

机器之心课程前相关材料 - Python 快速入门

考虑到参加本次培训的同学有可能不熟悉 Python，准备了一些简单的资料，和当前版本对于 Python 最佳的学习方案，供参考。

写在前面

如果你有任何编程语言的经验，尤其是 Ruby、JavaScript、Lua 这类动态语言，那么你会 Python 感到一些熟悉。如果你没有任何上述相关语言经验，也无妨，就本教程而言，我们只需要会使用基础语法，流程控制即可，因为数据集准备，甚至是模型推理、API 调用都可以使用我们已熟悉的语言来进行处理。

简单的入门资料

通常情况下，我们使用的 Python 版本会是 3.8~3.10（尤其是使用 Nvidia 镜像、PyTorch 镜像时），如果你的耐心十分好，可以从 [Python 官方教程](#) 开始看起。

如果你希望例子更多一些，可以阅读 FreeCodeCamp 的 [Python 教程](#)。

当然，不论你是使用上面的教程，还是手头有一本合适的入门的书籍，我都建议你实际试一试，比如使用在线的 [Playground](#)。

如果你更喜欢本地的环境，我推荐你使用 Docker 或者 Conda 来完成日常的 Python 程序的练习和编写，上面两个工具都能够提供干净的、可复现的环境，相比我们在某一套环境里来回折腾（养蛊），能够减少非常多不必要的麻烦。

Docker 中的 Python 环境准备（可选）

想顺利的完成实践，推荐你安装 Docker，不论你的设备是否有显卡，都可以根据自己的操作系统喜好，参考这两篇来完成基础环境的配置
《[基于 Docker 的深度学习环境：Windows 篇](#)》、《[基于 Docker 的深度学习环境：入门篇](#)》。当然，使用 Docker 之后，你还可以做很多事情，比如：之前[几十篇有关 Docker 的实践](#)，在此就不赘述啦。

当你完成各种操作系统中的 Docker 的安装后，就可以使用 Docker 来快速下载一个合适的 Python 运行环境啦。

参考《[节省时间：AI 模型靠谱下载方案汇总](#)》这篇文章中的内容，我们可以快速的初始化一个 Python 3.10 的环境：

```
# 下载一个只有 50MB 的 Python 轻量环境
docker pull python:3.10-slim
# 将本地目录挂载到容器里，你可以将 Docker 中折腾的内容保存到本地
docker run --rm -it -v `pwd`:/code python:3.10-slim bash

# 进入容器后，如果你觉得下载软件包什么的比较慢，可以设置一个软件源
sed -i 's/snapshot.debian.org/mirrors.tuna.tsinghua.edu.cn/g' /etc/apt/sources.list.d/debian.sources
pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple

# 进入你的目录
cd /code

# 接下来就可以随便学习（折腾）啦
python ... # (你的程序)，或者直接进入交互式终端 Coding
```

使用 Conda 来准备 Python 环境（可选）

如果你不喜欢，或者你的本地设备不能够安装 Docker，那么你也可以选择 Conda 来完成你的 Python 环境的准备。

你可以参考《[在搭载 M1 及 M2 芯片 MacBook 设备上玩 Stable Diffusion 模型](#)》这篇文章，从官方的[归档页面](#)，下载合适你的设备的安装包。

然后，执行安装文件，完成 Conda 的安装，一般情况，“一路 Next”即可。

在使用 Conda 时，先进行软件源配置操作。这样可以减少在下载软件包过程中造成的不必要时间浪费。使用 `vi ~/.condarc` 编辑 Conda 配置文件，在其中加入下面的内容（以“清华源”为例）：

```
channels:
```

```
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch/
- defaults
show_channel_urls: true
```

然后，我们就可以轻松的创建一个 Python 3.10 的环境，来运行我们的代码：

```
# 创建一个环境
conda create -n jiqizhixin python=3.10 -y
# 启用一个环境
conda activate jiqizhixin
```

