ML Introduction

generally: making predictions / decisions based on data *

human role → frame issue
gather data + organize/validate
design sol° space
select learning algorithm + parameters → learning from data (= problem of induction)
apply algorithm

solve 2 problems: 1) estimation 2) generalization *
↳ how well what we know a future results?

* { estimation: a b c
generalization: using a b c to predict something (outside of the given range)

problem / sol° → 6 characteristics
↳

1. **Problem class:** What is the nature of the training data and what kinds of queries will be made at testing time?
2. **Assumptions:** What do we know about the source of the data or the form of the solution?
3. **Evaluation criteria:** What is the goal of the prediction or estimation system? How will the answers to individual queries be evaluated? How will the overall performance of the system be measured?
4. **Model type:** Will an intermediate model be made? What aspects of the data will be modeled? How will the model be used to make predictions?
5. **Model class:** What particular parametric class of models will be used? What criterion will we use to pick a particular model from the model class?
6. **Algorithm:** What computational process will be used to fit the model to the data and/or to make predictions?

Supervised Learning

given dataset: $D_n = \{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$ (input → x, output → y) // dataset of "question/answer" pairs

↳ $x^{(i)} \in \mathbb{R}^d$ (vector input in D dimension)

what is supervised learning?
↳ system is given inputs & their associated outputs // y = "target values"

if outputs are drawn from a small, finite set → classification (1)
if outputs are from a large or continuous set → regression (2)

classification: assume $y^{(i)} \in \{-1, 1\}$?!
↳ 2 options = binary classification problem // otherwise, it's multi-class
↳ supervised implies that the output are given for each trainin set input

* goal of classification → given new input $x^{(n+1)}$, predict value of $y^{(n+1)}$

note: $\varphi(x)$ → feature representation $\in \mathbb{R}^d$ // human process of relating event → vector
↳ e.g. how to encode inputs into sets of informative characteristics

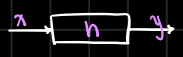
regression: like classification but $y^{(i)} \in \mathbb{R}^*$ // numeric predictions as the output
↳ predicts a cts value
↳ ex: housing prices, stock forecasting

Hypotheses

$h \in \mathcal{H}$

a space of possible solutions ⟶ need a hypothesis class

$y = h(x; \theta)$ // h is some rule that takes an x & spits out a y



} we find an h that agrees with our data in some way

assumptions we can make about data source or soln*: *

↳

- The data are independent and identically distributed.
- The data are generated by a Markov chain.
- The process generating the data might be adversarial.
- The "true" model that is generating the data can be perfectly described by one of some particular set of hypotheses.

⟶ used to reduce size of $\mathcal{H}$ (the set of possible hypotheses) & reduce resources

// what makes h_1 better than h_2?

↳ what makes one prediction better than another?

Loss Function

$L(g, a)$ ⟶ $g \in y \in \{+1, -1\}$ guess ⟶ the quality of a learned model's predictions is often expressed in terms of a loss function
!!! $a \in y \in \{+1, -1\}$ actual

how sad are we that we predicted g when a was the true answer?

types of loss:

1) 0-1 loss // predictions from finite domains

$$L(g, a) = \begin{cases} 0 & \text{if } g = a \\ 1 & \text{if } g \neq a \end{cases}$$

2) squared loss

$$L(g, a) = (g - a)^2$$

3) linear loss

$$L(g, a) = |g - a|$$

4) asymmetric loss // weighted loss

$$L(g, a) = \begin{cases} 1 & \text{if } g = 1 \;||\; a = 0 \\ 10 & \text{if } g = 0 \;||\; a = 1 \\ 0 & \text{otherwise} \end{cases}$$

worst case

how to evaluate quality of prediction? Loss!
↳ theory of rational agency

"select action that minimizes expected loss" // not always best approach

Hypothesis

evaluating hypotheses: training set error?
↳ what makes a hypothesis good?

like: small loss on new data // eventual goal

↳ proxy: small loss on training data // not the full story

training set error: $E_n(h) = \frac{1}{n}\sum_{i=1}^{n} L(\underbrace{h(x^{(i)})}_{\text{pred}}, y^{(i)})$ → the sum over n training samples

↳ each n is an (x, y) pair

$h(x^{(i)})$ → prediction hypothesis would make if given $x^{(i)}$ (e.g. my guess)
$y^{(i)}$ → actual value

↳ sum over all n's divide by n = error for entire training

test error: $E(h) = \frac{1}{n'}\sum_{i=n+1}^{n+n'} L(h(x^{(i)}), y^{(i)})$ → optimize this!

