Project Work I. - 2023/24/2

Parameter Efficient Fine-tuning of Large Language Models

Author: Sándor Zsombor

Supervisor: Csanády Bálint

What is parameter efficient fine-tuning?

What is PEFT?

Problem:

- Large language models have several billion parameters
- Full fine-tuning takes a lot of time and resources
- Prompt engineering is largely a manual process

PEFT:

- Purpose: achieve a performance similar to full fine-tuning, but only tuning a fraction of the original parameters
- General method: we freeze the parameters, add extra layers, which modify the output, and only tune the new layers

Used PEFT methods

Prompt tuning

- Takes inspiration from prompt engineering
- Prompt engineering: we try to „fix” the prompt to force the model in the direction of the wanted output
- Prompt tuning: do this, but after embedding
- We add virtual tokens, embed these, then concatenate them

Low-rank adaptation

- Large number of parameters in LLM layers, BUT the intrinsic rank is low
- Idea: fine-tune in a lower rank space
- Let $M \in \mathbb{R}^{n \times k}$ be a parameter matrix
- Add $\frac{\alpha}{r}AB$, where $A \in \mathbb{R}^{n \times r}$ and $B \in \mathbb{R}^{r \times k}$
- α and r are hyperparameters
- d dropout hyperparameter

Results

The experiment

IMDB dataset:

- Contains 50 000 movie reviews
- All labelled with `positive` or `negative` depending on the sentiment
- 50%-50% test-train split, balanced `positive`–`negative` ratio

Model:

- LLaMA2 family by Meta
- Experiments mainly with 7B model
- Final results also with 13B and 70B models

Accuracy without training (Stanford Alpaca type prompts):

Model size	7B	13B	70B
Accuracy	51.78%	82.564%	94.24%

7B results – Prompt tuning

- 2000 test samples used
- $N =$ train sample size
- $V =$ number of virtual tokens

N \ V	8	25	80
50	64.9%	61.2%	57.3%
150	86.25%	76%	83.55%
500	88.4%	89.75%	94.1%
2500	95.3%	95.55%	**96.15%**

7B results – LoRA

- 2000 test samples used
- N = train sample size
- d = dropout
- Experiments with $r = 8$, $\alpha = 16$

N \ d	0.0	0.1	0.2
50	91.05% (2)	90.4% (2)	92.15% (2)
150	95.4% (2)	94.1% (1)	95.3% (2)
500	95.3% (1)	96.05% (1)	96.35% (2)
2500	96.1% (1)	**96.85%** (2)	96.25% (2)

Maximizing performance

- Based on the 7B experiments, we tried to maximize accuracy on all models
- Used all train samples, ran for 2 epochs
- Used all test samples for evaluation
- LoRA, $r = 8$, $\alpha = 16$, $d = 0.1$, learning rate: $5 \cdot 10^{-4}$

Model size	7B	13B	70B
Accuracy	96.98%	97.32%	96.812%

- All models beat the previous best of 96.21%
- 70B preforms the worst, but this is expected, hyperparameter optimization is critical