LoRA (Low Rank Adaptation)

1. Specifically using in fine tuning of LLM

Let's breakdown LoRA

What is rank ?

https://www.youtube.com/watch?v=zksRGHYD76g

https://www.youtube.com/watch?v=cSj82GG6MX4

The minimum number of linearly independent rows in a matrix or equivalently the number of linearly independent columns in a matrix. rank is used in solving equations and analyzing data

What does low rank mean ?

Rank is smaller than the number of dimensions

provide a compact representation , thus reducing complexity.

Adaptation :

refers to the fine tuning process of models

Inspiration behind LoRA

INTRINSIC DIMENSIONALITY EXPLAINS THE EFFEC- TIVENESS OF LANGUAGE MODEL FINE-TUNING.pdf

Excerpt from the abstract is given below :

We empirically show that common pre-trained models have a very low intrinsic di- mension; in other words, there exists a low dimension reparameterization that is as effective for fine-tuning as the full parameter space. For example, by optimiz- ing only 200 trainable parameters randomly projected back into the full space, we can tune a RoBERTa model to achieve 90% of the full parameter performance levels on MRPC. Furthermore, we empirically show that pre-training implicitly minimizes intrinsic dimension and, perhaps surprisingly, larger models tend to have lower intrinsic dimension after a fixed number of pre-training updates, at least in part explaining their extreme effectiveness

Explained in simpler terms

Large foundation models can be tuned on a smaller parameters to achieve better performance , this is mainly because they are already trained on a larger set of features and are general purpose models.

Here's a simplified breakdown of the main points:

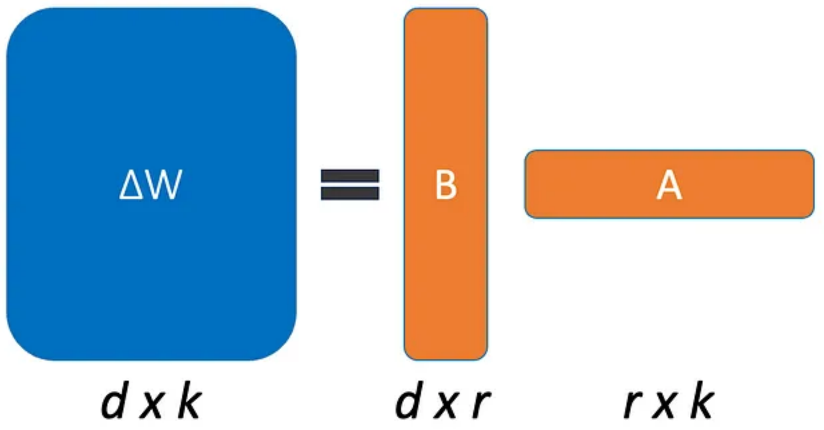
Low Intrinsic Dimension: Pre-trained models, despite having many parameters (settings that the model uses to make decisions), can be adjusted or "tuned" effectively by only changing a small subset of these parameters. This smaller set of parameters can still capture the main capabilities of the model.

Reparameterization Example: Imagine a model like RoBERTa, which has a vast number of parameters. The text suggests that by tweaking only about 200 of these parameters in a certain clever way (projecting them back into the full space of original parameters), the model can achieve almost the same performance as it would if you tweaked all the parameters. Specifically, it can reach 90% effectiveness just by adjusting this small number.

Pre-training Effect: When models are pre-trained (trained on a large amount of data before being fine-tuned for specific tasks), they tend to organize their parameters in a way that reduces the number of parameters that actually need to be tuned later. Surprisingly, the larger the model and the more training it initially receives, the fewer parameters need to be actively tweaked during fine-tuning to achieve high performance.

In essence, this means large, complex models are surprisingly efficient. Despite their complexity, you don’t need to adjust every single part of them to tailor them to a specific task; only a small part needs tweaking, which makes these models both powerful and efficient.

