**Big Data Analysis**

**Analyzing Neuroimaging Data with PySpark and Thunder**

Ji-Hyeong Han

## ( [jhhan@seoultech.ac.kr](mailto:jhhan@seoultech.ac.kr) )

Dept. of Computer Science and Engineering



**Contents**

1. PySpark
2. Thunder
3. Practice

2

# PySpark and Thunder

3

**Python**

## Python is a favorite tool for many data scientist

* + High-level language that is easy to use and learn
  + Extensive library system ranging from niche numerical calculations to web-scraping utilities to data visualization tools

## Important libraries

* + numpy / scipy / matplotlib
    - Recapitulate typical MATLAB functionality, including fast array operations, scientific functions, and a widely used MATLAB-inspired plotting library
  + pandas
    - Provide functionality similar to R’s data.frame
  + scikit-learn / statsmodels
    - Provide high-quality implementations of machine learning algorithms (e.g., classification, regression, clustering, matrix factorization) and statistical models
  + nltk
    - A popular library for natural language processing

4

**PySpark**

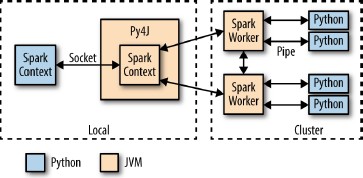
## PySpark API for interacting with Spark through Python

* PySpark is a particularly flexible tool for exploratory big data analysis because it integrates well with the rest of the PyData ecosystem, including matplotlib for visualization
* PySpark API document <http://spark.apache.org/docs/latest/api/python/index.html>

5

**PySpark**

* PySpark internal architecture



* + When PySpark’s python interpreter starts, it also starts a JVM with which it communicates through a socket
  + PySpark uses the Py4J project to handle this communication
  + The JVM functions as the actual Spark driver, and loads a “JavaSparkContext” that communicates with the Spark executors across clusters

6

**PySpark**

## Import Spark classes



* Initializing Spark
  + Create a SparkContext object



## RDDs

* + Parallelized collections are created by calling SparkContext’s parallelize method
  + Once created, the distributed dataset (distData) can be operated on in parallel

7

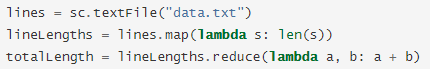
**PySpark**

## RDDs (cont’d)

* + External datasets
  + Once created, distFile can be acted on by dataset operations



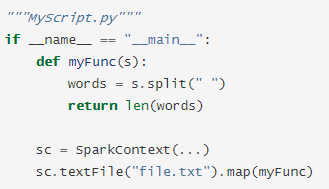
## RDD operations



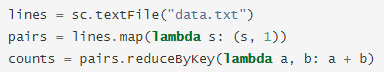
8

**PySpark**

## Passing functions



* Working with Key-Value pairs



9

**Thunder**

## Thunder is an ecosystem of tools for the analysis of image and time series data in Python

* + It provides data structures and algorithms for loading, processing, and analyzing image and time series data
  + Can be useful in a variety domains, including neuroscience, medical imaging, video processing, and geospatial and climate analysis

## Thunder can be used locally, but also supports large-scale analysis through the distributed computing engine Sparks

* + All data structures and analyses in Thunder are designed to run identically and with the same API whether local or distributed

## Thunder is designed around modularity and composability

* + The core thunder package only defines common data structures and read/write patterns
  + Most functionality is broken out into several related packages, like thunder-registration and thunder-regression

10

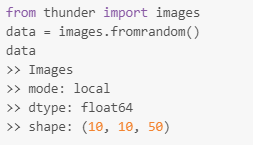
**Thunder**

## Use with Spark

* + Thunder is compatible with Spark versions 1.5 and higher
  + Install Thunder on both the master node and each worker node of the cluster



* + All the loading methods in Thunder take an optional argument “engine”, which can be passed a SparkContext



11

**Thunder**

## Two primary data types of Thunder

* + “images” and “series”
  + High-level wrapper for an ndarray

## Images

* + To represent a collection of images or volumes
  + Can be loaded from several on-disk formats, including tiff, png, and binary
  + Several domain-specific methods are available for image processing, filtering, and writing to external formats

## Series

* + To represent a collection of one-dimensional records that share a common index
  + Well suited to time series data
  + Domain-specific methods are available for filtering, preprocessing, and statistical aggregation conditional on the index 12

