sdmdl Documentation

Release latest

Oct 10, 2019

Contents

This toolkit was built to maximize ease of use while still offering in-depth parameter control for deep learning models. As such this package offers only a single class and a handful of functions. If further customization of these models is required it comes with a config.yml file that can be edited in order to change important model hyper parameters.

the SDMDL package works as follows:

```
model = sdmdl('path')
model.prep()
model.train()
model.predict()
```

CHAPTER 1

Installation

The installation of the sdmdl package can be performed in three steps:

1. Install GDAL separately as this is an external dependency that cannot be obtained through Python. This step also includes the installation of the GDAL python package.
2. Install the package locally, by using this code snippet:

```
import os
os.chdir('directory_path_of_repository_root')
pip install .
```

CHAPTER 2

Requirements

To create an sdmdl object and subsequently train deep learning models a few requirements need to be met.

1. Several input files (simply obtainable by copying or cloning the git repo).
2. A set of environmental rasters (.tif) which will serve as the source of data for the deep learning process. This project distinguishes between two types of environmental layers:
   i. Scaled layers, that need to be scaled during the process of preparing the data.
   ii. Non-scaled layers, that are already normalized or are categorical (e.g. 0 = not present while 1 = present).

   Note: all environmental layers need to have the same affine transformation and resolution to be usable for data preparations. This includes the file 'empty_land_map.tif' that is included in the git repo. This entails that the affine transformation and resolution of the input rasters needs to match the affine transformation and resolution of 'empty_land_map.tif'.
3. A set of occurrences (.csv or .xls) that will serve as training examples of where the species currently occurs. To be detectable as occurrence files, these tables need to have two required columns:
   i. 'decimalLatitude' or 'decimallatitude' holding the latitude for each occurrence.
   ii. 'decimalLongitude' or 'decimalLongitude', holding the longitude for each occurrence.

   Note: The occurrence coordinates are currently not checked before starting data preparations. So be aware that any obviously wrong values will cause an error. This includes any values of the wrong datatype (anything that is not numerical) and coordinates that are outside the spatial extent of the provided raster files.

Before running the tool a few important details should be taken into account:

1. Scaled tif layers should be inserted into the ''root/data/gis/layers/scaled''
2. Non-scaled tif layers should be inserted into the ''root/data/gis/layers/non-scaled''
3. Occurrences should be inserted into the ''root/data/occurrences''

CHAPTER 3

Configuration

If these locations are not convenient it is possible to change these locations using the config.yml file in “root/data”. Config.yml is initiated the first time an sdmdl object is created. And holds any relevant information on:

1. Detected raster files.
2. Detected occurrence files.
3. Model parameters:
    a. **integer** random_seed: makes a randomized process deterministic.
    b. **integer** pseudo_freq: number of sampled pseudo absences.
    c. **integer** batchsize: number of data points given to the model during training at once.
    d. **integer** epoch: number of (training) iterations over the whole data set.
    e. **integer** model_layers: number of nodes per layer. Adding extra items to the list makes the model deeper.
    f. **float** model_dropout: dropout deactivates a percentage of nodes during training (0 = no nodes are turned off and 1 = all nodes are turned off).
    g. **boolean** Verbose: if True prints progress bars

Note: changes to the config file are not updated automatically, for changes to take effect a new sdmdl objects needs to be created.

CHAPTER 4

Example

Step 1: create a sdmdl object:

```
model = sdmdl('directory_path_of_repository_root')
```

Step 2: prepare data:

```
model.prep()
```

Step 3: train the model(s):

```
model.train()
```

Step 4: predict global distribution:

```
model.predict()
```

Step 5: remove temporary data:

```
model.clean()
```

CHAPTER 5

Outputs

The output of step 2 consists of the following files:

- several temporary files that are used as inputs for step 3 and 4.

The output of step 3 consists of the following files:

- Performance metrics for each species, can be found at 'root/results/_DNN_performance/DNN_eval.txt'.
- Model files that save the final state of the best performing model after training. For each species two files can be found at 'root/results/species_name/' the files are named after their respective species, and have the file extensions: .h5 and .json.
- Feature impact graph that shows the importance of individual variables. This graph is included for every species and can be found at 'root/results/species_name/', the file is named after its respective species followed by 'feature importance' and has the file extension: .png.

The output of step 4 consists of the following files:

- Prediction map that shows the global predicted distribution of a species, on a scale from 0 to 1 indicating the probability of presence. One illustration is included for every species and can be found at 'root/results/species_name/', the file is named after its respective species followed by 'predicted map color' and has the file extension .png.
- Raster file with the global predicted distribution of a species (on a scale from 0 to 1). This file is included to allow further analysis using the species distribution. One raster file is included for every species and can be found at 'root/results/species_name/', the file is named after its respective species followed by 'predicted map' and has the file extension .tif.