Table of Contents

NJU HEP

Contact: bowen.zhang@cern.ch, or wechat: chambowen

(data) (knowledge) 列"迭代
(learning/training) 到 某种意义上的映&#x5C04
(function/trained
model) 到的映射关 代替人进&#x8

如何

很多...
视频课程 博客 教程 等等..
总有 个适合你的 方法。&#x6BD4
吴恩达等 中文 李宏毅等。
python 程序设 基础。

需要 到 程度

个人感觉对于时间精力&#x6709

在高能物理有哪 应用

自古以来从基础设施到&#x7269
自由搜索。

框架

ROOT TMVA

上 《粒子物理中实验方法
TMVA
了 讲义里有详细介绍 这&#x91C

其他在高能物理中&#

传统统 类似于 TMVA scikit-learn, xgboost

深度 深度神经网络 支持
GPU 加速 Keras, tensorflow, pytorch

使用Ntuple作为 样&#x672C

如何实践到高能物理的环

我设 了 个小例子 背景
HH->bbtautau

物理 析中对本底和信号进&

如何搭建 运行环境

如果你想在自己的电脑上&
Anaconda
管理程序包。如果想快捷&
(jupyter
notebook形式)。如果登录高能所&#x
singularity 获得 ml-base
镜像 里面预装了这个练 &#x4E2D

singularity shell -e docker://atlasml/ml-base:latest

如果有 ATLAS 账号 可以参考
ATLAS machine learning forum
获得软件硬件 源。

在这个例子中 我们使用熟&
IHEP 环境

第 部 的代码和 可以在这&#x91C
pth_bbtt_minimal。可以 readme 和 train.py
代码中注解练 。

目前读取 ROOT 文件的最佳方式

个人觉得最好用的 uproot
这个包 它可以轻松地把 &#x4E2A
ROOT 文件里面的 TTree 以及 TTree
里面的 TBranch 转换成 NumPy array pandas DataFrame
等格式 方便在任何 框架&#x91

目前有两个版本 3.x 和
4.x。例子中使用的 3.x
版本 新版本用户体验变化&#
ml-base 镜像里默认安装的 3.x。

在例子中 可以在 nn_inputs.py
里找到简单地用法 GitHub
网页上有更多进 步使用的&#

有了 ROOT
文件和转换方式 现在我们&#
pytorch tensor (Numpy array 转换而来) 作为
input pytorch
作为 框架训练神经网络 &
(不变质量和角度关)
和输出 (信号本底 类)
的映射关 。可以在 train.py
中看到神经网络的关键元&#
model, loss function, optimizer, learning rate, ..
以及 的基本 念 如训练/测&

使用Ntuple作为 样本的 般工&#x4F50

(ROC 曲线) ..

如果前面的设置没有 题的&#
train.py
应该能看到神经网络训练&

基于 lwtnn 的 inference

在训练好 个模型之后 需&#x8981
ROOT C++
的软件环境中应用 这个 &#x7A0B
inference testing。

目前 ATLAS 最常用的方法 ML forum
开发的 lwtnn 只支持用 Keras
训练和保存的模型。使用&
tau decay mode classification: keras training, inference with lwtnn。

但未来为了支持更多平台 &#
Onnx Runtime。

基于 onnxruntime 的 inference

继续前面的练 现在你的
output
文件夹应该已经保存了训&
onnx
格式的模型 第二部 的代码&
onnx_cpp_mininal。可以 readme 和 main.cpp
代码中注解练 。

此例简单包装了onnx
API 容易扩展到实际的 ROOT
析情形中。注意按照 readme
进行编译和运行（由于直&
onnxruntime 的预编译版本 需要 include
它的头文件并且保证链接&#

专题

THOR/loki 的

THOR 用来在 xAOD 上重新运行 tau
重建 法并转为轻量级的
M(Mini-)xAOD 的工具。

loki 可用 MxAOD
进行画较为标准的 tau
重建性能图以及调用 TMVA
进行多变量 析 主要用到
BDT。

更详细的介绍可以参考我&#
报告 以及他们的文档 THOR, loki。如果想试 下的话建议&#x
教程。

gitlab 链接在他们的名字里..