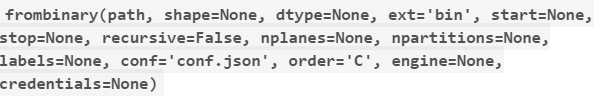
**Thunder**

## Image loading

* + Load images from an array



* + - First dimension will be used to index images, so remaining dimensions after the first should be the dimensions of the images
    - (3, 100, 200) means 3 x (100, 200) images
  + Load images from flat binary files



* + Load example image data



* + More load methods: <http://docs.thunder-project.org/image-loading>

13

**Thunder**

## Image methods

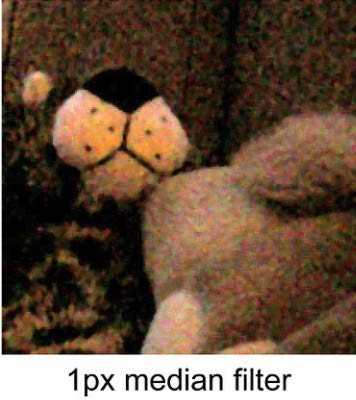
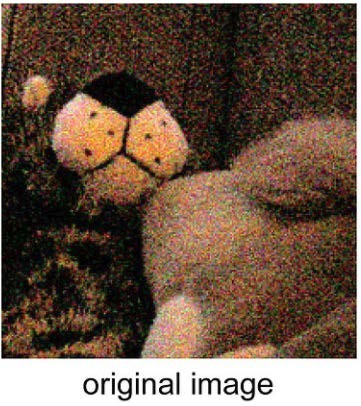
* + Spatially smooth images with a Gaussian filter



* + - sigma = size of the filter size as standard deviation in pixels
    - order = order of the Gaussian kernel
  + Spatially filter images using a median filter



* + - size = size of the filter neighborhood in pixels

14

# Practice

15

**Prepare Data**

## Download “Lec8\_FishLong.zip” from e-class

* Create “fish-long” folder in Jupyter Notebook and upload the zip file to the created folder
* Unzip the uploaded zip file in terminal

# cd fish-long

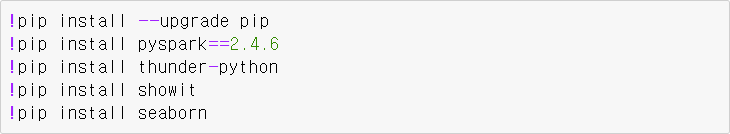
# unzip Lec8\_FishLong.zip

## Delete the uploaded zip file

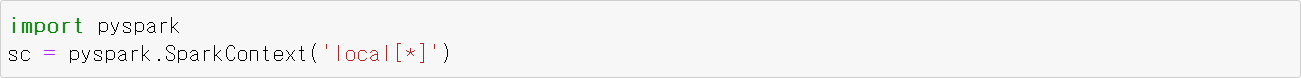
16

**PySpark and Thunder for Image Processing**

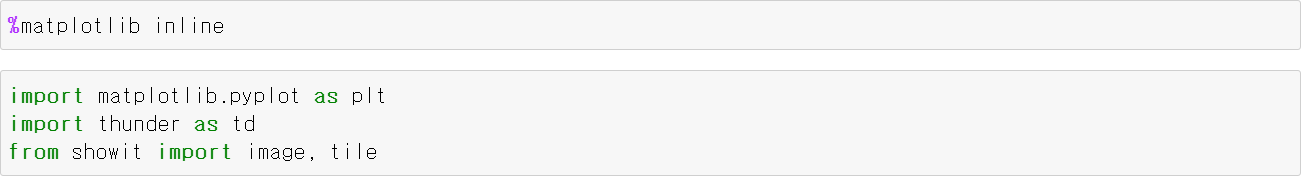
## Install pyspark and so on



* Start SparkContext with PySpark



* Import libraries

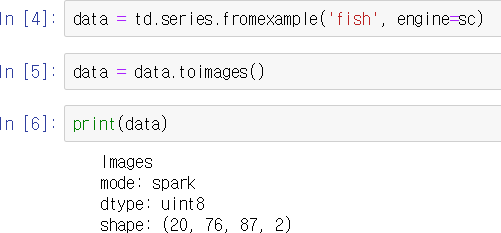


17

**PySpark and Thunder for Image Processing**

## Load sample image

* + Zebrafish’s brain image (neuro image)



18

**PySpark and Thunder for Image Processing**

## Plot the image



19

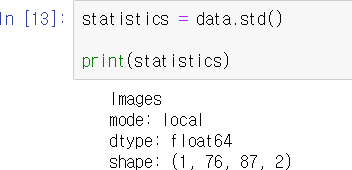
**PySpark and Thunder for Image Processing**

