Normalizing Flows

4/13/2025

3 Möjliga Poäng

Försök 1

4/8/2025
NÄSTA: Feedback från granskning

Poäng för försök 1:
Ej tillämpbart

Lägg till kommentar

Obegränsat antal försök tillåts
4/14/2025

Information

Your task is to implement a neural network that predicts the temperature, gravity, and metallicity of stars **AND the uncertainties of your predictions using Normalizing Flows** using the negative log-likelihood loss function. Use the *astronomy dataset, which* consists of light spectra that you used in the previous exercise. See the previous exercise **Astronomy CNN with PyTorch (https://uppsala.instructure.com/courses/102453/assignments/315724)** on how to obtain and normalize the data.

1. Implement the following three networks

The three following networks can all be implemented using different arguments and options from the **jammy_flows (https://github.com/thoglu/jammy_flows)** library.

Hint: Install Jammy Flows with `pip install git+https://github.com/thoglu/jammy_flows.git (https://github.com/thoglu/jammy_flows.git) --no-deps` *to avoid any installation problems with additional dependencies you don't need for this exercise.*

- a Normalizing Flow consisting of a *diagonal Gaussian*, i.e., the free parameters of the normalizing flow are the mean and standard deviation of each of the three labels. This is equivalent to the previous exercise, but the mean and standard deviation is implemented by using an affine normalizing flow through the Jammy Flows package.
- a Normalizing Flow consisting of a full 3-D Gaussian, i.e., the free parameters are the means of the three labels and the full covariance matrix between the three labels.
- a Normalizing Flow consisting of several *Gaussianization Flows* plus one *Affine Flow* to predict arbitrary uncertainty PDFs.

2. Quantify how well you were able to determine the uncertainties.

3. Visualize a few predicted PDFs from the Gaussianization Flow.

Implementing a Normalizing Flow model with Jammy Flows can be technically challenging. Therefore, we provide a template (**B03train_normalizing_flow_template.py (https://uppsala.instructure.com/courses/102453/files/8331928?wrap=1)** **(https://uppsala.instructure.com/courses/102453/files/8331928/download?download_frd=1)**) that implements the model definition, the three flows you are supposed to use, and the loss function. If you use this code, please make sure that you understand it.

Remember always to include:

- A **written summary** (0.5–1 A4 page) covering (submitted either as PDF or directly as text):
  - What you did and how
  - What results you obtained
  - What challenges you encountered and what could be improved
- A **PDF (or similar format)** with all **result plots**, each with a short explanation
- Your **code**, preferably as a link (e.g., GitHub, Google Colab, etc.) so we can view it easily.

	Filnamn	Storlek	
	stellar_prediction.py	32,1 kB	

	Filnamn	Storlek	
	Exercise_...w (3).pdf	1,29 MB	

```
import time
import sys
import os
import argparse
import glob
import subprocess
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset, random_split
from matplotlib import pyplot as plt
import jammy_flows
from scipy.stats import norm
from sklearn.preprocessing import MinMaxScaler
from torch.utils.data import Dataset, DataLoader, random_split
import pylab
from sklearn.preprocessing import StandardScaler
import random


# Set random seed for reproducibility
seed = 42
torch.manual_seed(seed)
random.seed(seed)
np.random.seed(seed)
torch.manual_seed(seed)
if torch.cuda.is_available():
    torch.cuda.manual_seed_all(seed)


DATA_PATH = r"C:\Users\Erik\Desktop\Advanced Applied Deep Learning In Physics And
Engineering\Datasets\Astronomy"
fp64_on_cpu = False

# Hyperparameters
#learning_rate = 1e-4

names = ["$T_{eff}$", "$\log g$", "$[Fe/H]$"]



# ============================== Jammy Flow functions ==============================

# Defining the normalizng flow model is a bit more involved and requires knowledge
of the jammy_flows library.
# Therefore, we provide the relevant code here.
class CombinedModel(nn.Module):
    """
    A combined model that integrates a normalizing flow with a CNN encoder.
    """

    def __init__(self, encoder, nf_type="diagonal_gaussian"):
        """
        Initializes the normalizing flow model.

        Parameters
        ----------
```

(https://uppsala.instructure.com/courses/102453/modules/items/1334990)

(https://uppsala.instructure.com/courses/102453/