LLM Basics I

- An LLM learns a model that uses the current

  pre-context

  to predict

  the next token.

- We give the following example:

Pre-context		The next token
I	$\rightarrow$	am
I am	$\rightarrow$	a
I am a	$\rightarrow$	machine
I am a machine	$\rightarrow$	learning
I am a machine learning	$\rightarrow$	researcher

LLM Basics II

- This setting is called autoregressive prediction in machine learning.
- Assume we have D documents. For document i, let us assume

  $\boldsymbol{x}_{i,1:j}$: contexts from 1st to jth words
  $\boldsymbol{y}_{ij}$: $(j+1)$st word.
- Then we have many

  (target, instance)

  pairs for training.
- For example, we have

  $\boldsymbol{x}_i$: "I am a machine learning researcher" and

LLM Basics III

$$\begin{aligned} \boldsymbol{x}_{i,1:1} &\quad \text{I} \\ \boldsymbol{x}_{i,1:2} &\quad \text{I am} \\ &\vdots \end{aligned}$$

- From the basic concept of supervised learning, we can solve an optimization problem

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{D} \sum_{j} \xi(f(\boldsymbol{\theta}; \boldsymbol{x}_{i,1:j}), \boldsymbol{y}_{ij})$$

to obtain the model, where

  - $\boldsymbol{\theta}$ is the model parameters,

LLM Basics IV

- $f(\boldsymbol{\theta}; \boldsymbol{x}_{i,1:j})$ is a function that gives an approximation of the target $\boldsymbol{y}_{ij}$,
- $\xi(\cdot)$ is the loss function.

- However, training a supervised learning model requires that

$$\boldsymbol{y}_{ij} \text{ and } \boldsymbol{x}_{i,1:j}$$

are vectors, but they are now word sequences.

LLM Basics V

- Having vectors is important because, for example, we can then define a simple loss function such as

$$\|\boldsymbol{y}_{ij} - f(\boldsymbol{\theta}; \boldsymbol{x}_{i,1:j})\|^2$$

  to ensure that $\boldsymbol{\theta}$ leads to an output as close to $\boldsymbol{y}_{ij}$ as possible.
- From what we learned in supervised learning, if instances and targets are vectors, and we agree that the following procedures can be implemented:
  - network architectures (i.e., our f function),
  - stochastic gradient methods,

LLM Basics VI

- automatic differentiation,

then we "believe" that an LLM can be implemented.

- We will explain how we can convert $(\boldsymbol{y}_{ij}, \boldsymbol{x}_{i,1:j})$ to vectors.
- Subsequently, we abuse the notation a bit so that $\boldsymbol{x}_{i,1:j}$ is a word sequence as well as the converted vector.

Tokenization I

- We split each document to several tokens and map each token to an integer ID.
- An example is as follows:

Document	I am a machine learning researcher						
	↓						
Tokenized text	I	am	a	machine	learning	researcher	
	↓						
Token IDs	25	7	100	2	10	1000	

Tokenization II

- We let "Vocabulary" be the huge dictionary of all words (tokens) considered.
- Thus, |Vocabulary| is its size.
- For easy understanding, we consider a word as a token.
- In practice, we often break a word to several tokens.

Tokenization III

- If each word is a token, for every word in the sentence we can check the corresponding ID in the Vocabulary dictionary:

  the **25th** word in our dictionary is **“I”**,
  the **7th** word in our dictionary is **“am”**,
  the **100th** word in our dictionary is **“a”**,
  the **2nd** word in our dictionary is **“machine”**,
  the **10th** word in our dictionary is **“learning”**,
  the **1000th** word in our dictionary is **“researcher”**.

Position Information I

- We can let

$$\boldsymbol{x}_{i,1:j} \in \{0,1\}^{|\text{Vocabulary}|} \tag{1}$$

be an indicator vector to reflect the occurrence of its words.