## Subsampling the image

 20

**PySpark and Thunder for Image Processing**

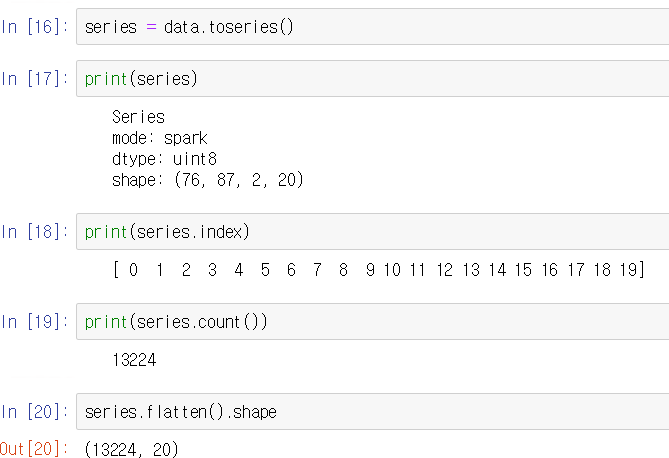
## Aggregate the images based on standard deviation



21

**PySpark and Thunder for Image Processing**

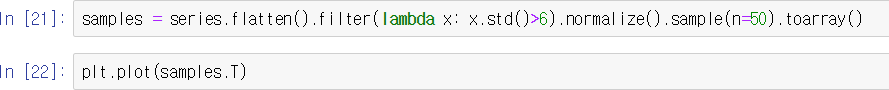
## Convert image to series data

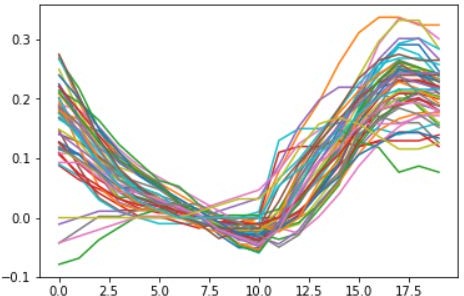


22

**PySpark and Thunder for Image Processing**

## Flatten the series data and filtering on standard deviation

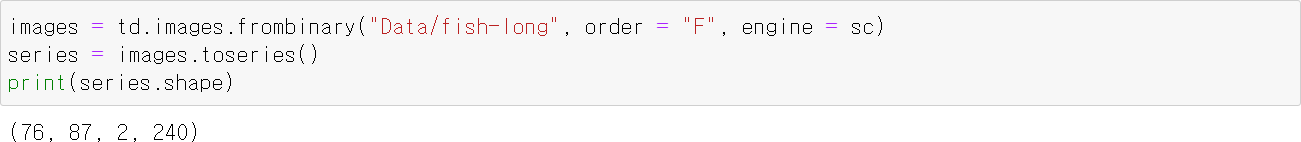




23

**Clustering Neuron Types with PySpark and Thunder**

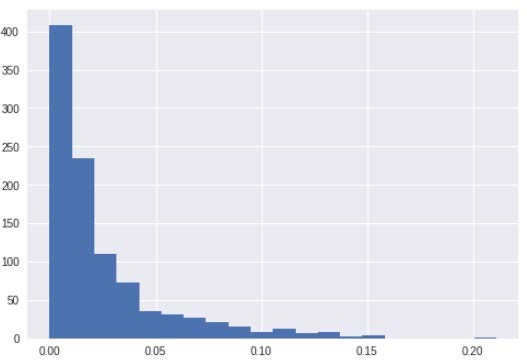
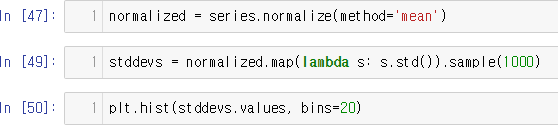
## Using more longer time data



