CSCI 699: Theory of ML

What are the goals for the class?

→ How much data is needed for learning? } overarching questions
→ How much computation needed for learning? }
→ How much memory need for learning?
→ Tradeoffs b/w these resources?
→ How to be robust to worst-case perturbations?
" to out-of-distribution data?
How to learn models which are fair?
→ How to understand these questions for deep nets?

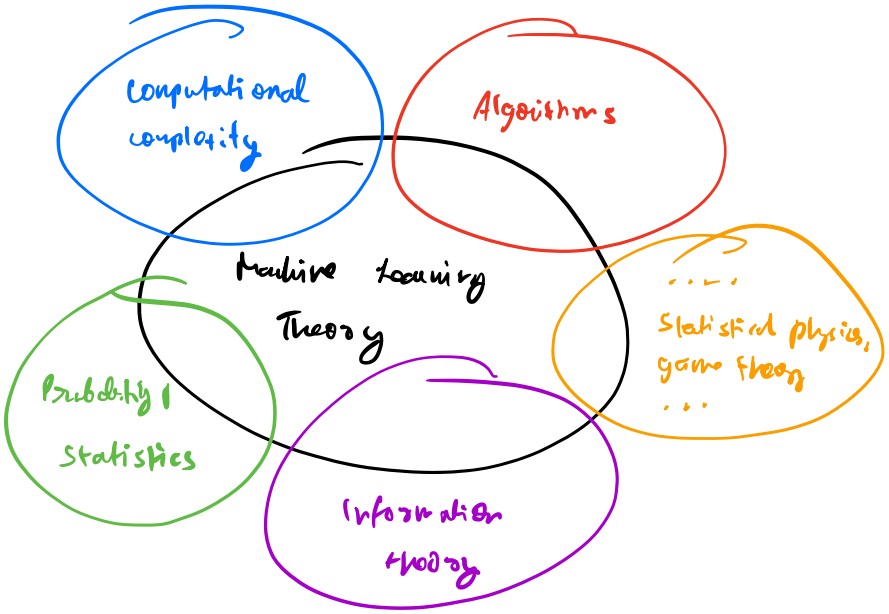
Tools for today's class:

→ What is ML? What is ML theory?
→ Define supervised learning
→ PAC learning
→ PAC learnability results.

what is ML? what is ML theory?

ML: programs that can automate or improve based on data.

Goal of this class: understand the theory behind ML.



why study ML theory?

→ Inform practical algorithms, help in designing new algorithms / models for new settings.

→ Boosting: Valiant - Kearns asked "can you boost a weak learner algo. which gets 51% accuracy on binary classification to a strong learner which gets 95% accuracy-?" led to boosting, Adaboost etc.

→ Optimization algo. for ML.

→ Modern desiderata (fairness, robustness, privacy)

→ To develop a better understanding of learning, computation.

Supervised learning: A gentle introduction

Training set: input/output pairs

Learn some pattern to accurately predict outputs of unseen inputs

for e.g. image classifications

machine translation

SAT solving

$\mathcal{X}$: input space

$\mathcal{Y}$: output space

$\mathcal{X} \subset \mathbb{R}^d$

$\mathcal{Y} \in \{\pm 1\}$

Come up with predictor $f(x)$ which predicts output of x

Loss function: $\ell(f(x), y)$

Classification problem: $\ell(f(x), y) = \mathbb{1}(f(x) \neq y)$ (0/1 loss)

Regression problem: $\ell(f(x), y) = (f(x) - y)^2$ (squared loss)

Distribution P over (x, y)

Risk of supervised learning problem:

$$R(f) = \mathop{\mathbb{E}}_{(x,y)\sim P} \ell(f(x), y)$$

We don't actually know P!

training/test paradigm:

i.i.d. data: get n samples drawn i.i.d from P.

training set S.

$$f^* = \min_{\text{functions } X \to Y} \mathbb{E}_{(x,y) \sim P} \ell(f(x), y)$$

Empirical risk minimizer:

Commit to a set of functions $\mathcal{F}$ beforehand.

$$f_{S,ERM} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum \ell(f(x_i), y_i)$$

Choose function f_S from training set S.

$$R(f_S) - R(f^*) = \underbrace{R(f_S) - R(f_{S,ERM})}_{\text{Computation error}} + \underbrace{R(f_{S,ERM}) - \inf_{f \in \mathcal{F}} R(f)}_{\text{Estimation error}} + \underbrace{\inf_{f \in \mathcal{F}} R(f) - R(f^*)}_{\text{Approximation error}}$$

- Computation: finding the ERM might be expensive.
- Estimation: only have samples.
- Approximation: How much worse is it to work over $\mathcal{F}$.

