Artificial Intelligence-Homework 4

Name： 张弛（ZHANG Chi）

SID： 12110821

Link of pull request: https://github.com/SustechSTA303/STA303-Assignment04/pull/53

Introduction

In this assignment, we are asked to make the conformal prediction of self-selected models by using the `TorchCP` library. For my work, I use 2 models (*ResNet18* and *DenseNet121*) and 2 datasets (*FashionMNIST* and *CIFAR-100*) to test the performance of 4 CP scores and 3 predictors.

- There are 3 main code files in my pull request.
  - data_prepare.py: to download and prepare the data. The function `load_data` returns the training data, calibration data, and test data by giving the specified dataset name.
  - model_prepare.py: to download and prepare the model. The function `training_model` can help train the model on a specified dataset and the default epoch is 20.
  - ass4.ipynb: the main file to test the performance. I try all permutations and combinations of 4 CP scores and 3 predictors.
- Datasets
  - The **FashionMNIST dataset** comprises 60,000 grayscale images of Zalando fashion items, with 10,000 additional test images. **With 10 distinct categories** such as T-shirt and Dress, the task involves classifying each image into one of these fashion classes.
  - In contrast, the **CIFAR-100 dataset** presents a more complex challenge, featuring 60,000 color images with **100 fine-grained categories** grouped into 20 superclasses. Each class has 500 training images and 100 testing images. CIFAR-100 requires classifying images into diverse categories, offering a broader range of objects for recognition compared to the fashion-focused FashionMNIST dataset.
  - These datasets stand as benchmarks for image classification tasks, each contributing unique characteristics to assess the performance of machine learning models.
- Model: I use the pre-trained models **ResNet18** and **DenseNet121** and retrain them using both two datasets. For each model, I modify its last linear layer so that it can work normally on new data sets and tasks. Therefore, there are a total of 4 models for following procedures, e,g. `densenet121_cifar100, densenet121_fashionmnist, resnet18_cifar100, resnet18_fashionmnist` .

  Here is the code of the model loading function in `model_prepare.py` .

```python
## model_prepare.py

def load_resnet18(num_classes):
    # 加载预训练的 ResNet18 模型
    resnet18 = torchvision.models.resnet18(pretrained=True)

    resnet18.fc = nn.Linear(resnet18.fc.in_features, num_classes)
    return resnet18

def load_densenet(num_classes):
    # 加载预训练的 DenseNet 模型
    densenet = torchvision.models.densenet121(pretrained=True)

    # 修改最后一层的全连接层
    in_features = densenet.classifier.in_features
    densenet.classifier = nn.Linear(in_features, num_classes)

    return densenet
```

Experiment

Experiment preparation

Conformal prediction (CP) is a framework in machine learning and statistics that provides confidence estimates for individual predictions made by a model. In the following part, I will introduce 4 score functions and 3 predictors, respectively:

- CP scores:
  - **THR (Threshold Conformal Predictors):** THR is designed for generating confidence intervals based on statistical confidence levels. It is grounded in statistical principles, providing a reliable assessment of prediction confidence and suitability for classification tasks.

    - **APS (Adaptive Prediction Sets):** APS is an adaptive prediction set method that considers the adaptability and flexibility of prediction sets. By adaptively adjusting the prediction set, it enhances robustness and accommodates diverse data features. APS excels in addressing changes and complexities in data.
    - **SAPS (Sorted Adaptive Prediction Sets):** SAPS is a sorted adaptive prediction set method that improves performance by sorting the prediction set. Sorting aids in better understanding the importance and impact of samples, making the prediction set more interpretable. SAPS is particularly useful in scenarios where the sorting of samples is needed. **The weight is set to 0.2.**
    - **RAPS (Regularized Adaptive Prediction Sets):** RAPS is an improvement over APS, introducing regularization to enhance performance. Regularization helps prevent overfitting and improves the algorithm's generalization on various datasets. RAPS may have advantages in handling large-scale, high-dimensional datasets and enhancing prediction robustness. **Here I set the penalty as 1.**
- predictors:
    - **ClusterPredictor** is a class-conditional conformal predictor designed to handle situations with many classes. The method likely involves techniques for efficient prediction and calibration within a large number of class labels.
    - **ClassWisePredictor** is a method designed for multi-class classification. It focuses on class-conditional conformal prediction, offering confidence levels for predictions in each class. It is particularly suitable in scenarios where distinguishing between multiple classes is crucial.
    - **SplitPredictor**

