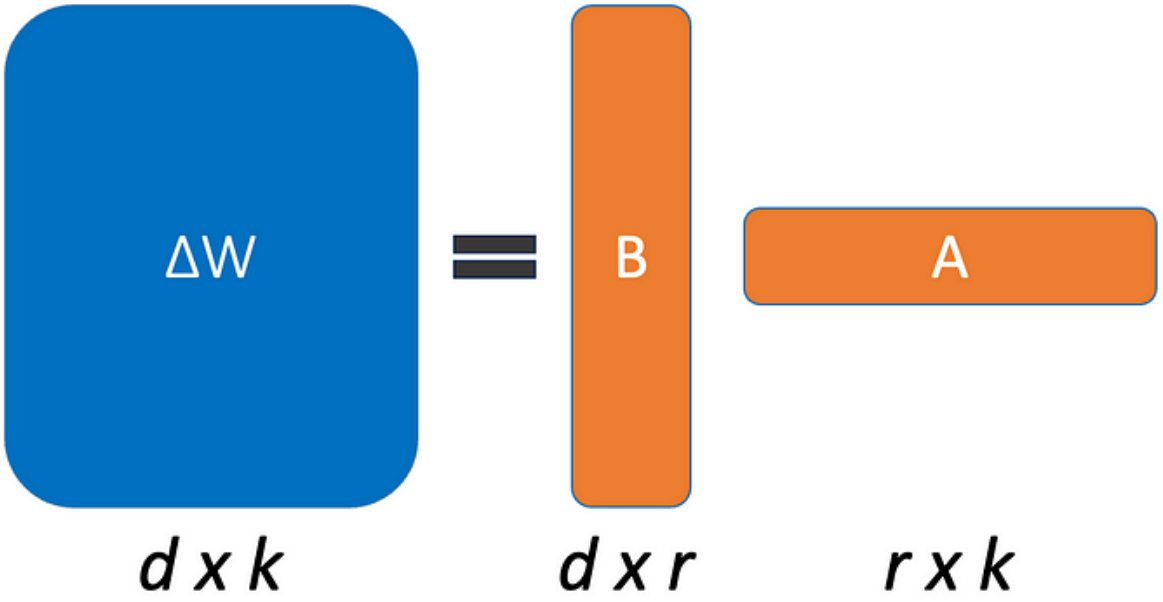

More about LoraConfig from PEFT

Parameter-Efficient Fine-Tuning (PEFT) enable efficient adaptation of pre-trained models to downstream applications without fine-tuning all…

Manyi

Follow

androidstudio · September 5, 2023 (Updated: September 5, 2023) · Free: Yes

Parameter-Efficient Fine-Tuning (PEFT) enable efficient adaptation of pre-trained models to downstream applications without fine-tuning all the model's parameters. PEFT supports the widely-used w-Rank Adaptation of Large Language Models (LoRA).

To create a LoRA model from a pretrained transformer model, we

```
Copy

from peft import LoraConfig
config = LoraConfig(
    r=8,
    lora_alpha=16,
    target_modules=["q", "v"],
    lora_dropout=0.01,
    bias="none"
    task_type="SEQ_2_SEQ_LM",
)
```

Let's go over the arguments in LoraConfig.

LoRA Dimension / Rank of Decomposition *r*

For each layer to be trained, the $d \times k$ weight update matrix $\mathbf{\Delta W}$ is represented by a low-rank decomposition $\mathbf{BA}$, where $\mathbf{B}$ is a $d \times r$ matrix and $\mathbf{A}$ is a $r \times k$ matrix. The rank of decomposition r is << $\min(d,k)$. The default of r is 8.

$\mathbf{A}$ is initialized by random Gaussian numbers so the initial weight updates have some variation to start with. $\mathbf{B}$ is initialized by by zero so $\mathbf{\Delta W}$ is zero at the beginning of training.

None

Alpha Parameter for LoRA Scaling *lora_alpha*

According to the LoRA article Hu et. al., $\mathbf{\Delta W}$ is scaled by α / r where α is a constant. When optimizing with Adam, tuning α is roughly the same as tuning the learning rate if the initialization was scaled appropriately. The reason is that the number of parameters

with the pretrained weights no matter what *r* is used. That's why the authors set *α* to the first *r* and do not tune it. The default of *α* is 8.

Modules to Apply LoRA to *target_modules*

You can select specific modules to fine-tune. According to https://github.com/microsoft/LoRA/blob/main/README.md, loralib only supports nn.Linear, nn.Embedding and nn.Conv2d. It is common practice to fine-tune linear layers. To find out what modules your model has, load the model with the transformers library in Python and then print(model). The default is *None*. If you want to fine-tune all linear layers, do this

Copy

```
import re
pattern = r'\((\w+)\): Linear'
linear_layers = re.findall(pattern, str(model.modules))
target_modules = list(set(linear_layers))
```

Dropout Probability for LoRA Layers *lora_dropout*

Dropout is a technique to reduce overfitting by randomly selecting neurons to ignore with a dropout probability during training. The contribution of those selected neurons to the activation of downstream neurons is temporally removed on the forward pass, and any weight updates are not applied to the neuron on the backward pass. The default of *lora_dropout* is 0.

`as Type for Lora *bias*

Bias can be 'none', 'all' or 'lora_only'. If 'all' or 'lora_only', the

as the base model would have without adaptation. The default is *None*.

Task Type *task_type*

It seems that everything works just fine without specifying task_type. Possible task types include CAUSAL_LM, FEATURE_EXTRACTION, QUESTION_ANS, SEQ_2_SEQ_LM, SEQ_CLS and TOKEN_CLS.

Other Arguments

The remaining arguments including fan_in_fan_out, modules_to_save, layers_to_transform and layers_pattern are less frequently used.

Reference

PEFT: https://pypi.org/project/peft/ LoRA: https://arxiv.org/abs/2106.09685 Microsoft LoRA: https://github.com/microsoft/LoRA/blob/main/README.md LoRA config: https://github.com/huggingface/peft/blob/main/src/peft/tuners/lora/config.py

#lora #peft #fine-tuning #transformers #llm