- For example, if

$$\boldsymbol{x}_{i,1:2} \text{ is “I am”} \tag{2}$$

Position Information II

then we have

$$\boldsymbol{x}_{i,1:2} = [\overbrace{\underbrace{0, \ldots, 0, 1}_{7\text{th}}, 0, \ldots, 0, 1}^{25\text{th}}, 0, \ldots]^\top.$$

- However, from the indicator vector and the Vocabulary, we can extract only the set of words in $\boldsymbol{x}_{i,1:j}$ instead of the word sequences.
- That is, we cannot recover the order of the words.
- We need another indicator vector to reveal the position information.

Position Information III

- To do so, first we choose a maximum length T.
- No matter how long the document is, we always consider only up to the Tth word.
- Then we can expand the indicator vector in (1) to

$$\boldsymbol{x}_{i,1:j} \in \{0,1\}^{|\text{Vocabulary}| \times T}$$

so that

$$\boldsymbol{x}_{i,1:j} = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccccccc} & & & & 7 & & & & 25 & \end{array} \\ \begin{bmatrix} 0 & \dots & 0 & 0 & 0 & \dots & 0 & 1 & 0 & \dots \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 & 0 & 0 & \dots \\ & & & & \vdots & & & & & \\ & & & & \vdots & & & & & \end{bmatrix} \end{array}.$$

Position Information IV

- Our $\boldsymbol{x}_{i,1:j}$ is now a matrix. If we insist on that the input is a vector, we can do a conversion by, for example, concatenating all columns.
- With the position information, we can exactly recover our word sequence.

Output of the Network I

- For output, assume our network can generate

$$\hat{\boldsymbol{y}}_{ij} = f(\boldsymbol{\theta}; \boldsymbol{x}_{i,1:j}) \in [0,1]^{|\text{Vocabulary}|}$$

  indicating the occurrence probability of each word.
- Consider the example in (2). To predict the next token, we have

$$\boldsymbol{y}_{ij} \text{ is “a”}$$

  from the ground truth of the training data.

Output of the Network II

- Thus we hope that our predicted $\hat{\boldsymbol{y}}_{ij}$ satisfies

$$(\hat{\boldsymbol{y}}_{ij})_{100} \approx 1$$

and

$$\text{other elements of } \hat{\boldsymbol{y}}_{ij} \approx 0.$$

Autoregressive Settings I

- Up to now what we have is

$$f(\boldsymbol{\theta}; \boldsymbol{x}_{i,1:j}) \in [0,1]^{|\text{Vocabulary}|}$$
$$\uparrow$$
$$\text{network } f(\boldsymbol{\theta}; \boldsymbol{x}_{i,1:j})$$
$$\uparrow$$
$$\boldsymbol{x}_{i,1:j} \in \{0,1\}^{|\text{Vocabulary}| \times T}$$

Autoregressive Settings II

- Everything looks good so far. However, as described later, an issue is that we treat

$$\boldsymbol{x}_{i,1:1}, \boldsymbol{x}_{i,1:2}, \ldots$$

  as independent vectors.
- We have inputs like

$$\begin{aligned} \boldsymbol{x}_{i,1:1} &\quad \text{I} \\ \boldsymbol{x}_{i,1:2} &\quad \text{I am} \\ \boldsymbol{x}_{i,1:3} &\quad \text{I am a} \end{aligned}$$

Autoregressive Settings III

- They are related. For example, $\boldsymbol{x}_{i,1:1}$ is a sub-string of $\boldsymbol{x}_{i,1:2}$, so we should not treat them as independent vectors.
- To have efficient training and prediction, we must have a way so that the same operation is not conducted multiple times.
- We know

$$\boldsymbol{x}_{i,1:j} \text{ includes } x_{i,1}, x_{i,2}, \ldots, x_{i,j}.$$

Autoregressive Settings IV

- Therefore, we should design $f(\cdot)$ a way so that the following sequence of calculation can be efficiently done.

$$f(\boldsymbol{\theta}; \boldsymbol{x}_{i,1:1}), \ldots, f(\boldsymbol{\theta}; \boldsymbol{x}_{i,1:j}), \ldots \tag{3}$$

- That is, we hope things are coupled as indicated in Figure 1.