更简单的学习方法

现在是模型时代，大家人肉若干的模型、几乎人人都有 GitHub Copilot，所以其实没有必要完全像上个时代一样苦学，借助模型，这个学习过程可以非常简单。

如果你 VSCode Copilot 的使用权限，我们可以直接在程序中问 Copilot “这段代码什么意思”。

peft

资源管理器

打开的编辑器

PEFT

src/peft

peft_model.py

9+

src > peft > peft_model.py > PeftModel

```
104     class PeftModel(PushToHubMixin, torch.nn.Module):  
130     ....  
131  
132     def __init__(self, model: PreTrainedModel, peft_config: PeftConfig,  
133                  adapter_name: str = "default",  
134                  autocast_adapter_dtype: bool = True,  
135                  ) -> None:  
136         super().__init__()  
137         self.modules_to_save = None  
138         self.active_adapter = adapter_name  
139         self.peft_type = peft_config.peft_type  
140         # These args are special PEFT arguments that users can pass. They need to be re-  
141         # forward.  
142         self.special_args = None  
143  
144         self._is_prompt_learning = False  
145         if self._is_prompt_learning:  
146             self._forward_fn = self._forward_fn_for_prompt_learning  
147             self._backward_fn = self._backward_fn_for_prompt_learning  
148             self._backward_fn_for_backward = self._backward_fn_for_backward_for_prompt_learning  
149             self._backward_fn_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward  
150             self._backward_fn_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward  
151             self._backward_fn_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward  
152             self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward  
153             self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward  
154             self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward  
155             self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward  
156             self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward  
157             self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward  
158             if hasattr(model, "backward_for_backward"):  
159                 self._backward_fn_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward  
160             else:  
161                 self._backward_fn_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward  
162             self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward  
163             if hasattr(model, "backward"):  
164                 self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward = self._backward_fn_for_backward_for_backward_for_backward_for_backward_for_backward  
165             else:
```

分析选中文本中的单词

Compile Files

转到定义

转到声明

转到类型定义

转到引用

快速查看

查找所有引用

显示调用层次结构

显示类型层次结构

Copilot

重命名符号

更改所有匹配项

重构...

源代码操作...

Open Changes

Open on Remote (Web)

File Annotations

File History

共享

剪切

复制

复制为

粘贴

将所选内容添加到聊天

将文件添加到聊天

在编辑器中启动

对此进行解释

修复此

生成文档

生成测试

main Launchpad 0 △ 11 0 125.92 KiB 自动分析单词 行 13 / 4 UTF-8 LF Python 3.11.7 64-bit Prettier

程序会自动开始解释你好奇的，或者调整你想调整的内容。

聊天

```
self._peft_config = None
cls = PEFT_TYPE_TO_MODEL_MAPPING[peft_config.peft_type]
self.base_model = cls(model, {adapter_name: peft_config}, adapter_name)
self.set_additional_trainable_modules(peft_config, adapter_name)

if hasattr(self.base_model, "_cast_adapter_dtype"):
    self.base_model._cast_adapter_dtype(
        adapter_name=adapter_name, autocast_adapter_dtype=autocast_adapter_dtype
)
```

</> Workspace
> 已使用 5 个引用

这段代码是一个Python类的构造函数，用于初始化一个模型，该模型可以通过添加适配器（adapter）来扩展预训练模型的功能。这个构造函数接受四个参数：`model`（一个预训练模型），`peft_config`（配置信息），`adapter_name`（适配器的名称，默认为"default"），以及`autocast_adapter_dtype`（一个布尔值，用于决定是否自动转换适配器的数据类型，默认为True）。

首先，通过`super().__init__()`调用父类的构造函数来进行初始化。然后，初始化一些属性，包括`modules_to_save`（用于保存模型中需要额外训练的模块），`active_adapter`（当前激活的适配器名称），`peft_type`（从配置中获取的PEFT类型），以及`special_peft_forward_args`（一个集合，用于存储在前向传播时需要特别处理的参数）。

接下来，根据`peft_config`中的`is_prompt_learning`属性判断是否为提示学习（prompt learning）模式。如果是，那么将配置信息存储在`_peft_config`中，并将传入的`model`设置为基础模型（`base_model`），然后调用`add_adapter`方法添加一个适配器。如果不是提示学习模式，`_peft_config`被设置为None，通过`PEFT_TYPE_TO_MODEL_MAPPING`字典根据PEFT类型找到对应的模型类，然后使用这个类创建一个新的基础模型实例，并调用`set_additional_trainable_modules`方法设置额外的可训练模块。

最后，检查基础模型是否有`_cast_adapter_dtype`方法，如果有，则调用这个方法来根据`autocast_adapter_dtype`参数决定是否自动转换适配器的数据类型。

这个构造函数的设计体现了模块化和可扩展性，通过适配器机制允许用户根据需要扩展预训练模型的功能，同时也提供了灵活的配置选项来适应不同的训练需求。

这个构造函数中的`add_adapter`方法是做什么用的？

询问 Copilot 或键入 / 以获取命令

```
src > peft > Python 3.11.7 64-bit > Prettier
```

行 151, 列 1 (已选择36) 空格: 4 UTF-8 LF

```
104     class PeftModel(PushToHubMixin):
105         """
106
107         def __init__(
108             self,
109             model: PreTrainedModel,
110             peft_config: PeftConfig,
111             adapter_name: str = "default",
112             autocast_adapter_dtype: bool = True
113         ) -> None:
114             super().__init__()
115             self.modules_to_save = []
116             self.active_adapter = None
117             self.peft_type = peft_config.peft_type
118             # These args are specific to forward pass.
119             self.special_peft_forward_args = set()
120
121             self._is_prompt_learning = False
122             if self._is_prompt_learning:
123                 self._peft_config = peft_config
124                 self.base_model = model
125                 self.add_adapter(adapter_name)
126             else:
127                 self._peft_config = None
128                 cls = PEFT_TYPE_TO_MODEL_MAPPING[peft_config.peft_type]
129                 self.base_model = cls(model, {adapter_name: peft_config}, adapter_name)
130
131             if hasattr(self.base_model, "_cast_adapter_dtype"):
132                 self.base_model._cast_adapter_dtype(
133                     adapter_name=adapter_name, autocast_adapter_dtype=autocast_adapter_dtype
134                 )
135
136             if getattr(model, "is_torchscript_compatible", False):
137                 self._is_torchscript_compatible = True
138
139             self.set_additional_trainable_modules(peft_config, adapter_name)
140
141             self._is_torchscript_compatible = False
142
143             self._is_torchscript_compatible = True
144
145             self._is_torchscript_compatible = False
146
147             self._is_torchscript_compatible = True
148
149             self._is_torchscript_compatible = False
150
151             self._is_torchscript_compatible = True
152
153             self._is_torchscript_compatible = False
154
155             self._is_torchscript_compatible = True
156
157             self._is_torchscript_compatible = False
158
159             self._is_torchscript_compatible = True
160
161             self._is_torchscript_compatible = False
162
163             self._is_torchscript_compatible = True
164
165             self._is_torchscript_compatible = False
```