Tau Decay Mode Classification 的

安装特殊版本 THOR / loki

由于有 新功能（如直接产&
NNDecayMode branch 步骤如下

1. download and install THOR / tauRecToolsDev

```
git clone -b NNDecayMode ssh://git@gitlab.cern.ch:7999/zhangb/THOR.git THOR

cd THOR

git clone -b NNDecayMode ssh://git@gitlab.cern.ch:7999/zhangb/tauRecToolsDev.git
tauRecToolsDev

source setup.sh
```

wait for a few minutes for building the executables

2. download a testing sample

suppose you have set up rucio

```
rucio get --nrandom 1
valid1.425200.Pythia8EvtGen_A14NNPDF23LO_Gammatautau_MassWeight.recon.AOD.e5468_s3674_r12946
```

3. run a test job (500 events)

suppose the valid1.425200.Pythia8EvtGen_A14NNPDF23LO_Gammatautau_MassWeight.recon.AOD.e5468_s3674_r12946 folder is under /my/path

```
thor StreamNNDecayMode /Main.py -n 500 -dt TAU -i /my/path -o local_test
```

check what you got in local_test. MxAOD preserve the xAOD structure but it's per event, not ideal for training. Ntuple is per tau, more straightforward for training.

4. recommanded grid job submission command

```
# In R22Gammatautau.txt:

# SUBSTREAM = Gammatautau_PreProdv3 # <- you can change this to your preferred name

#
valid1.425200.Pythia8EvtGen_A14NNPDF23LO_Gammatautau_MassWeight.recon.AOD.e5468_s3674_r12946
# <- this is the lastest (until 2021.10.22) R22 preproduction sample

thor StreamNNDecayMode /Main.py -dt TAU -r grid -g R22Gammatautau.txt --gridstreamname
NNDecayMode --gridrunversion 22-04 --nFilesPerJob 4 # <- for more see thor --help and the
```

```
website documentation
```

5. once you have the samples, then you can use the ntuple output stream in training. my lastest ntuples are:

user.zhangb.NNDecayMode_Gammatautau_PreProdv3.425200.Pythia8EvtGen_A14NNPDF23LO_Gammatautau_Ma

they can be reached by 'rucio', try to download to your workspace

The Tau tracks, Conversion tracks, Neutral pfos, shot pfos variables are the inputs, and the true decay modes (encoded in 0->1p0n, 1->1p1n, 2->1pXn, 3->3p0n, 4->3pXn) is the target, in the terminology of supervised learning.

6. my scripts to train the Deepset network can be downloaded by:

```
git clone --recursive git@githubNOSPAMPLEASE.com:peppapiggyme/NNAlgs.git # <- it also
downloads the lwtnn package which will be used for converting the model
```

the code is not very user friendly, but i'll try to explain the procedure

my favourate working environment is https://www.atlas-ml.org/, you can sign up an account and aquire resources, but the code is supposed to be working on any platform with conda

7. set the corresponding information in "config/DatasetBuilder.py":

```
self._dataset.paths = walk_dir("/data/zhangb/r22-04/", "tree") # <- change to your path to
ntuples
```

the structure is:

.
└── r22-04
└──
user.zhangb.NNDecayMode_Gammatautau_PreProdv3.425200.Pythia8EvtGen_A14NNPDF23LO_Gammatautau_Ma
├── user.zhangb.27084970._000001.tree.root
├── ...

unfortunately, the setting for length is rigid..

```
self._dataset.length = 3264114 * 5 # <- for now don't worry, later when the size of your
input is changed, this must be changed accordingly, you can read the size from the log
```

recommanded batch size

```
self._dataset.batch_size = {"Train": 1000, "Validation": 100000, "Test": 100000}
```

8. other setttings in:

config/ConfDSNN-R22.yaml # <- general job settings and global hyperparameters

config/ArchDSNN-R22.yaml # <- the hyperparameters of the network architecture

the model it self is defined in nnalgs/algs/Models.py, the functional model is ModelDSNN

9. train and presist the NN model

the workflow:

1) generate LMDB file that holds all the numpy arrays (this is a single large file ~20GB for training), later the data loading is managed by a DataGenerator, so we don't need to load all data into RAM. The LMDBs are saved in data/lmdb/decaymode/.

