Lab 12 of SP19-BL-ENGR-E599-30563

Lab 12 will demonstrate you an end-to-end pipeline workflow of clustering image files by using K-means algorithm.

Goal

- Understand the workflow of machine learning application.
- Combine the knowledge learned in the class

Deliverables

Change the training iteration number from 10 to 2 and run the workflow pipeline by Harp-DAAL K-means. Submit the generated daal.pdf file

Evaluation

Lab participation: credit for 1 point based upon a successful completion of the lab tasks

The Image Clustering

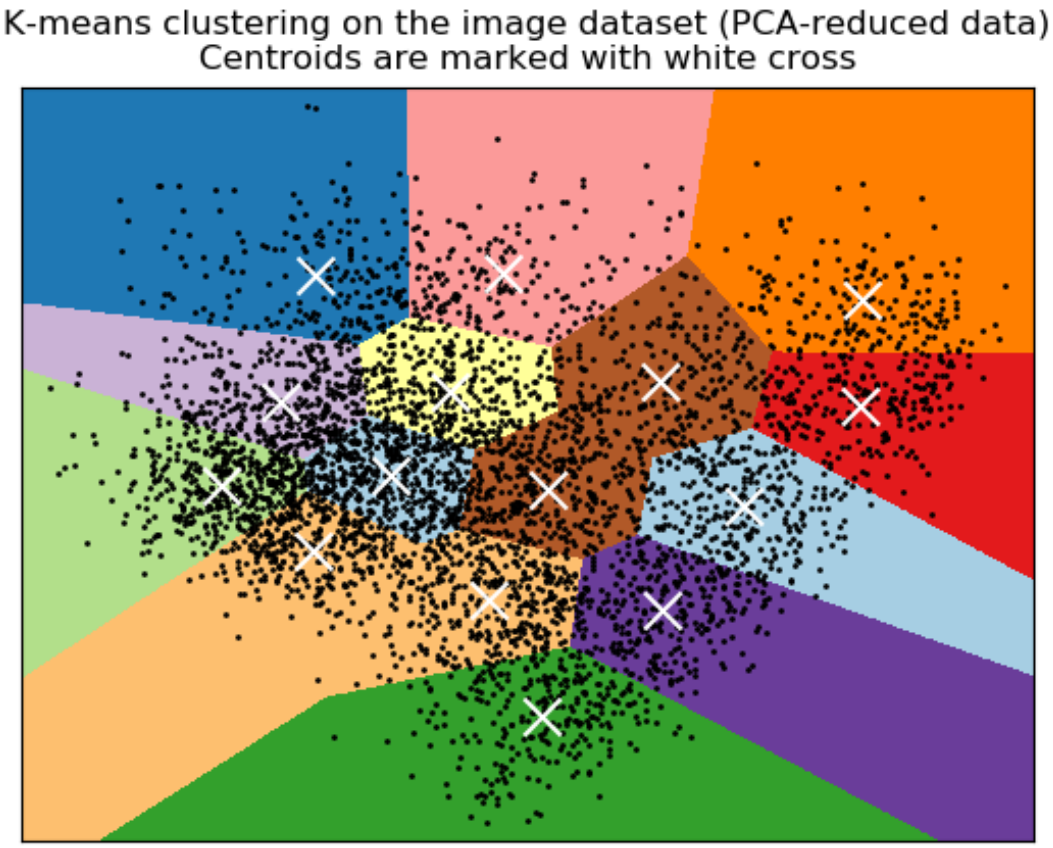
Image clustering is a typical unsupervised machine learning application. People use it to process large volume of unlabeled image files before the classification algorithm.

Database of ImageNet

Online database such as ImageNet includes millions of pictures, which challenges the performance and scalability of algorithms such as K-means clustering. Nevertheless, the pre-processing stage of raw is as important as the training phase, and we usually combine different tools in a workflow pipeline in such cases.

Here, we have a hands-on demo presented at Supercomputing Conference 2017 to give you a flavor of building up the workflow pipeline from raw image files to labeled clustered objects.

The hands-on codes take three steps to construct a workflow and use python language to interface different tools.