当然，除了解释代码含义，我们还可以将代码发送到聊天内容中，询问代码改进的方法、添加代码的测试代码、修正跑不动的代码等等。

如果说你没有 Copilot 或者非常讨厌使用 VSCode，那么你也可以使用各种 ChatBot 产品，比如准确率非常高的 Claude 网页版（使用 AWS Bedrock Claude 是一样的，[可参见文章，开通 Claude](#)），来完成上面的“傻瓜式学习”。

比如，我将使用下面的 Prompt，来让模型干活，给我们解释下面代码的含义：

你是 Google 首席 Python 工程师，擅长编写 Python 代码，始终准确回答用户的问题，对下面的每一行代码进行详细的解释。

```
`python
from peft import AutoPeftModelForCausalLM
from transformers import AutoTokenizer
import torch

model = AutoPeftModelForCausalLM.from_pretrained("ybelkada/opt-350m-lora")
tokenizer = AutoTokenizer.from_pretrained("facebook/opt-350m")

model = model.to("cuda")
model.eval()
inputs = tokenizer("Preheat the oven to 350 degrees and place the cookie dough", return_tensors="pt")

outputs = model.generate(input_ids=inputs["input_ids"].to("cuda"), max_new_tokens=50)
print(tokenizer.batch_decode(outputs.detach().cpu().numpy(), skip_special_tokens=True)[0])

"Preheat the oven to 350 degrees and place the cookie dough in the center of the oven. In a large bowl, combine the
flour, baking powder, baking soda, salt, and cinnamon. In a separate bowl, combine the egg yolks, sugar, and vanilla."
`
```

你是 Google 首席 Python 工程师，擅长编写 Python 代码，始终准确回答用户的问题，对下面的每一行代码进行详细的解释。



```
from peft import AutoPeftModelForCausalLM
from transformers import AutoTokenizer
import torch

model = AutoPeftModelForCausalLM.from_pretrained("ybelkada/opt-350m-lora")
tokenizer = AutoTokenizer.from_pretrained("facebook/opt-350m")

model = model.to("cuda")
model.eval()
inputs = tokenizer("Preheat the oven to 350 degrees and place the cookie dough", return_tensors="pt")

outputs = model.generate(input_ids=inputs["input_ids"].to("cuda"),
```

Claude 3.5 Sonnet ▾

Claude currently cannot access the internet or reference links

Add content

Use a project ▾

不出意外，你会得到详尽的回答，这应该能够帮助你更快速的入门，和解决你 7x24 的好奇心。

Preheat the oven to 350 degrees and place the cookie dough in the center of the

好的,我将为您详细解释这段Python代码的每一行:

```
python
```

Copy

```
from peft import AutoPeftModelForCausalLM
```

这行代码从peft库中导入AutoPeftModelForCausalLM类。PEFT代表Parameter-Efficient Fine-Tuning,这是一种用于高效微调大型语言模型的技术。

```
python
```

Copy

```
from transformers import AutoTokenizer
```

从transformers库中导入AutoTokenizer类,这个类用于自动加载适合特定模型的分词器。

```
python
```

Copy

```
import torch
```

导入PyTorch库,这是一个流行的深度学习框架。

```
python
```

Copy

```
model = AutoPeftModelForCausalLM.from_pretrained("ybelkada/opt-350m-lora")
```

加载一个预训练的PEFT模型,这个模型是基于OPT-350M模型使用LoRA (Low-Rank Adaptation) 技术微调的。

```
python
```

Copy

```
tokenizer = AutoTokenizer.from_pretrained("facebook/opt-350m")
```

加载与OPT-350M模型匹配的分词器。

python

Copy

```
model = model.to("cuda")
```

将模型移动到CUDA设备(GPU)上以加速计算。

python

Copy

```
model.eval()
```

将模型设置为评估模式,这会禁用一些训练特定的行为,如dropout。

举一反三，你可以向 AI 问“这段代码使用 Python 怎么写”、“完善这段代码，实现一个如此这般的功能”，来完成简单的函数的编写，来节约时间，更快达成目标。

周末愉快。

--EOF