15-780: Lecture 2

Aditi Raghunathan

Jan 15 2023

North-star models

GPT-4, Claude, Llama

- "Large language models"
- Exceptional Multidisciplinary Performance
- Great as coding assistants, writing assistants etc

GPT4 performance

Simulated exams	GPT-4 estimated percentile
Uniform Bar Exam (MBE+MEE+MPT)[1]	298/400 ~90th
LSAT	163 ~88th
SAT Evidence-Based Reading & Writing	710/800 ~93rd
SAT Math	700/800 ~89th
Graduate Record Examination (GRE) Quantitative	163/170 ~80th
Graduate Record Examination (GRE) Verbal	169/170 ~99th
Graduate Record Examination (GRE) Writing	4/6 ~54th
USABO Semifinal Exam 2020	87/150 99th–100th
USNCO Local Section Exam 2022	36/60
Medical Knowledge Self-Assessment Program	75%
Codeforces Rating	392 below 5th

North-star models

OpenAI’s CLIP model

- Bridges vision and language
  - Text-to-image generators
- General purpose capable vision models
- Image search and retrieval

Supervised learning

- At their core, LLMs and CLIP are **prediction models**

$$\text{Input} \xrightarrow{f} \text{Target}$$

MNIST example

Input	Target
	6
	2

LLMs

Input	Target
"The sun rises in the"	*"east"*
"After missing the bus, she decided to walk to the"	*"store (or "office" or "park")*
$x_1, x_2, \ldots x_{i-1}$	$p(x_i \mid x_1, x_2, \ldots x_{i-1})$

CLIP

Given B images $I^{(1)}, I^{(2)}, \ldots I^{(B)}$ and corresponding captions $T^{(1)}, T^{(2)} \ldots T^{(B)}$ we shuffle them up and model

Input

"A red bicycle parked beside a wooden fence"

"A playful golden retriever sitting on green grass"

"A curious tabby cat lounging on a cozy couch"

Target

the correct pairings of a batch of (image, text) examples

Supervised learning

- At their core, LLMs and CLIP are **prediction models**

$$\text{Input} \xrightarrow{\quad f \quad} \text{Target}$$

How do we obtain f?

How to do prediction?

Input

Target

6

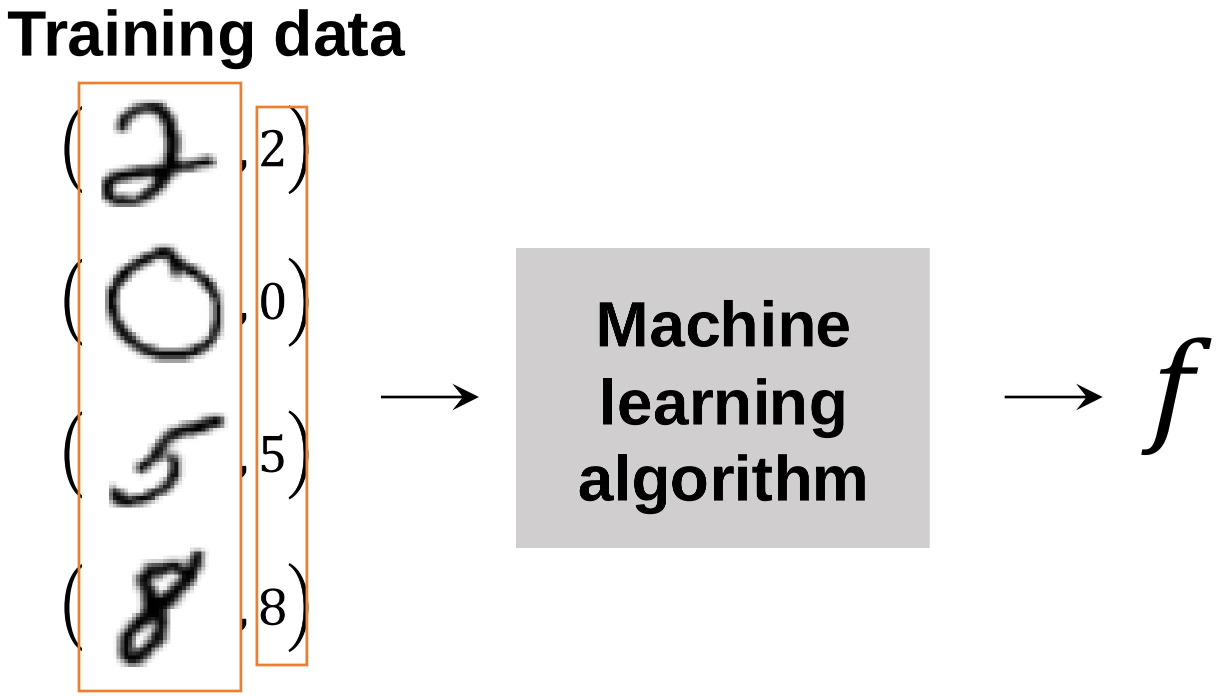
2

Write a program that classifies handwritten 6s from 2s

Supervised learning

Input $\longrightarrow f \longrightarrow$ Target

- Collect a large volume of images and their corresponding numbers
- Write ML algorithm “figure out” what f is

