GDAL OTB QGOS in a virtual environment

The Orfeo (https://www.orfeo-toolbox.org/tag/machine-learning/) machine learning library and the translator library for raster and vector geospatial data formats GDAL (https://gdal.org/) do not play well with the open source geographic information system QGIS (https://qgis.org) outside of the resource intensive GUI environment.

The systems have differing and partially conflicting dependencies, and the GUI environment makes rapid testing across the various tool boxes cumbersome. This fact becomes apparent, for example, when developing a pipeline to perform vector based satellite image classification with a neural network under OTB.

This repository demonstrates how to setup a virtual computer that can support GDAL, OTB and QGIS functionality seamlessly. It also includes examples of how to use the setup to apply various classifiers including vector support machines, random forests and neural networks across GDAL, OTB and QGIS.

1 – Create and launch a VM
Create a virtual computer.
Choose Ubuntu 18.04 LTS, with at least 16GB of RAM.
Start the new VM.

2 - Install QGIS
Upload the batch file basics+qgis.sh
Make that file executable.

```
chmod +x basics+qgis.sh
```

Run that file

```
sh basics+qgis.sh
```

3 - Install OTB in a virtual environment with conda
Upload the batchfile conda-packages.sh and the context file environment.yml
Make that file executable.

```
chmod +x conda-packages.sh
```

Run that file

```
sh conda-packages.sh
```

4 - Create a conda environment with defined dependencies.
(OTB 7.2 requires python 3.7, for example)

```
conda env create -f environment.yml
```

5 - Personalize
Add other libraries to the OTB environment as needed

```
conda install -c conda-forge pillow
conda install -c conda-forge geopandas
```

6 – Setup the directory structure

Create the following directories: code, data, results, rasterimages, vectorfiles (mkdir X).
Place image assets (.tif) in the rasterimages directory and vector assets (.shp) into the vectorfiles directory. Add the corresponding .dbf, .prj, .shx files for each .shp file.

7 - Test the setup

a) Test QGIS

Run the qgis test in the base environment.

```
python3 qgis_test.py
```

b) Test OTB

Activate the OTB environment.

```
conda activate OTB
```

Run the otb test

```
python3 otb_test.py
```

You can now use QGIS in the base installation and enable the conda environment to access OTB functionality. You can also move across environments to access libraries from either environment with python scripts as outlined below (see vector_classify_top.sh):

```
echo "Starting OTB-QGIS pipeline...\n\n"

conda run -n OTB python3 /home/code/otb_code_A.py

python3 /home/code/qgis_code_A.py

conda run -n OTB python3 /home/code/otb_code_B.py

python3 /home/code/qgis_code_B.py
```

Since the directory structure you setup will be identical in both the QGIS and OTB environments, intermediate filed produced will be available to processes running in either environment, allowing for data to be shared at no extra computational cost. All settings across the script modules are stored in the 'settings.txt' resource file and imported to the individual modules.

Examples:

After installation and testing you can perform operations as such (provided you have already prepared your data accordingly.

Pixel based classification (SVM or RF)

1) Open the settings.txt file and change the parameters “rasterimage” and “rastershapefile” to your data.

2) Enable the OTB environment:
conda activate OTB

3) Run the program with the classifier of choice as argument (libsvm or rf only)
python3 otb_raster_classify.py svm

Object based classification (ANN)

1) Open the settings.txt file and change the parameters “rasterimage” and “pointsfile” to your data.

2) Run the program:
sh vector_classify_top.sh

This script runs all the processes across multiple otb and qgis scripts to perform the object based classification.