A 2D clustering results

Code Walk-through and Important API

Please copy the folder at `/share/jproject/lc37/imgKmeans` into your home directory.

```
cp -r /share/jproject/lc37/imgKmeans ~/
cd imgKmeans/
```

Open the script file named `run_kmeans.sh`

```
vim run_kmeans.sh
```

The bash script divides the workflow pipeline into four parts

- Set up required environment variables
- Extract features from raw image files
- Select Sklearn (local mode) or Harp-DAAL to run K-means training jobs
- Post-process results and label the objects in each cluster

Set up Environment

All of the python packages used in this hands-on session are pre-built and installed at `/share/jproject/lc37/anaconda2-3.5`. Before you run the scripts, make sure to add following contents to your `~/.bashrc` file under your home directory.

```
vim ~/.bashrc
```

```
# added by Anaconda2 5.3.0 installer
# >>> conda init >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$(CONDA_REPORT_ERRORS=false '/share/jproject/lc37/anaconda2-
3.5/bin/conda' shell.bash hook 2> /dev/null)"
if [ $? -eq 0 ]; then
    \eval "$__conda_setup"
else
    if [ -f "/share/jproject/lc37/anaconda2-3.5/etc/profile.d/conda.sh" ]; then
        . "/share/jproject/lc37/anaconda2-3.5/etc/profile.d/conda.sh"
        CONDA_CHANGEPS1=false conda activate base
    else
        \export PATH="/share/jproject/lc37/anaconda2-3.5/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda init <<<
```

```
source ~/.bashrc
```

After that, you shall be able to find all of the packages in anaconda2, which are required by this hands-on code

Package	version	Usage
python	>= 2.7.12	Programming Language
numpy	>= 1.13.1	Library for large, multi-dimensional arrays and matrices
scipy	>= 0.19.1	Python Scientific Computing Library
scikit-learn	>= 0.19.0	Machine Learning in Python
matplotlib	>= 2.0.2	Matplotlib: Python plotting
pillow	>= 3.2.0	Python Imaging Library
scikit-surprise	>= 1.0.4	A Python scikit for recommender systems

The following content in `run_kmeans.sh` defines all of the environment variables for running Sklearn and Harp-DAAL packages. Pay attention to line 6 of selecting the python binary shipped with Ananconda2-3.5, which shall avoid the pollution from system default python packages.

```
4 homedir=`dirname $0`
5 homedir=`dirname $homedir`
6 pythonLocal=/share/jproject/lc37/anaconda2-3.5/bin/python
7
8 ## ------------- set up path for harpdaal kmeans -------------
9 setupdaalenv()
10 {
11
12 ## replace this var by your harp labsession branch installation path
13 export MyLabSRoot=/N/u/lc37/Project/LabSession/harp
14
15 ##
16 export HARP_JAR=$MyLabSRoot/ml/java/target/harp-java-0.1.0.jar
17 export HARP_DAAL_JAR=$MyLabSRoot/ml/daal/target/harp-daal-0.1.0.jar
18 export DAALROOT=$MyLabSRoot/third_party/daal-2018
19
20 ## load daal so files into hdfs
21 hdfs dfs -mkdir -p /Hadoop
22 hdfs dfs -mkdir -p /Hadoop/Libraries
23 hdfs dfs -rm /Hadoop/Libraries/*
24 hdfs dfs -put ${DAALROOT}/lib/intel64_lin/libJavaAPI.so /Hadoop/Libraries/
25 hdfs dfs -put ${DAALROOT}/../tbb/lib/intel64_lin/gcc4.4/libtbb*
/Hadoop/Libraries/
26 # check that safemode is not enabled
27 hdfs dfsadmin -safemode get | grep -q "ON"
28 if [[ "$?" = "0"  ]]; then
29     hdfs dfsadmin -safemode leave
```

```
30 fi
31
32 }
33
34 ## the python path
35 export PYTHONPATH=${homedir}
```

The `setupdaalenv` function prepares the harp-daal runtime variables. Please replace the `MyLabSRoot` by using your own harp installation path in previous lab sessions

```
export MyLabSRoot=<Path-to-your-harp-lab-codes-installation>
```

Finally set up the user-defined `PYTHONPATH`

Feature Extraction

Feature extraction step pre-processes the raw image data and generate the features in the form of dense vectors with a given dimension.