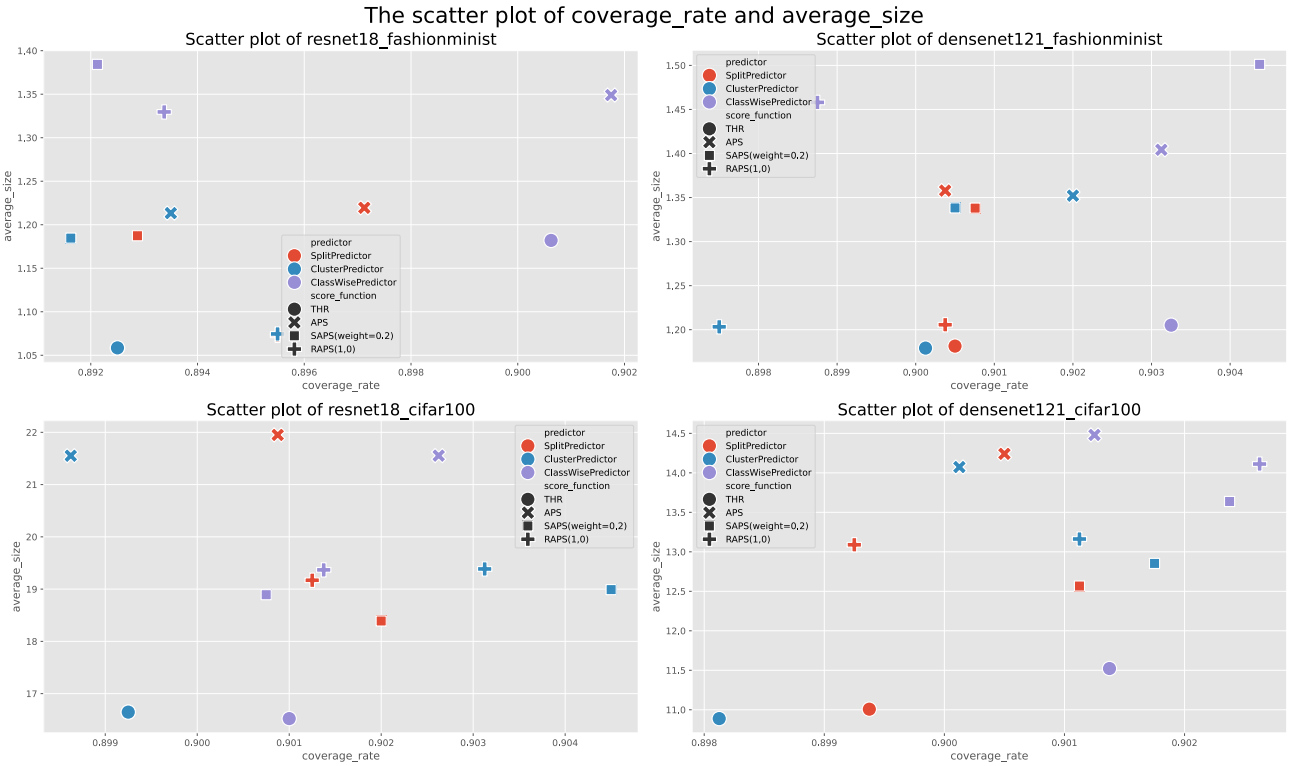
The accuracy of two models in two datasets' testing dataloaders is shown in the table below (epoch = 30). It is more difficult to classify images into 100 categories.

	Cifar100 (100 classes)	Fashionmnist (10 classes)
ResNet18	42.95%	85.32%
DenseNet121	49.66%	83.91%

There are 2 metrics used in my experiment, coverage rate and average size. The former means the probability of a model's prediction containing the correct class, measured by dividing the number of predictions containing the correct class by the total number of predictions. The latter indicates the average size of the prediction interval, with the significant level of 0.01. It is obvious that the the bigger the coverage rate and the smaller the average size be better.

Experiment Results

For each of 4 models (`densenet121_cifar100, densenet121_fashionmnist, resnet18_cifar100, resnet18_fashionmnist`), I plot a scatter plot of their coverage rates and average sizes. I use different colors to distinguish distinct predictors while taking different symbols to represent various score functions. The whole plots are shown below. Some points are blocked from each other because the values are too close, but it is only limited to the same symbol.



