairs, and delayed rewards
Typical output	Classification or regression model	Clusters, low-dimensional embeddings	Policy

Discriminative v.s. Generative

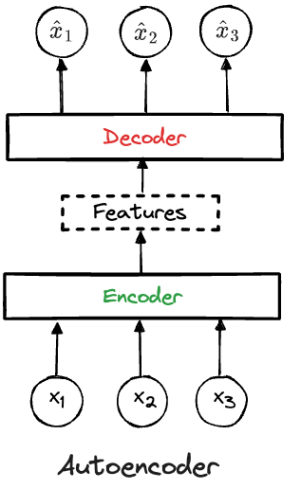
- Discriminative models: Learn $p(y|x)$ — the boundary that best separates classes.
  Predict a label given features.
- Generative models: Learn $p(x, y)$ or $p(x)$ the full data-generation process.
  Generate new samples & predict labels by modeling data distribution.
  Some generative models are trained with unlabelled data (*unsupervised learning*)

Generative Model

- Goal: learn the *data distribution* itself to draw new sample
- Applications
  - Image synthesis (DALL·E, Midjourney)
  - Music creation (Jukebox, Riffusion)
  - Super-resolution & photo up-scaling (Topaz, Snapchat filters)
  - Data augmentation for low-resource tasks.

Autoencoder

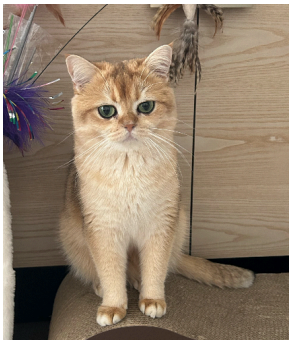
- Goal: to learn the distribution of inputs $p(x)$
- Key concepts
  - Encoder: learn hidden features from inputs
  - Decoder: reconstruct inputs from hidden features
  - Loss function: $\|\hat{x} - x\|_2$
- Applications
  - Image compression
  - Anomaly detection (high reconstruction error suggests outlier).
  - ...
- Notable generalization: variational autoencoder (VAE)
  Use **variational approximation** help with sampling
  (More to be discussed in Bayesian analysis)



Link to illustration

Generative Adversarial Network (GAN)

- Goal: Generate new samples that are similar to the unlabelled training data
- Problem: no labels!
  To be rigorous: there are observations with TRUE labels, but there are no observations with FALSE labels.
- Idea:
  Generate fake images!
- Problem with this idea:
  Data are of very poor quality;
  No feasible ways to improve the generative model.



Generative Adversarial Network (GAN)

- A better idea:
  1. Train a model to judge real v.s. fake
  2. Train another model to generate images to fool the other model
- GAN is a Duel of two nets:

    - Discriminator **D**: judges real vs fake
    - Generator G: generate fake samples from noise
    - Iterate till convergence
- Training objective

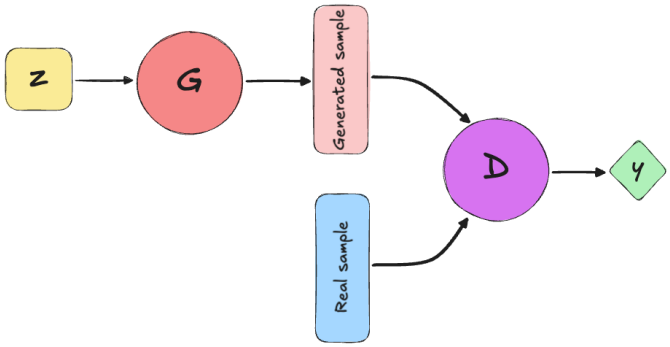
$$\min_G \max_D \{\mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log\{1 - D[G(z)]\}]\}$$

Generative Adversarial Network (GAN)

- Training objective

$$\min_G \max_D \{\mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log\{1 - D[G(z)]\}]\}$$

- Discriminator wants
    - $D(x) = 1$ for real samples
    - $D(x) = 0$ for fake samples
- Generator wants
    - $D(x) = 1$ for fake samples
- Train D and G using alternating gradient updates

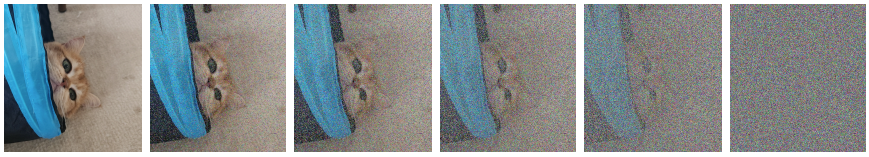


Link to illustration

Diffusion Methods

- Goal: Generate new samples that are similar to the unlabelled training data
- Modified goal: learn a mapping from noise distribution to the true sample distribution

- Idea:
  - Train a model to remove a little bit of noise at each step
- Why this works
  - Injecting noise in true data is easy (diffusion process)
  - Train transformers for denoising



Reinforcement Learning

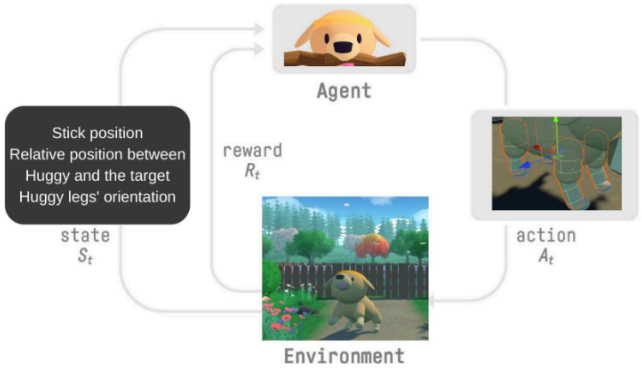
Aspect	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Goal	Learn a mapping from x to y	Discover hidden structure/patterns in x	Learn a policy that maximizes reward
Data	Labeled data	Unlabeled data	State-action pairs, and delayed rewards
Typical output	Classification or regression model	Clusters, low-dimensional embeddings	Policy

Reinforcement Learning

Reinforcement learning (RL) is an interdisciplinary area of machine learning and optimal control concerned with how an intelligent agent should take actions in a dynamic environment in order to maximize a reward signal. Wiki

- State s_t, rewards r_t, action a_t
- Goal: maximize long-term rewards

Training Huggy the dog by HuggingFace:



Two Popular Methods: Intuition

- Q-Learning
  - Learn a table of “quality” scores **Q(s, a)** for every *state–action* pair.
  - Keeping a cheat-sheet of expected scores for every possible move, then following the best score.
  - Best for Solve small, discrete tasks fast
- Policy-gradient Methods
  - Skip the value table and **tune the policy directly**: a neural network outputs action probabilities.
  - Tweaking the instincts of the agent directly to increase probability to take the best action
  - Best use for Act in high-dimensional / continuous spaces, or generate smooth behaviours

---