Python for Data Science

Gopi Subramanian

Warm up

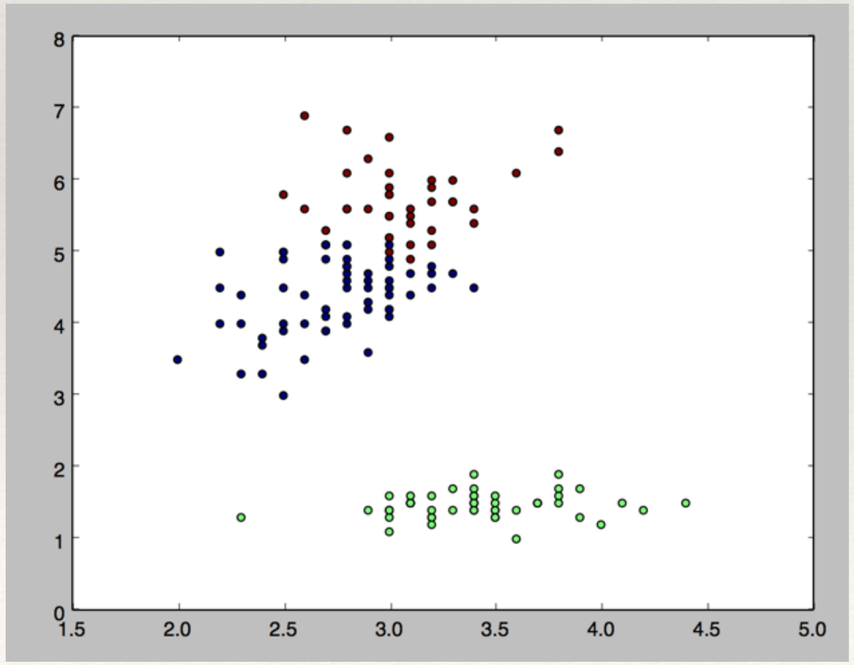
Nearest Neighbour

Compare Sentence
Jaccard's distance

```
"""
A Pure Python Nearest Neighbour
Algorithm

Mar-07-2016
Gopi Subramanian
"""

# Small input
data = 'A dog chased a cat.\n \
        The cat ran away from the dog.\n \
        Cat are from tiger family.\n \
        Dog is loyal. \n \
        Cat and dog is pet.'

# Process to get words
sentences = data.split('\n')
words = [ sent.strip().lower().split(' ') for sent in sentences ]

def jacc_similarity(l1,l2):
    s1 = set(l1)
    s2 = set(l2)
    return len(s1.intersection(s2)) /(1.0 * len(s1.union(s2)))

# Get all possible pairs and their
# Similarity
for i in range(len(words)):
    for j in range(i+1,len(words)):
        print i,j,jacc_similarity(words[i], words[j])

# Find Nearset Neighbours
test = "some cat pet".split(' ')
sim = [ jacc_similarity(test,word) for word in words]

max_sim = max(sim)

# Cluster assignment
for i  in range(len(sim)):
    if sim[i] == max_sim:
        print words[i], sim[i]
```

Unsupervised Learning

K-Means Clustering




```
"""
A simple python script to introduce
numerical libraries,

    * scikit-learn,
    * numpy
    * matplotlib

1. Load iris dataset
2. Perform KMeans
3. Plot output

Mar-07-2016
Gopi Subramanian
"""

# Load Libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import datasets

# Let us use Iris dataset
iris = datasets.load_iris()
X = iris.data
Y = iris.target

# Fit KMeans
model = KMeans(n_clusters = 3)
model.fit(X)

# Plot output
fig = plt.figure(1)
plt.clf()
plt.scatter(X[:,1],X[:,2],c = model.labels_)
plt.show()

```

Supervised Learning

Decision tree classifier

A Decision Tree Classifier

Accuracy Metrics to evaluate the classifier

```
"""
A simple python script to introduce
numerical libraries,

    * scikit-learn,
    * numpy
    * matplotlib

1. Load iris dataset
2. Perform Classification

Mar-07-2016
Gopi Subramanian
"""

# Load Libraries
import numpy as np
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score


# Let us use Iris dataset
iris = datasets.load_iris()
x = iris.data
y = iris.target

# Build a classifier
estimator = DecisionTreeClassifier()
estimator.fit(x,y)
predicted_y = estimator.predict(x)

# Find model accuracy
print "Model accuracy = %0.2f"%(accuracy_score(y,predicted_y) * 100) + "%\n"

```

Libraries

Library	Features
Numpy	N Dimensional Array Objects
Matplotlib	Graph Library
Scikit-Learn	Machine Learning Algorithms
Pandas	R Like Data Frame for Data Analysis
NetworkX	Graph Mining
NLTK	Text Processing
Simulation	SimPy
Disco	Light weight Distributed Computing (Map Reduce)
Theano	Neural Networks / Deep Learning

Numpy

- Numerical Python
- Enriches Python Data Structures with multi-dim arrays and matrices
- Scipy` - A companion library with function minimisation and fourier transforms functions.
- Python when combined with Numpy, Scipy and Matplotlib is a good alternative to MATLAB or OCTAVE.
- http://www.python-course.eu/numpy.php, A good tutorial on bumpy