Firstly look at the overall situation of the four models in two values. Comparing the abscissa of the four graphs, we can find that the **coverage rate of the four models is from 0.89 to 0.90, which is not much different**. Compared with the vertical axis, **the running model on CIFAR-100' s average size will be larger** than the model running on FashionMNIST (CIFAR-100 is more difficult for classification with 100 classes), and the `densenet121_cifar100` performs better than `resnet18_cifar100` , with average size of 11 to 14.5 overall smaller than 17 to 22.

Next, we compare the influence of different score functions and predictors on the two indicators. In the model tested by the FashionMNIST dataset (the above two figures), the prediction interval size is within 1.5, and both of them perform well (the data is relatively simple), so there is no obvious regularity. It is worth mentioning that `ClassWisePredictor` (purple dot) always appears on the right and top of the image, denoting that the value of two metrics will be higher than other methods. In the next two graphs, we can see that **different symbols have different distributions for average size, showing the characteristics of APS > RAPS(1,0) > SAPS(weight=0.2) > THR, and the values are from high to low.**

Conclusion

In conclusion, the experiment focused on the conformal prediction of self-selected models using the `TorchCP` library. The study involved the utilization of two models, namely *ResNet18* and *DenseNet121*, trained on two diverse datasets—*FashionMNIST* and *CIFAR-100*. The exploration incorporated four CP scores (THR, APS, SAPS, RAPS) and three predictors (ClusterPredictor, ClassWisePredictor, SplitPredictor).

1. **Model Performance:**
    - The models, *ResNet18* and *DenseNet121*, demonstrated varying accuracies on the testing datasets. *ResNet18* achieved 42.95% accuracy on *CIFAR-100* and 85.32% on *FashionMNIST*, while *DenseNet121* achieved 49.66% and 83.91%, respectively.
    - *CIFAR-100* posed a more challenging classification task with 100 classes, reflected in the lower accuracy compared to *FashionMNIST*.
2. **Influence of Score Functions and Predictors:**
    - The coverage rates of the four models ranged from 0.89 to 0.90, with slight variations.
    - Notably, in the FashionMNIST dataset, the ClassWisePredictor consistently exhibited higher values in both coverage rate and average size compared to other predictors.
    - Different score functions demonstrated distinct distributions in average size, with APS > RAPS(1,0) > SAPS(weight=0.2) > THR in terms of performance.

For future improvements, consider expanding the experiment with additional models and datasets to assess generalizability. Fine-tune hyperparameters for CP scores and predictors to optimize performance. Explore algorithmic enhancements for improved efficiency and robustness. Incorporate extended evaluation metrics for a comprehensive assessment of model performance.

Appendix

In this section, I put all of my results into 4 tables for different models. For each grid, there are two values separated by a comma, where the former is the value of coverage rate and the latter is the value of average size.

resnet18_fashionminist

	THR()	APS()	SAPS(weight=0.2)	RAPS(1,0)
SplitPredictor	0.8925,1.05825	0.897125,1.2195	0.892875,1.187375	0.8955,1.073
ClusterPredictor	0.8925,1.0585	0.8935,1.213375	0.891625,1.184625	0.8955,1.0745
ClassWisePredictor	0.900625,1.182	0.90175,1.349	0.892125,1.38425	0.893375,1.329625

densenet121_fashionminist

	THR()	APS()	SAPS(weight=0.2)	RAPS(1,0)
SplitPredictor	0.9005,1.181375	0.900375,1.357875	0.90075,1.338	0.900375,1.205625
ClusterPredictor	0.900125,1.179125	0.902,1.352125	0.9005,1.33825	0.8975,1.20325
ClassWisePredictor	0.90325,1.205125	0.903125,1.40425	0.904375,1.501125	0.89875,1.458

resnet18_cifar100

	THR()	APS()	SAPS(weight=0.2)	RAPS(1,0)
SplitPredictor	0.89925,16.645	0.900875,21.9505	0.902,18.389875	0.90125,19.169625
ClusterPredictor	0.89925,16.64625	0.898625,21.552	0.9045,18.990625	0.903125,19.38575
ClassWisePredictor	0.901,16.52225	0.902625,21.554125	0.90075,18.893375	0.901375,19.368

densenet121_cifar100

	THR()	APS()	SAPS(weight=0.2)	RAPS(1,0)
SplitPredictor	0.899375,11.0075	0.9005,14.24125	0.901125,12.566875	0.89925,13.089375
ClusterPredictor	0.898125,10.88825	0.900125,14.0735	0.90175,12.853	0.901125,13.16225
ClassWisePredictor	0.901375,11.52225	0.90125,14.480375	0.902375,13.637125	0.902625,14.11175

Reference

https://github.com/ml-stat-Sustech/TorchCP