LEC1 DSA5204

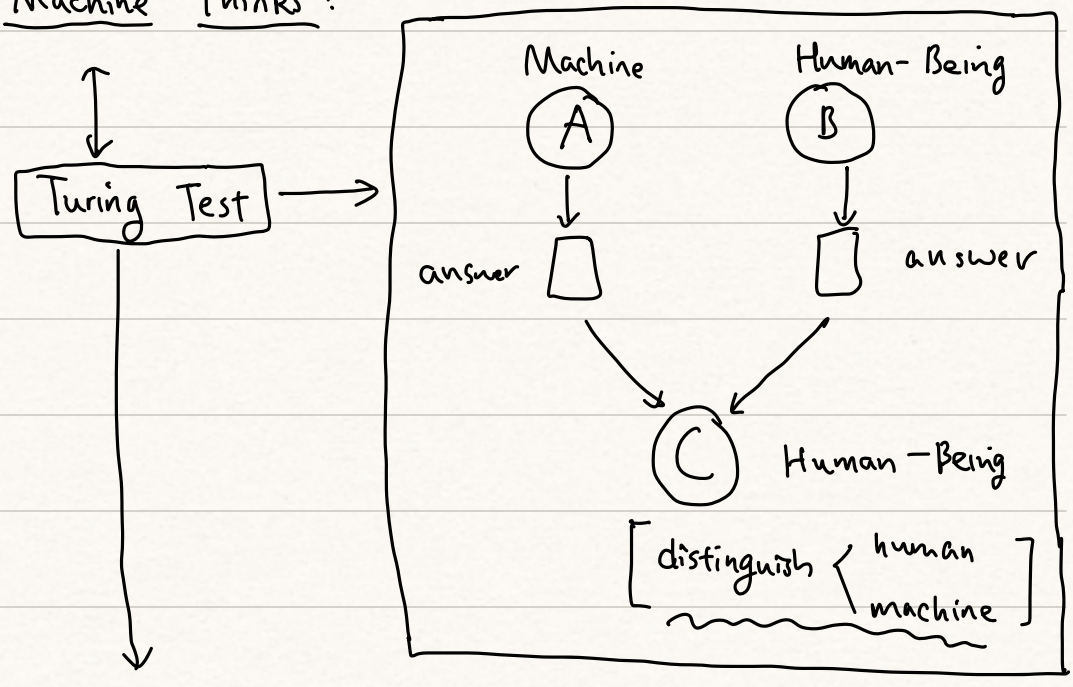
→ Low Yi Rui S17-05-19

AI → { understand / replicate } intelligence

replicate → do intelligent task

MOVIE → <<her>>

Turing → Can Machine Thinks ?



How machine learns to do things human-like ?

(Machine Learning)

→ Adversarial Examples ⇒ try to analyze the change of response when given different inputs

Topics

1. architecture model
2. learning algo ⭐ → learn how to learn
3. practical techniques
4. application
5. active areas of DL research

Machine Learning Basis

1. Task : T ⇒ problem setting
   - prediction
     - regression
     - classification
   - transcription
   - translation
   - imputation
   - generation

   Experience : E ⇒ data

   Performance Measure : P (improve P through E)
   - MSE
   - Accuracy
   - image generation ⇒ hard to define

⇒ Q: How to set learning objective to improve P ?

2. Linear Regression & Canonical Form

Oracle : f^*

Learning : $\hat{f} = \underset{f \in \mathcal{H}}{\operatorname{argmin}}$ Distance (f, f^*) → Empirical Risk Minimization (ERM Framework)

$\hat{f}$: RV depends on training data

$:= \underset{f \in \mathcal{H}}{\operatorname{argmin}}$ $R_{emp}(f, f^*)$ → empirical loss

$= \underset{f \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{2N} \sum_{i=1}^{N} (y_i - f(x_i))^2$

$\mathcal{H}_{linear} := \{ f : f(x) = w^T x, \; w \in \mathbb{R}^d \}$

$= \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2N} \sum_{i=1}^{N} (y_i - w^T x)$

$= \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \| y - Xw \|_2^2$

$X = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} \in \mathbb{R}^{n \times d}$ Design Matrix

Benefit of Linearity → exact solution

⇒ To solve this, we have $\hat{w} = (X^T X)^{-1} X^T y$

For more complicated model (NN), we do not have closed-form sol $\hat{=}$ iterative optimization algo (SGD, ADAM, ...)