Mathematical intuition behind LoRA



So based on these observations , microsoft researchers proposed that the the change in model weights $deltaW$ also has low intrinsic dimensions. As explained before dimension is related to the rank of a matrix , therefore LoRA suggests to fine tune through a low rank matrix.

This can be explained mathematically as follows

$W_0 + deltaW = W_o +BA$

where ,

W_o : Is the weights of the original model which they didn’t touch

B : matrix of dimension $d* r$

A : matrix of dimension $r * k$

i.e both B and A are low rank matrices whose product is exactly the change in model weights ($delataW$)

for example the product of a $3 X3$ matrix can be denoted as the product of a $3X1$ matrix and $1X3$ matrix .

The implication isn’t that evident in the case of a 4 X 4 matrix . Consider a 200 X 200 matrix , it can be decomposed as a 200 X 2 matrix and 2 X 200 matrix , essentially what we will be doing is saving only 8000 parameters in memory instead of 40000 parameters.

mathematical equation in the forward pass will be something like this

$h = W_0x + deltaW x = W_0 x + BAx$

We then scale $ΔWx$ by $α / r$, where $α$ is a constant.

where ,

r : rank , this determines to what extent you want to decompose the matrices. typical values of r range from 1 to 64 . it expresses the amount of compression on the weights

$α$: constant

1. $α / r$ is like a scaling factor .amount of change that is added to the original model weights. therefore it balances the weights of the pretrained model and the adaption to a new task
2. This can be compared to how the learning rate controls the step size of parameter updates during optimization. Similarly, α controls the magnitude of updates to the weights in LoRA.

alpha and r are hyper parameters

when ,

α / r = 0 , original model

α / r = 1 , fully fine tuned model

α / r is done to reduce the number of weights updated , because with higher rank there will be more values .

Hyperparameters

Hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning. The prefix 'hyper_' suggests that they are 'top-level' parameters that control the learning process and the model parameters that result from it.

Relating α / r to learning rate

Learning Rate in Optimization Algorithms:

- In traditional optimization algorithms such as stochastic gradient descent (SGD) or its variants like Adam, the learning rate is a hyperparameter that controls the step size of parameter updates during training.
- The learning rate determines how much the model's parameters are adjusted in each iteration based on the gradient of the loss function with respect to those parameters.
- Tuning the learning rate is crucial for effective optimization, as a too large learning rate can cause the optimization process to diverge, while a too small learning rate can result in slow convergence.

α / r in LoRA:

- In LoRA, α / r serves a similar purpose to the learning rate in optimization algorithms, but with a specific focus on scaling the updates to the pre-trained weights during adaptation.
- α is a constant parameter that controls the overall magnitude of the updates, while r represents the rank of the low-rank decomposition used for weight updates.
- Dividing α by r ensures that the updates are scaled appropriately relative to the rank of the decomposition, maintaining consistent adaptation across different rank values.

For Example:

- Imagine you have a pre-trained weight matrix W0 with certain values.
- When adapting the model using LoRA with a rank of 4 (r=4), the updates to W0 (ΔW) are scaled by α/4.
- If α is set to 8, then the updates are scaled by 8/4 = 2.
- Regardless of the chosen rank, the updates are scaled consistently with the pre-trained weights, ensuring stable adaptation.

In summary, by setting α to a fixed value (e.g., 8) divided by the rank, LoRA ensures consistent scaling of updates relative to the pre-trained weights, promoting stable and effective adaptation of the model across different tasks and rank values.

why is scaling needed ?

this scaling helps stabilise other hyperparameters like learning rate.

What is the ideal rank ?

an important question to ask ourselves when choosing the rank is “did the base model already see similar data or is it substantially different ?”

for example if ur base model was trained in english and want it to be able to answer questions in french , then in that case u will have to use a very high rank , this would essentially mean u are performing full fine tuning.

Abstract

problem statement

are existing solutions good ?

LoRA.pdf

https://www.youtube.com/watch?v=DhRoTONcyZE

https://www.youtube.com/watch?v=X4VvO3G6_vw

References