Introduction

if you can build a product that works 100% of the time, do NOT use ML

ML = learn from experience

Kinds of ML

① supervised learning
predicts targets given inputs
"develop model f_θ that maps input x_i to a prediction $f_\theta(x_i)$

a) Regression
"How many", "How much" problems

b) classification
- look at feature set and predict which class an example belongs to.
easier to do this in terms of probability
- Cross entropy risk/loss function ... potential risk of going wrong.
probability of being wrong

c) tagging
- multi label classification ← classifying classes that are NOT mutually exclusive

d) search & ranking
- which result is MOST useful from unranked results
- relevance filter

diff

e) recommender systems
- emphasis on personalization to specific users in context of recommender systems
- censored feedback → only people who feel strongly about a subject would provide feedback
- feedback loop → product A is bought more b/c recommended

more ... ↓

system thinks product A is better like bought more

(1) Sequence learning

most methods throw away past data after every discrete output ...

for some cases, we want to build a model based on continuous input

eg: sequence of input → sequence of output ⟺ seq 2 seq problem

↳ eg: tagging and parsing audio speech

automatic speech recognition

text to speech

machine translation - maintain alignment of subject / actor when translating between languages

eg: dialogue problems

(2) Unsupervised learning

any kind of vague, open-ended data science problem

eg: clustering

subspace estimation (if linear - principal component analysis)

representation learning

causality and probabilistic graphical models

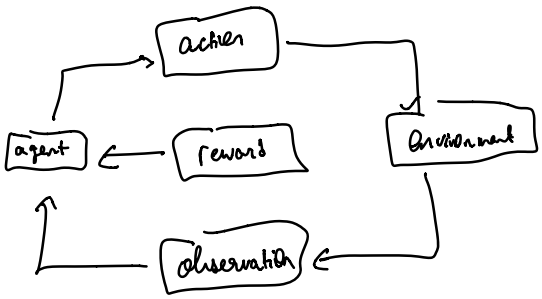
generative adversarial networks (GAN)

we want agents, not predictive models ⟺ considering interactions w/ a model

Reinforcement Learning

general problem statement where an agent interacts with an environment over a series of timesteps.

@ timestep t,



extremely general

(1) credit assignment problem → determine which actions to blame/reward for an outcome

(2) partial observability → current observation may not tell you everything about current state

(3) ~~de~~ at any timestep, deciding whether to exploit best currently known strategy or explore the action space giving up some short term reward in exchange for knowledge.

- When an environment is completely known ⇒ RL problem becomes a Markov Decision Process
- when the state does not depend on the previous actions, we call the problem a contextual bandit problem
- when no state, just set of available actions w/ initially unknown rewards, a multi-armed bandit problem

Hebbian Learning Rule → neurons learn through positive reinforcement

↳ bears for Rosenblatt's perceptron learning algorithm