Example of image files

The script invokes the tools in `tool/devkit-1.0/feature` to extract features, and converts the feature vectors into Numpy array file `15scene.npz`, which will be delivered to the second stage of the workflow pipeline

```
${pythonLocal} $homedir/src/make_15scene.py --input $homedir/data/15scene/ --
output 15scene
```

K-means Clustering (Training)

With the extracted feature vectors in Numpy array, where each feature vector refers to a raw picture data, we use K-means clustering algorithm to train them. We have two options here,

- The K-means implementation from Scikit-learn

- The K-means codes from Harp-DAAL

To use K-means by Scikit-learn

```
${pythonLocal} $homedir/tutorial/demo_kmeans_local.py 15scene.npz 10
```

To use K-means by Harp-DAAL

```
${pythonLocal} $homedir/tutorial/demo_kmeans_daal.py 15scene.npz 10
```

, where `15scene.npz` is the Numpy feature vectors, and the iteration number is 10

Retrieve Training Data Information

It first retrieves the three columns from `15scene.npz`

- `feature` column is the digital values of feature vector
- `labels` column is the digital value of each label
- `rids` column is the readable text description of each label

Here each `label` is a centroid in K-means, and each feature vector (image object) shall be tagged by one label. The value `K` is the number of labels

Training by Scikit-Learn

To use scikit learn, import the package

```
39 from sklearn.cluster import KMeans
```

Line 123 chooses the `KMeans` of scikit learn as the estimator and feed the parameters

```
123 _kmeans = KMeans(init='random', n_clusters=n_digits, max_iter=1000, tol=0,
n_init=1, n_jobs=1)
```

Line 124 feed the estimator with the training data and start the training process

```
124 bench_k_means(_kmeans, name="local", data=data)
```

After the training process, each of the feature vector (image object) has been tagged a label, and the program saves the labels result in a file named `local.cluster`

Training by Harp-DAAL K-means

In previous lab sessions, we have already used the Harp-DAAL K-means through Python interface. This time, Harp-DAAL K-means is part of the workflow and accepted the feature vectors from numpy array and store it in HDFS. In `demo_kmeans_daal.py`, it includes the `DAALKMeans` package

```
46 from daal_kmeans import DAALKMeans
```

Similarly, it creates an estimator by using Harp-DAAL K-means and feeds the estimator with training data

```
127 _kmeans = DAALKMeans(n_clusters=n_digits, max_iter=1000, n_init=1,
workdir="/15scene-work")
128 bench_k_means(_kmeans, name="daal", data=data)
```

After the training process, each of the feature vector (image object) has been tagged a label, and the program saves the labels result in a file named `daal.cluster`

Visualizing the Results

Finally, to show the raw images with assigned labels, we run the following codes in `run_kmeans.sh`

```
91 ${pythonLocal} $homedir/tutorial/show_kmeans.py $homedir/data/15scene/
${runver}.cluster 10
```

This script will output a pdf file containing each image with a description label.

The Labels to the Clustered Images

Run the Hands-on Codes

To run the hands-on codes, please make sure to launch the Hadoop service first

```
## log in to your namenode
$ ssh j-xxx
$ cd $HADOOP_HOME/sbin
$ ./start-dfs.sh
$ ./start-yarn.sh
```

To check the status of Hadoop service

```
$ yarn node -list
```

You shall have either one or two nodes at the screen output.

To run the training by Scikit-Learn on local node

```
./run_kmeans.sh
```

Before running Harp-DAAL K-means training, you could run a `clean_tmp_files.sh` to remove generated intermediate data

```
./clean_tmp_files.sh
```

To run Harp-DAAL on two nodes

```
./run_kmeans.sh daal
```

If you only has one Harp node, you may also run this script but first navigate to line 17 of file `./tutorial/daal_kmeans.py` You shall change the value of variable `n_node` from 2 to 1, and launch

```
./run_kmeans.sh daal
```