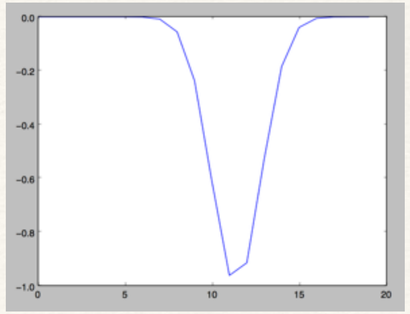
Function Minimisation

Scipy' Minimize scalar function

An exponential function

-exp^(-(x-.7)^2)

```
0.699999999784
  fun: -1.0
 nfev: 10
  nit: 9
    x: 0.6999999997839409
```

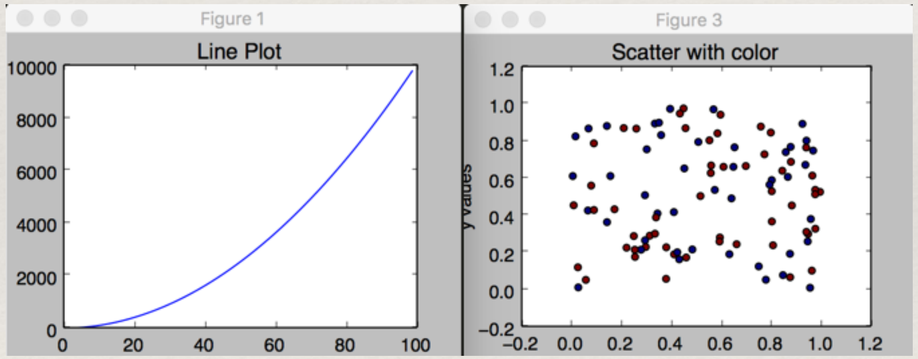


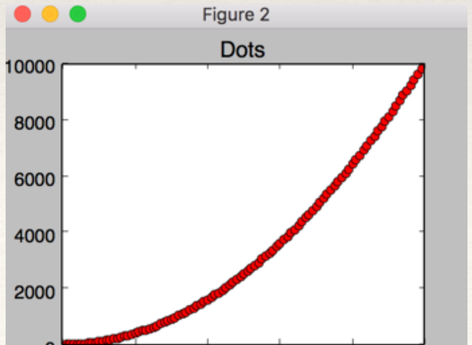

```
"""
Function minimiztion using
scipy

Mar-07-2016
Gopi Subramanian
"""

from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt

# Define a function
def afunction(x):
    return -np.exp(-(x-.7)**2)

# Generate some input
vals = np.arange(-5,5,0.5)
fvals =[afunction(x) for x in vals]

# Plot the function
plt.figure(2)
plt.plot(fvals)

plt.show()

print optimize.brent(afunction)

#Alternatively
print optimize.minimize_scalar(afunction)
```

Matplotlib

- 2D Plotting Library
- Plots, Histograms, Scatterplots, bar graphs, error charts
- http://matplotlib.org
- Pyplot’ collection of command style functions
- Pylab’ combines numpy’ and pyplot'

Matplotlib






```
"""
Introducing Matplotlib
Mar-07-2016
Gopi Subramanian
"""

import numpy as np
import matplotlib.pyplot as plt

plt.close('all')
# Sample x y data for line and simple dot plots
x = np.arange(1,100,dtype=float)
y = np.array([np.power(xx,2) for xx in x])

# Line Plot
plt.figure(1)
plt.plot(x,y)
plt.xlabel('x values')
plt.ylabel('y values')
plt.title('Line Plot')

# Dot plot
plt.figure(2)
plt.plot(x,y,'or')
plt.xlabel('x values')
plt.ylabel('y values')
plt.title('Dots')

# Sample x,y data for scatter plot
x = np.random.uniform(size=100)
y = np.random.uniform(size=100)
labels = np.random.randint(2,size=100)

# Scatter plot
plt.figure(3)
plt.scatter(x,y,c=labels)
plt.xlabel('x values')
plt.ylabel('y values')
plt.title('Scatter with color')

plt.show()
```

Scikit-Learn

- Python Machine Learning Library
- http://scikit-learn.org/stable/
- Leverages Numpy, Scipy and matplotlib
- Range of Machine Learning Algorithms.
- Easily extendable. Very Clean API

Scikit-Learn API

- Every estimator object should have the following functions
  - fit (X, Y) for classifiers, fit(X) for clustering
  - predict(X)
  - fit_predict(X,Y)

```
model = LinearRegression(normalize=True,fit_intercept=True)
model.fit(x,y)
y_p = model.predict(x)
```

```
model = Ridge(normalize=True,alpha=0.015)
model.fit(x,y)
y_p = model.predict(x)
```

Train / Test Split

- Split data into train and test
- Stratified split of data by Y variable

```
"""
Split data as train and test

Mar-07-2016
Gopi Subramanian
"""

# Load Libraries
import numpy as np
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.crossvalidation import train_test_split


# Let us use Iris dataset
iris = datasets.load_iris()
x = iris.data
y = iris.target

# Stack x and y togetehr
input_dataset = np.colstack([x,y])
train,test = train_test_split(input_dataset,test_size = 0.2)

# Perform Stratified Split data by class variable
stratified_split = StratifiedShuffleSplit(y,test_size = 0.2, n_iter = 1)

for train_indx,test_indx in stratified_split:
    train = input_dataset[train_indx]
    test =  input_dataset[test_indx]

```

More.....