Autoregressive Settings V

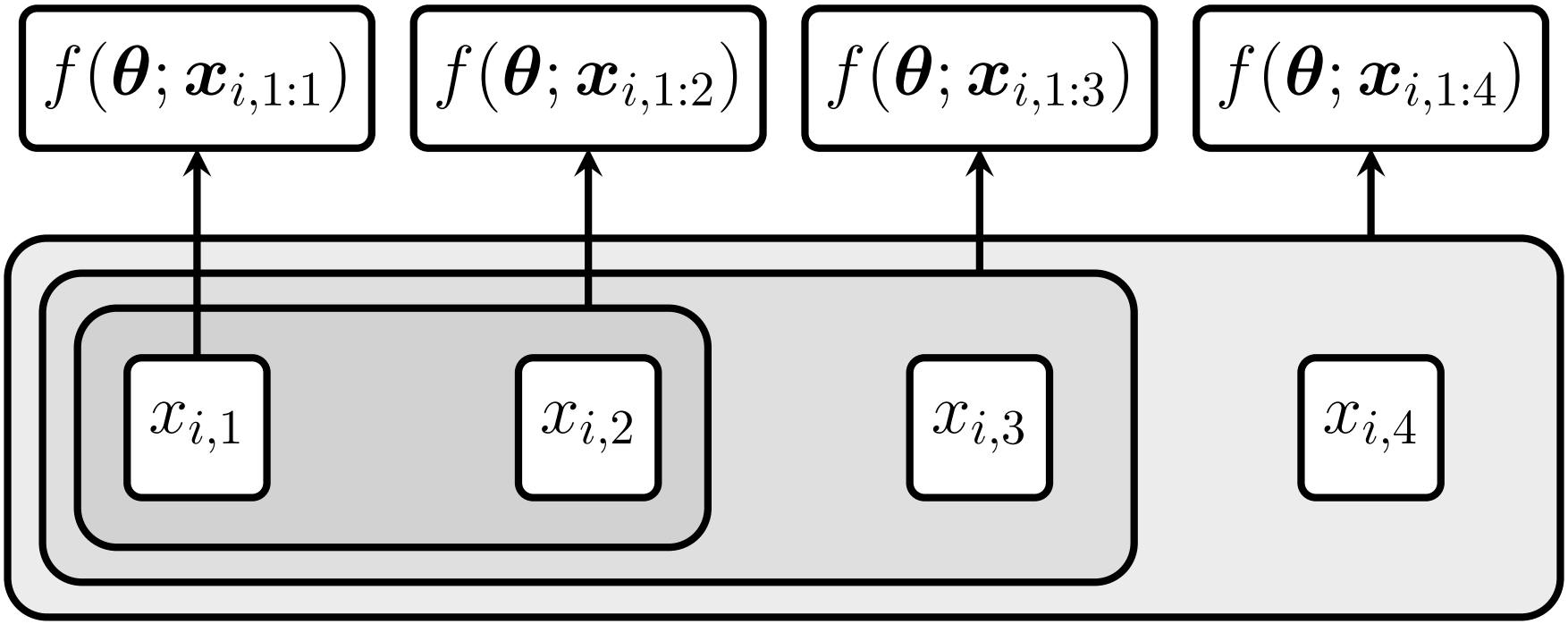

Figure: A sequence of next-token predictions

Autoregressive Settings VI

- But ensuring that operations used to calculate $f(\boldsymbol{\theta}; \boldsymbol{x}_{1:j-1})$ can also be for $f(\boldsymbol{\theta}; \boldsymbol{x}_{1:j})$ is not an easy task. Subsequently we introduce an auto-regressive setting for our purpose.

Summary I

- What we have shown is a high-level illustration of LLM.
- The remaining task is to design a suitable function f.