↳ $n+1$ → n' is data that the learning algorithm didn't get to see

Model Type

recall: goal of an ML system is to estimate / generalize
↳ based on data provided

1) No Model → can predict directly from training data
   ↳ in simple cases
   ↳ no use of intermediate data

2) Prediction Rule → 2 step process
   a. "fit" a model to training data
   b. use model to make predictions

   model in PR = some hypothesis $y = h(x, \theta)$ // h is some function
   ↓
   θ = some vector of ≥ 1 parameter values (determined by fitting model to data & fixed)
   ↳ given new $x^{(new)}$, we predict $h(x^{(new)}, \theta)$

   the fitting problem: find value of θ that minimizes training error

   $E_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{L(h(x^{(i)}; \theta), y^{(i)})}_{\text{loss fn}}$

   • note: not always good → overfitting bad

Model Class

model class: set of possible models
↳ parameterized by a vector of parameters Θ

model selection ⟶ pick model class // ex. neural nets
model fitting ⟶ pick model in selected class

ex: linear fn $h(x; \theta, \theta_0) = \theta^T x + \theta_0 \longrightarrow \Theta = (\theta, \theta_0)$

Learning Algorithms

goal: to find a hypothesis w good training set error

- consumes dataset
- returns hypothesis

how to come up with LAs? ⟶ 1. be smart
2. optimization methods // most common + best practice

problem: "what set of computational instructions should we run to find a good model?"

Unsupervised Learning

does not involve I/O pairs
↳ given dataset D, find patterns / inherent structure in D
↳ clustering, density estimation, dimensionality reduction ⟶ 3 USL methods // elaborate later

1) clustering

↳ given samples $x^{(1)}, \dots, x^{(m)} \in \mathbb{R}^n$
↳ goal: find a "partitioning" that groups similar samples // reminds me of k-Nearest Neighbors algorithm?

soft clustering ⟶ assigns "weight" of membership for an element split between clusters

2) density estimation // clustering can be used within here

↳ given samples $x^{(1)}, \dots, x^{(m)} \in \mathbb{R}^n$ drawn i.i.d. from some $Pr(X)$
↳ predict probability $P(x^{(m+1)})$

3) dimensionality reduction

given samples $x^{(1)}, \dots, x^{(m)} \in \mathbb{R}^n$, re-represent samples as points in a d-dimensional space // $d < D$

Sequence Learning

goal: learn a mapping from input sequences $x_0, \ldots, x_n$ to output sequences $y_1, \ldots, y_m$
↳ state machine mapping ... // research later
↳ not completely unsupervised → we are told $x_1 \rightarrow y_1 \ldots x_n \rightarrow y_n$ (and etc.)

Reinforcement Learning *

goal: learn mapping $x \rightarrow y$ without direct instructions of which $x^{(i)} \rightarrow y^{(i)}$
↳ no specified training set

structure: agent interacting w an environment
↳

- The agent observes the current state, $x^{(0)}$.
- It selects an action, $y^{(0)}$.
- It receives a reward, $r^{(0)}$, which depends on $x^{(0)}$ and possibly $y^{(0)}$.
- The environment transitions probabilistically to a new state, $x^{(1)}$, with a distribution that depends only on $x^{(0)}$ and $y^{(0)}$.
- The agent observes the current state, $x^{(1)}$.
- ...

→ find a policy π mapping $x \rightarrow y$ that maximizes long term rewards
→ agent's actions affect rewards & its continued observations of env

other settings: semi-supervised, active learning, transfer / meta learning (later!)