- Cross Validation
- Parameter Search - Grid Search and Random Search
- Data Sets - Iris, Boston Housing, Create Data Sets for classification and clustering
- Evaluation Metrics - Accuracy, Root Mean Square….
- Stochastic Gradient Descent, Radom Projections….

Text Mining Using Python

- NLTK - Natural Language Toolkit
- http://www.nltk.org
- NLTK plugins for Scikit-Learn

RAKE - Rapid Keyphrase extraction

- RAKE is an extremely efficient keyword extraction algorithm and operates on individual documents.
  - Its language and domain independent.
- Given a document, stop word list and a list of phrase delimiters, RAKE extracts candidate phrases.
- The next step is to find out the frequency of the individual words in these phrases.
  - Frequency is the count of occurrence of the word.
- Find Degree of a word is sum of length of all the phrases where the word occurs.
- Finally scoring for each word is done by, degree(word)/ freq(word)
- Phrase scores are calculated by sum of individual word scores in that phrase.
  - Phrase with very large scores are considered to be keyphrases for the document.

- https://github.com/subramgo/RAKE

Text Search Engine

- https://github.com/subramgo/Vritti
- Word Associations
- Document Clustering - Non Negative Matrix Factorisation
- Collocations
- Uses PyLucene - Document Index

Sentiment Mining

http://sentiment.christopherpotts.net/index.html#data

A Nice Tutorial and Several Python implementations by Christopher Potts from Stanford

Simulation Using Python

- SimPy
- http://simpy.readthedocs.org/en/latest/index.html
- Let us see a simple example
- Say a new store is opened, we want to simulate how many checkout counters should be open at any point in time. Since we don't have historical data we simulate.

```
"""
Simulating a Super Market Aisle

Let us say we want to decide how many ailes should be
functional in a new super market. Since it is a
new shop, we dont have historic information. We simulate
and check out.


Mar-09-2016
Gopi Subramanian
"""


import simpy
import random
import time


time_spent_at_counter = []

class Counter(object):
    def __init__(self,env,no_counters,scan_time):
        self.env = env
        self.counters =  simpy.Resource(env, no_counters)
        self.item_scan_time = scan_time

    def checkout(self, customer, no_items):
        yield self.env.timeout(no_items * self.item_scan_time)

def customer(env, name, counter, no_items):
    #total_customers+=1
    with counter.counters.request() as request:
        yield request
        enter = env.now
        yield env.process(counter.checkout(name, no_items))
        leaves = env.now
        #print '%s Enters at %.2f leaves counter at %.2f'%(name,enter,leaves)
        elapsed = leaves - enter
        time_spent_at_counter.append(elapsed)

```

```
def setup(env,no_counters, scan_time):
    counters = Counter(env, no_counters, scan_time)
    # Initial customers
    for i in range(10):
        no_items = random.randint(5,35)
        env.process(customer(env, 'Customer %d' % i, counters, no_items))
    hour = 0
    while True:
        # Every hour
        yield env.timeout(random.randint(48,62))
        hour = hour + 1
        no_customers = random.randint(30,200)
        i += 1
        start = i
        for i in range(start,start+no_customers):
            no_items = random.randint(5,35)
            env.process(customer(env, 'Customer %d' % i, counters,no_items))
        print  'Hour %d, Number of Customers %d, %.3f Average minutes spent by a customer at counter ' \
```

```
No_counters             = 2
# Time it takes to scan an item
item_scan_time          = 0.3

env = simpy.Environment()
env.process(setup(env, No_counters, item_scan_time))

# Simulation in minutes
# 8 hours
env.run(until=480)
print  '%.3f Average minutes spent by a customer at counter ' \
%(sum(time_spent_at_counter) / (1.0*len(time_spent_at_counter)))

```

```
Hour 1, Number of Customers 190, 6.960 Average minutes spent by a customer at counter
Hour 2, Number of Customers 50, 7.471 Average minutes spent by a customer at counter
Hour 3, Number of Customers 67, 7.425 Average minutes spent by a customer at counter
Hour 4, Number of Customers 117, 6.750 Average minutes spent by a customer at counter
Hour 5, Number of Customers 133, 6.829 Average minutes spent by a customer at counter
Hour 6, Number of Customers 110, 6.910 Average minutes spent by a customer at counter
Hour 7, Number of Customers 98, 6.519 Average minutes spent by a customer at counter
Hour 8, Number of Customers 63, 6.426 Average minutes spent by a customer at counter
6.391 Average minutes spent by a customer at counter
```

Thank You….

- Code and Presentation
- https://github.com/subramgo/Py4DS