2) at the same time (or before), generate pre-processing descriptions of the input variables (normalisation of input variables). The pre-process file is data/json/decaymode/variables.json. Later this file is useful for model convertion with lwtnn.

3) From outside NNAlgs, do:

```
python NNAlgs Train NNAlgs/config/ConfDSNN-R22.yaml
```

to train the model. This will take long time (1-2 days) because our dataset is huge. But it depends on the batch size of training and the power/condition of your machine.

4) When the training is finished, the log will tell you the best (smallest validation loss) model

> Minimum val_loss 0.5460284352302551 at epoch 55

so then you can convert this model using the simple script

```
source save_weights_for_lwtnn.sh baseline_dsnn_r22 55 # <- the 1st argument is the path to your saved folder, the second argument is the trained weights(epoch) you want to save to your model (i.e. the best one)
```

After this step succeeded, a new file named saved/baseline_dsnn_r22/weights-for-lwtnn.json is created. This file is useful to apply the model in C++ code (THOR/tau reconstruction) later

10. Apply the model back to THOR to produce the for performance evaluation.

At this time, you can evaluate the performance in python level if you want. It's fine and good, but I want to show you the standard way.

1） copy your model to THOR/data/ and go to build directory. Then do

```
cmake ../THOR && make -j 4
```

2) This time we'll use the MxAOD output stream produced by thor and we'll modify the job option file THOR/share/StreamNNDecayMode/Main.py

uncomment the classifier and comment the tree tool because we don't produce ntuples this time.

```
# # Decorator for decaymode classification variables

# tool = ROOT.tauRecToolsDev.NNDecayModeTree("NNDecayModeTree")

# Config + tool=

# Application of the classifier (to be added)

# MxAOD ->application and loki plotting (no )
```

```
# FlatTree ->for training, etc

tool = ROOT.tauRecToolsDev.NNDecayModeClassifier("NNDecayModeClassifier")

CHECK(tool.setProperty("OutputName", "NNDecayModeR22"))

CHECK(tool.setProperty("ProbPrefix", "NNDecayModeR22Prob_"))

CHECK(tool.setProperty("WeightFile", "THOR/weights-for-lwtnn.json"))

Config + tool=
```

save this and run thor to produce the MxAOD again. Check what you have in the MxAOD output. You should see the NNDecayModeR22 decay mode types of taus.

11. then one can put the in loki to make the performance plots.

1) install loki

```
git clone --recursive -b NNDecayMode ssh://git@gitlab.cern.ch:7999/zhangb/loki.git loki

cd loki

source setup.sh

cd examples
```

change the path to your samples:

```
data_path_Gtt = '/publicfs/atlas/atlasnew/higgs/hh2X/zhangbw/MxAOD/r22-03-App'

python NNDecayModePlots.py # <- i have defined several plots here

# note that careful cut is made to these samples so that the events that are used for the
plots (i.e. testing) is orthogonal to the events that were used for training.
```

12. finally, after waiting for a few minutes, the plots should be produced.

most representive one is the efficiency matrix:

Tau Energy Scale (TES) 的

- Tau performance at the start of Run2: ATL-PHYS-PUB-2015-045 <- 最低层次的 hadronic tau 能量刻度
- Tau performance and measurements with early Run2 data: ATLAS-CONF-2017-029 <- 这里有基于 Boosted Regression Tree (BRT) 的 TES 的较详细 。

关于 Tau 重建和鉴别的其他

- (Internal) Tau combined performance working group twiki
- More tau public results <- 可以在年度的 TauWG Workshops

里找到很多有用的信息

-- BowenZhang - 2021-09-29

This topic: Sandbox > NanjingML
Topic revision: r3 - 2021-10-22 - BowenZhang