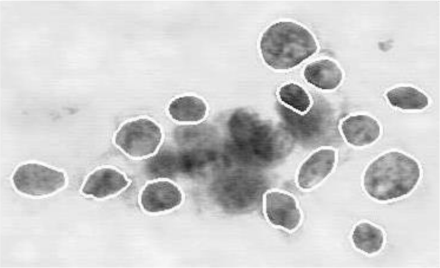


Notation

- Inputs $x \in \mathbb{R}^n$ $[x_1, x_2 \dots x_n]$ x_i: i^{th} index
- Target $y \in \{1, \dots k\}$ k possible outcomes
- Multiple: $x^{(1)}, x^{(2)} \dots x^{(B)}$

  $y^{(1)}, y^{(2)} \dots y^{(B)}$

Input representation



- Breast tumor classification example
  - Numerical description of the physical and structural characteristics of the cell

$[\text{avg size}, \text{smallest size}, \dots] \in \mathbb{R}^n$

- Images 

28×28

  - Flattened representation

$[x_1, x_2 \dots] \in \mathbb{R}^{28 \times 28}$

x_i = i^{th} pixel when flattened

  - Tensor representation (Channels, Height, Width)

Input representation: text

- Tokenization and vocabulary

"the sun rises in the"
x_1 x_2 x_3 x_4 x_5
10 23 42 12 10

just words for now
$|V|$ items
each word has some index in vocabulary

- One-hot encoding $x_i = j^{th}$ index $\Rightarrow \psi(x_i) = [0, 0, \dots, 1_k, \dots, 0]$ $|V| \times L$

Input rep of $x_1 \dots x_L$ is a vector in $\mathbb{R}$
concatenation of one-hot vectors of each word $[\psi(x_1), \psi(x_2) \dots \psi(x_L)]$

- Embedding : in practice, $\phi : \mathbb{R}^{|V|} \rightarrow \mathbb{R}^d$ a d-dim

embedding of each word is concatenated $[\phi(x_1), \dots \phi(x_L)]$

Hypothesis and hypothesis class

Hypothes function $h_\theta : \mathbb{R}^n \longrightarrow \mathbb{R}^k \longrightarrow$ scores of k classes

$\rightarrow$ use real-valued scores for ease of optimization

$\rightarrow$ $[h_\theta(x)]_i$: score of class i

$\rightarrow$ prediction from $h_\theta(x)$: typically argmax

Hypothesis and hypothesis class

linear function $\theta \in \mathbb{R}^{k \times n}$

$$h_\theta(x) = [\theta_1^T x, \ \theta_2^T x \ \ldots \ \theta_k^T x]$$

each score is a different linear combination of inputs

$$\mathcal{H} = \{h_\theta \mid \theta \in \Theta\}$$

set of hypothesis functions parametrised by Θ

hypothesis class

What makes a good hypothesis?

Loss functions

Loss function $\ell(h_\theta(x), y) \in \mathbb{R}$
measures how well prediction
$h_\theta(x)$ matches with y

- Zero-one error

$$\ell_{0-1}(h_\theta(x), y) = \mathbb{1}[\arg\max(h_\theta(x)) \neq y]$$

→ loss is zero if argmax prediction from $h_\theta(x)$ matches y

→ difficult to optimize!!

Loss functions

Given $h_\theta(x) \in \mathbb{R}^k$

Ex: $h_\theta(x) = [1, -2.5, 4]$

- Convert to "probabilities"
  - Positive : via exponentiation
  - Sum to 1 normalization $N = \sum_{i=1}^{k} \exp(h_\theta(x))_i$
- Softmax

$[\exp(1), \exp(-2.5), \exp(4)]$

$N = \exp(1) + \exp(-2.5) + \exp(4)$

"probabilities" $= \left[\frac{\exp(1)}{N}, \frac{\exp(-2.5)}{N}, \frac{\exp(4)}{N}\right]$

$$\mathrm{softmax}(h_\theta(x)) = \left[\frac{\exp(h_\theta(x)_i)}{\sum_{i=1}^{k} \exp(h_\theta(x))_i}\right]$$

Loss functions

- Cross entropy loss $\ell_{CE}(h_\theta(x), y) = -\log \text{softmax}(h_\theta(x))_y$

neg log probability of label y as predicted by converting $h_\theta(x)$ to probabilities

