LLM Finetuning

Finetuning is also known as transfer learning where we retrain a pretrained model on the specific usecase like: private company data

Finetuning

- Full finetuning
- Parameter efficient Fine tuning (PEFT)

~~Why finetuning~~?

Finetuning vs RAG

RAG provides knowledge whereas finetuning can embed this knowledge with response style to the model.

Rag: Easy to update knowledge.
Low cost

Finetune: Handles emotions / slang
Answers in brand tone.

A real time LLM finetuning example:

Llama-3.2-1B is a pretrained model whose foundation is auto complete only.

Llama-3.2-1B-Instruct is a finetuned version which can behave like chatgpt.

Full fine tuning:

In this we train all the weights of a model. for eg if a model has 70Billion parameter, we would train all the parameters again.

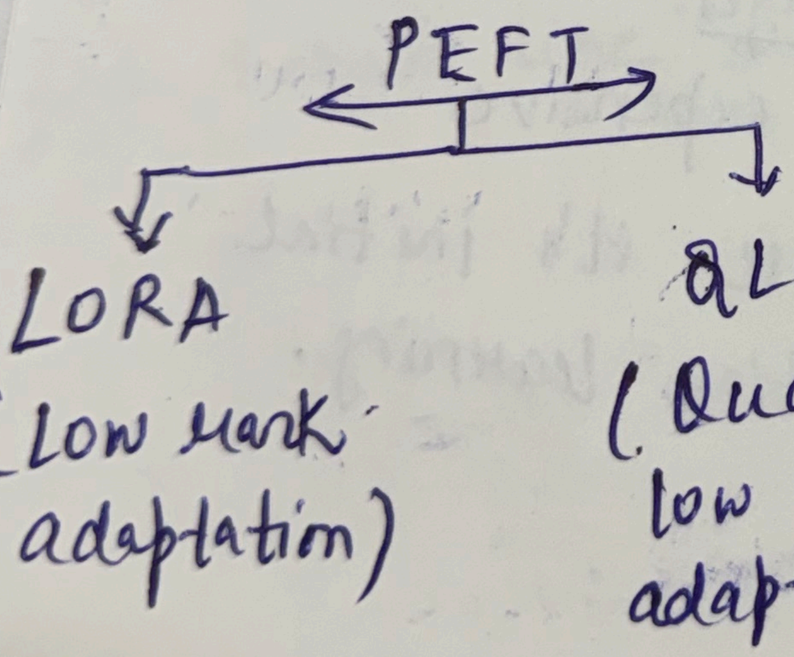
Disadvantages:

1) very expensive
2) can lose it's initial foundation learning.

PEFT

(parameter efficient Finetuning)

Here, we don't train all the layers(weights) we freeze them & only train few of them or add some new weights.



LORA

In LORA, we freeze all the pretrained weights and add a new weight & update the final weight.

$$W' = W + \Delta W$$

W = frozen old weights

ΔW = Delta weight, the new computed weights

W' = new weight calculated.

In peft, the whole model is frozen but a small set of trainable parameter is added to the model.

Question now: How is this memory efficient? Because Here also if we update 70B parameters then it's same consumption.

How LoRA works

LoRA breaks the ΔW bigger matrices into smaller subset of dot product matrices.

eg: $\Delta W = 5 \times 5$ matrix

LoRA makes it into

$(2 \times 5) \cdot (5 \times 2)$

And their dot product would give (5×5) matrix only but we would not update (5×5) as ΔW

instead, we would train (5×2) & (2×5) separately

Now, 5×2 = 10 parameters

2×5 = 10 parameters

Summing them would give only 20 parameters which is less than 25 parameter of (5×5) ΔW matrix.

This is small difference here but imagine

$$\underset{(d\times d)}{\Delta W} = \underset{(d\times r)}{A} \cdot \underset{(r\times d)}{B}$$

Where d = 512 so

d×d = 512 × 512 = 262144

Whereas

A(d×r) = 512 × 8 = 4096

B(r×d) = 8 × 512 = 4096

A+B = 8192 only

Here, we replaced r with 8, Now this 'r' is called rank in (LORA).

This 'r' is a hyperparameter which ranges from 2, 4, 6, 8, 10

For model size

< 1B params

r = 4, 8, 16

1B - 7B params

r = 8, 16, 32

> 13B params

r = 16, 32, 64

> 70B params

r = 64, 128 etc.

QLORA

Before starting QLORA, let's understand 'Quantization'.

Quantization is a process of converting high precision numbers like float32 into lower precision formats like int8 to reduce memory & computation in ML models.

Let's take an example:

Float32 list range = -1 to 1

$$\begin{bmatrix} -0.91 \\ -0.78 \\ -0.39 \\ +0.22 \\ 0.28 \\ 0.87 \end{bmatrix}$$

Now let's Quantize it into int2.

As we know int2 will have range of -2 to 1 and can store 4 values only because 2^n formula

so $2^2 = 4$

so we can make 4 buckets of $[-2, -1, 0, 1]$

Now, we would simply map float32 list to int2 list as

$[-2, -2, -1, 0, 0, 1]$

Question is how we Quantized?

$x = [-1, 1]$ float32

$q = [-128, 127]$ int8

$$\text{Scale} = \frac{x_{max} - x_{min}}{q_{max} - q_{min}}$$

$$q = \text{round}\left(\frac{x - \text{centre\_point of } x)}{\text{Scale}}\right)$$

$$\text{Scale} = \frac{1 - (-1)}{128 - (-127)} = \frac{2}{255}$$

$$= 0.0078$$

Now for a value of float 32,

-0.91

$$q = \text{round}\left(\frac{-0.91 - 0}{0.0078}\right)$$

$q = -116$

Here 0 was the centre point of the float32 list.

So for -0.91 can be quantized to -116 in int8

Indirectly, we are mapping to the nearest value.

Now, QLORA

So it combines 4 bit Quantization (compressing model weights) with LORA (training smaller adapter matrices).

Qlora uses a concept of <u>double quantization</u>.

In this let's say a model of 7b parameters needed to be quantized then

you create a block of 64 parameters, similarly creating 109 million blocks and Quantize them.

Now, each block will have different scale, so now Quantize these scale values again