An example : Linear regression

$X = \{ x : x \in \mathbb{R}^d, \|x\|_2 \le 1 \}$ (unit ball)

$Y = \mathbb{R}$ if $y = \langle w^*, x \rangle$, then approximation error $= 0$.

$\ell(f(x), y) = \frac{1}{2} (f(x) - y)^2$

$\mathcal{F} = \{ f_w(x) = \langle w, x \rangle, w \in \mathbb{R}^d \}$

Computation error : Can compute ERM using closed-form solution.

$$\hat{R}_S(w) = \frac{1}{2n} \sum_{i=1}^{n} (\langle w, x_i \rangle - y_i)^2$$

$$0 = \frac{\partial \hat{R}_S(w)}{\partial w} = \frac{1}{n} \sum_{i=1}^{n} x_i (\langle w, x_i \rangle - y_i)$$

$$\sum x_i y_i = \left(\sum_{i=1}^{n} x_i x_i^T w \right)$$

$$w = \left(\sum x_i x_i^T \right)^{-1} \sum x_i y_i$$

$$w = (x x^T)^{-1} x y$$

$$x = \begin{pmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{pmatrix} \quad d \times n$$

time complexity : $x x^T$: $O(nd^2)$ ($d \times n$ matrix, $n \times d$ matrix)

$(x x^T)^{-1}$: $O(d^3)$

$x y$: $O(nd)$

$(x x^T)^{-1} x y$: $O(d^2)$

$= O(nd^2 + d^3)$

* Gradient descent : $w_0 \leftarrow 0$

$w_{t+1} \leftarrow w_t - \eta_t \nabla \hat{R}_S(w_t)$

$$w_{t+1} = w_t - \eta_t \left(\frac{1}{n} \sum_{i=1}^{n} \underbrace{\phi_i(\langle w_t, \phi_i \rangle - y_i)}_{O(d)} \right)$$

per step complexity : $O(nd)$

in T steps, computation error is e^{-T}.

* stochastic gradient descent

Estimation error : How much data do you need to find a solution which has small error on the distribution ?

Approximation error : If true model is not linear,

PAC learning (Probably Approximately Correct learning)

$X \in \mathbb{R}^d$

Label space: $Y = \{0, 1\}$

Loss function $\ell(f(x), y) = \mathbb{1}(f(x) \neq y)$

Crucial idea: no assumption over the dist. of X

but $y = h^*(x)$ for some $h \in \mathcal{H}$

[realizability assumption]

Learner knows $\mathcal{H}$

Def (PAC learnability):

A hypothesis class $\mathcal{H}$ is PAC-learnable if $\exists$ a learning algorithm with the following property: for every $\epsilon, \delta \in (0,1)$, every distribution D over X and every $h \in \mathcal{H}$, when the algorithm is given $m_{\mathcal{H}}(\epsilon, \delta)$ samples drawn from D & labelled by h, the alg. produces a hypothesis $\hat{h}$ s.t. with probability $1-\delta$, $R(\hat{h}) \leq \epsilon$. (This prob. is over randomness in training set & any internal alg. randomness.)

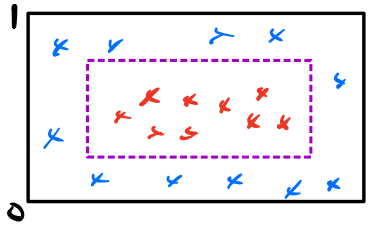
with $m_{\mathcal{H}}(\epsilon, \delta)$ samples

Example: Axis-aligned rectangles.

$X = [0,1] \times [0,1]$

$\mathcal{H}$ = rectangles

D: uniform dist.



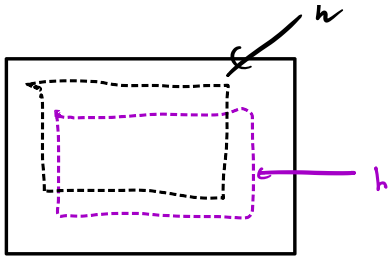
Failure probability δ:

Could get bad set of samples →

?

Approximation parameter ε:

Can only do approximately correct (cannot recover h)



Any dist D:

Maybe the distribution is such that we cannot learn h, but could still predict well