- Maximize likelihood of observed data

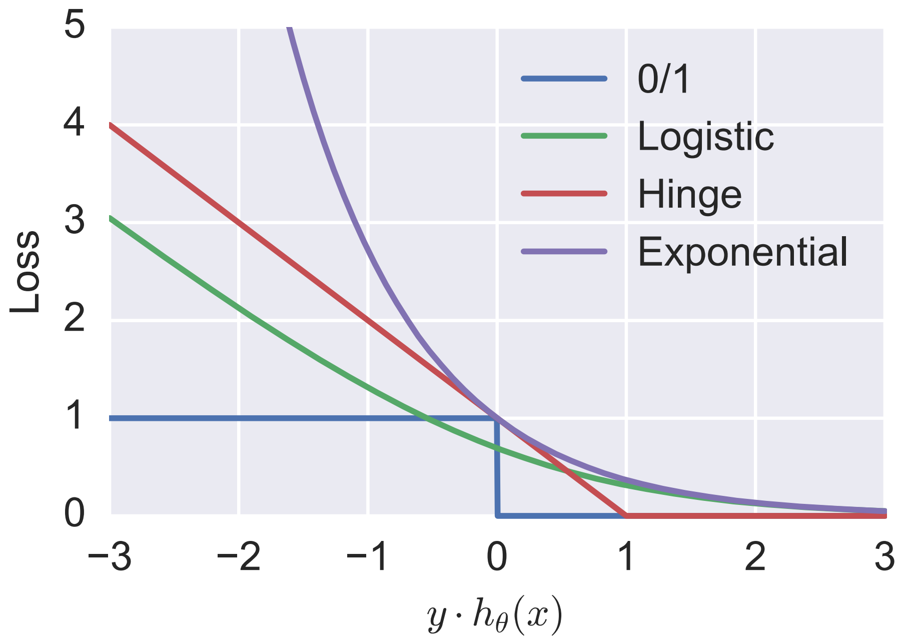
$$p_\theta(y|x) = \text{softmax}(h_\theta(x))_y$$

under this probability defn, minimizing cross entropy is same as maximizing log likelihood

Loss functions: binary classification

$\to \mathbb{R}$

Model only one logit $h_\theta(x)$ as the "score" of positive class and convert to a probability via sigmoid function



$$\ell_{0/1} = 1\{y \cdot h_\theta(x) \le 0\}$$
$$\boxed{\ell_{\text{logistic}} = \log(1 + \exp(-y \cdot h_\theta(x)))}$$
$$\ell_{\text{hinge}} = \max\{1 - y \cdot h_\theta(x), 0\}$$
$$\ell_{\exp} = \exp(-y \cdot h_\theta(x))$$

Loss functions: language modeling

$$-\sum_{i=1}^{L} \log p(x_i \mid x_1, \ldots x_{i-1})$$

"the sun rises in the east"

$$p(x_i \mid x_1 \ldots x_{i-1}) = \text{softmax}\big(\underbrace{h_\theta(x_{1 \ldots i-1})}_{\in \mathbb{R}^{|V|}}\big)$$

Loss functions: CLIP

For every pair of image $I^{(i)}$, text $T^{(j)}$,

we compute "scores" $S^{ij} = \dfrac{\phi(I^{(i)}) \cdot \phi(T^{(j)})}{\|\phi(I^{(i)})\| \cdot \|\phi(T^{(j)})\|}$

Prediction task 1:

which text corresponds to image $I^{(i)}$ ($I^{(i)}$ is input, $y=i$ is target prediction)

$h_\theta(I^{(i)}) = [S^{i1}, S^{i2}, \dots S^{iB}]$

loss: $\text{Cross-entropy}(h_\theta(I^{(i)}, i)) = -\log \dfrac{\exp(S^{ii})}{\sum_{j=1}^{B} \exp(S^{ij})}$

Analogous loss for predicting which image maps to text $T^{(i)}$: $-\log(\exp(S^{ii})) / \sum_{j=1}^{B} \exp(S^{ji})$

Piazza poll

Training procedure

- Minimize train loss

Training data:

Inputs and corresponding targets

$X^{(1)}, y^{(1)}$

$X^{(2)}, y^{(2)}$

$\vdots$

$X^{(B)}, y^{(B)}$

$$\boxed{\underline{\text{Train loss}}(\theta) = \sum_{i=1}^{B} \ell\left(h_\theta(X^{(i)}), y^{(i)}\right)}$$

Coming up

- How to minimize training loss? **(Optimization)**
  - (Stochastic) gradient descent
  - Momentum-based methods
  - Adaptive gradient methods

- When/why does that work? **(Generalization)**
  - "Classical" view
  - Revisit after discussing deep networks

Any questions?