toppred

Release 0.0.2

Thijs van Ede

Jan 19, 2023

CONTENTS:

Extension to sklearn.metrics to allow metrics for classifiers that output a top `n` prediction. Some classifiers output confidence levels for each class. Oftentimes, you want to evaluate the performance of such classifiers assuming the correct prediction is the top `n` predictions with the highest confidence level. This library serves as an extension to the functions provided by sklearn.metrics to allow for evaluating classifiers that do not output a single prediction per sample, but rather a range of top predictions per sample.

CHAPTER

ONE

INSTALLATION

The most straightforward way of installing `toppred` is via pip:

```
pip3 install toppred
```

1.1 From source

To install this library from source, simply clone the repository:

```
git clone https://github.com/Thijsvanede/toppred.git
```

Next, you can install the `toppred` from source:

```
pip install -e <path/to/toppred/>
```

1.1.1 Dependencies

`toppred` requires the following python packages to be installed:

- Numpy: https://numpy.org
- Pandas: https://pandas.pydata.org/
- Scikit-learn: https://scikit-learn.org/stable/index.html

All dependencies should be automatically downloaded if you install `toppred` via pip. However, should you want to install these libraries manually, you can install the dependencies using the requirements.txt file

```
pip install -r requirements.txt
```

Or you can install these libraries yourself

```
pip install -U numpy pandas sklearn
```

CHAPTER

TWO

USAGE

The main usage of this library is to compute metrics over the top-n predictions of a given classifier. In the normal case, a classifier gives a single prediction per sample, often in the form of an array:

```
import numpy as np

y_true = np.asarray([0, 1, 2, 1, 0]) # True labels
y_pred = np.asarray([0, 1, 1, 0, 0]) # Predicted labels
```

However, a classifier could also return the top n most likely predictions, e.g.:

```
import numpy as np

y_true = np.asarray([0, 1, 2, 1, 0]) # True labels
y_pred = np.asarray([                # Predicted labels
    [0, 1, 2],
    [1, 0, 2],
    [1, 2, 0],
    [0, 1, 2],
    [0, 1, 2],
])
```

In this case, we would like to be able to compute the performance when:

- The correct prediction is the most likely prediction (`y_pred[:, 0]`)
- The correct prediction is in the top 2 most likely predictions (`y_pred[:, :2]`)
- The correct prediction is in the top 3 most likely predictions (`y_pred[:, :3]`)

For this purpose, `toppred` provides two functions:

> `top_classification_report()`, produces a classification report similar to `sklearn.metrics.classification_report`. See usage example. `top_predictions()`, provides an iterator over the top n most likely predictions and can be combined with many `sklearn.metrics`. See usage example.

See the *Reference* API for detailed information, and our usage examples below.

2.1 Examples

2.1.1 top_classification_report

One of the main functions of `toppred` is to create a `top_classification_report()` similar to scikit-learn's classification_report. When we have a numpy array `y_pred` shape `(n_samples, n_top_predictions)`, we can use it as follows to generate a `top_classification_report()`.

```
# Imports
import numpy as np
from toppred.metrics import top_classification_report

# Define inputs
y_true = np.asarray([1, 2, 3, 2, 1]) # Ground truth values
y_pred = np.asarray([                # Sample prediction values
    [1, 2, 3],                       # We have a top 3 predictions for each
    [2, 1, 3],                       # input sample. I.e.,
    [1, 2, 3],                       # y_true.shape[0] == y_pred.shape[0].
    [3, 1, 2],
    [1, 2, 3],
])

# Compute and print top classification report
print(top_classification_report(
    y_true = y_true,
    y_pred = y_pred,
    labels = [0, 1, 2, 3],                 # Optional, defaults to None
    target_names = ['N/A', '1', '2', '3'], # Optional, defaults to None
    sample_weight = [1, 2, 3, 4, 5],       # Optional, defaults to None
    digits = 4,                            # Optional, int, defaults to 2
    output_dict = False,                   # Optional, If true, return as dictionary
    zero_division = "warn",                # Optional, defaults to "warn"
))
```

2.1.2 metrics

Besides the `top_classification_report()`, you may want to compute other metrics for the top `n` results in your prediction. To this end, we provide the `top_predictions()` which takes the `y_pred` of shape `(n_samples, n_top_predictions)` and yields a `y_pred` of shape `(n_samples,)` for each top `i` predictions that is compatible with most `sklearn.metrics` functions.

```
# Imports
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from toppred.predictions import top_predictions

# Define inputs
y_true = np.asarray([1, 2, 3, 2, 1]) # Ground truth values
y_pred = np.asarray([                # Sample prediction values
    [1, 2, 3],                       # We have a top 3 predictions for each
    [2, 1, 3],                       # input sample. I.e.,
```

(continues on next page)

(continued from previous page)

```
    [1, 2, 3],                      # y_true.shape[0] == y_pred.shape[0].
    [3, 1, 2],
    [1, 2, 3],
])

# Use top_predictions to generate a y_pred value that is correct if the
# prediction is in the top n predictions
for top, prediction in top_predictions(y_true, y_pred):
    # Compute common metrics
    accuracy  = accuracy_score (y_true, prediction)
    precision = precision_score(y_true, prediction, average='macro')
    recall    = recall_score   (y_true, prediction, average='macro')
    f1        = f1_score       (y_true, prediction, average='macro')

    print(f"Metrics top {top+1} predictions:")
    print(f"    Accuracy : {accuracy}")
    print(f"    Precision: {precision}")
    print(f"    Recall   : {recall}")
    print(f"    F1_score : {f1}")
    print()
```

2.2 Probabilities

Some classifiers, including many neural networks do not give direct top n results, but instead provide a probability (confidence level) for each class, producing an output such as:

```
import numpy as np

y_true = np.asarray([0, 1, 2, 1, 0]) # True labels
y_prob = np.asarray([ # Prediction probability
    [0.7, 0.2, 0.1],  # class 0 -> 0.7, class 1 -> 0.2, class 2 -> 0.1
    [0.2, 0.7, 0.1],  # etc.
    [0.1, 0.7, 0.2],
    [0.8, 0.1, 0.1],
    [0.7, 0.2, 0.1],
])
```

In those cases, we can obtain a prediction for the top n most likely values:

```
# Get top n most likely values
n = 3

# Example: y_prob is numpy array
y_pred = np.argsort(y_prob, axis=1)[:, -n:]

# Example: y_prob is pytorch Tensor
y_pred = torch.topk(y_prob, n).indices.cpu().numpy()
```

This results in the prediction array:

```
array([[0, 1, 2],
       [1, 0, 2],
       [1, 2, 0],
       [0, 1, 2],
       [0, 1, 2]])
```

CHAPTER

THREE

REFERENCE

This is the reference documentation for the methods provided by the `toppred` module.

3.1 Metrics

As a substitute for `sklearn.metrics.classification_report` we offer the `top_classification_report()` method to produce a classification report containing the metrics for all top `n` predictions given as `y_pred`. See *Usage* for usage examples.

3.2 Predictions

When you want to compute metrics not supported in this library, we provide an iterator that yields `y_pred` arrays for each top `i` prediction that are compatible with `sklearn.metrics`. See *Usage* for usage examples.

CHAPTER

FOUR

CONTRIBUTORS

This page lists all the contributors to this project. If you want to be involved in maintaining code or adding new features, please email t(dot)s(dot)vanede(at)utwente(dot)nl.

4.1 Code

- Thijs van Ede

CHAPTER

FIVE

LICENSE

MIT License

Copyright (c) 2022 Thijs van Ede

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.