3. Model Capacity & { overfitting, underfitting }

$\mathcal{H}_{\text{linear}} \subseteq \mathcal{H}_{\text{quadratic}} \subseteq \cdots$

↓ Small hypothesis space

↓ big hypothesis space

4. Linear Basis Model (extension of LR)

↓ → (improve the model capacity)

enlarge the hypothesis space

$$\mathcal{H}^{\phi}_{\text{LBM}} := \{ f : f(x) = w^T \phi(x),\ w \in \mathbb{R}^D \}$$

$$= \sum_{i=1}^{D} w_i \phi_i(x) \qquad \phi(x) = \begin{pmatrix} \phi_1(x) \\ \vdots \\ \phi_D(x) \end{pmatrix} \in \mathbb{R}^D$$

(ϕ is fixed by design $\Rightarrow$ $\phi : x \in \mathbb{R}^d \longmapsto \phi(x) \in \mathbb{R}^D$)

↳ { polynomial

radial basis (Gaussian)

→ ✡ universal approximator { Fourier Basis

Polynomial Basis

(D sufficiently large)

→ Easy to solve ! $\Longrightarrow$ $\hat{w} = (\Phi^T \Phi)^{-1} \Phi^T y$

$$\Phi = \begin{pmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_n)^T \end{pmatrix} \in \mathbb{R}^{n \times D}$$

5. Large Model Capacity $\Longrightarrow$ over-fitting { large testing error

small training error

(behave bad on unseen data)

use some Learning Theory to interpret this:

a) Generalization Error $\leq$ ① Model Capacity + ② Best we can do given the hypothesis space

b) We are interested in

$$\min_f R_{pop}(f) = \mathbb{E}_{x\sim\mu}[L(f(x), f^*(x))]$$

$\downarrow$ approximation (estimate the population risk)

$$\min_f R_{emp}(f) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i), f^*(x_i))$$

$R_{pop}(f) \approx R_{emp}(f)$

$R_{pop}(f)$ → ground-truth value

$R_{emp}(f)$ → RV estimator w.r.t training data → BIASED

unbiased estimator → testing set split

unbiased estimator → Cross Validation Estimator

6. K-class classification Algo

a) **Binary Classification**

$$\mathcal{H} = \{f : f(x) = \text{sigmoid}(w^T\phi(x)), w \in \mathbb{R}^D\}$$

b) **K-class Classification**

$$\mathcal{H} = \{f : f(x) = \text{softmax}(W\phi(x)), W \in \mathbb{R}^{K\times D}\}$$

Q: How to train?

A: change the objective (KL-divergence for distribution)

$(y_i - g(W\phi(x_i)))^2$ $\quad$ $g: \mathbb{R}^K \rightarrow \{1, 2, \ldots, K\}$

Unreasonable

Disadvantage for MSE Loss

1) Data is nominated, not ordinal → $y_i = 0, 1, 2, 3, \ldots$ for diffent class (ordinal)

2) Hard to optimize (calculate gradient)

Strategy

1) one-hot encoding for y_i

2) use softmax function to generate distribution

$$\mathcal{H} = \{ f : f(x) = \text{softmax}(W\phi(x)),\ W \in \mathbb{R}^{K \times D} \}$$

3) Loss Function Design

$$\begin{cases} L(y, y') = 0 & \text{if } y' = y \\ L(y, y') \approx 0 & \text{if } y' \approx y \end{cases}$$

(distance measure for distribution)

Good choice is : KL-divergence

$$L(y, y') = -\sum_{k=1}^{K} y_k \log y_k'$$

$$= KL(y, y')$$

$\begin{cases} y_k \rightarrow \text{ground-truth} \\ y_k' \rightarrow \text{prediction} \end{cases}$