24

**Clustering Neuron Types with PySpark and Thunder**

## Compute the standard deviation and plot a histogram of 10% sample

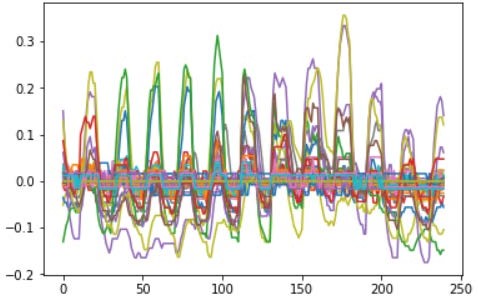


25

**Clustering Neuron Types with PySpark and Thunder**

## If we plot randomly chosen 50 neurons



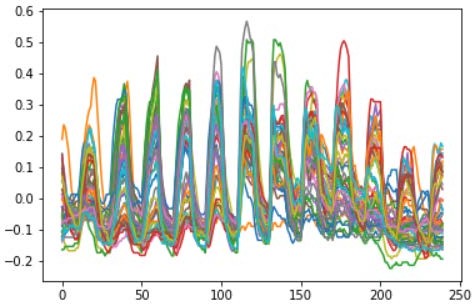


26

**Clustering Neuron Types with PySpark and Thunder**

## If we choose a threshold of 0.1 to look at the most “active” series



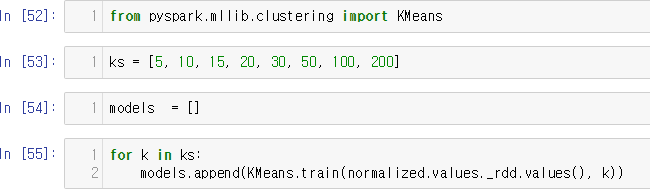


27

**Clustering Neuron Types with PySpark and Thunder**

## Cluster the neurons based on the activate patterns

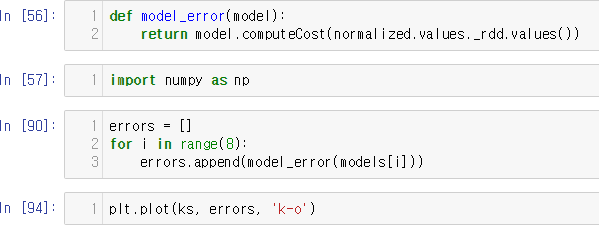
* + Using K-means clustering

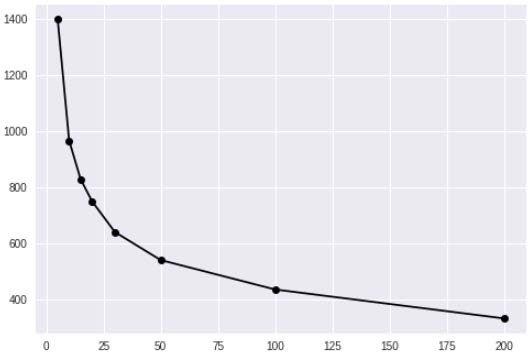


28

**Clustering Neuron Types with PySpark and Thunder**

## Calculate and plot the cost for each clustering model

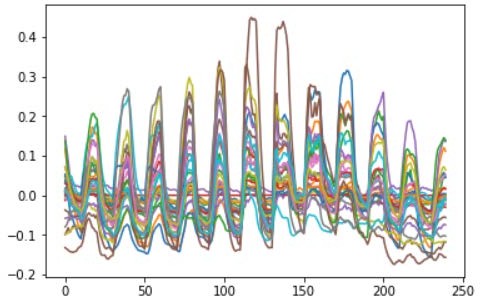


29

**Clustering Neuron Types with PySpark and Thunder**

## We’ll choose K=30 and visualize cluster centers

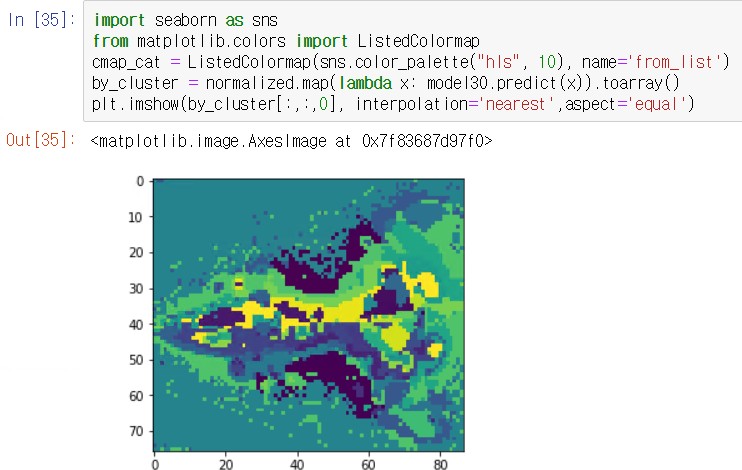




30

**Clustering Neuron Types with PySpark and Thunder**

## Plot the brain images themselves with the voxels colored according to their assigned cluster



31