Welcome to CSCI4155/CSCI6505
Machine Learning 2017
Administrivia

• Instructor: Dr. Thomas Trappenberg
• Email: tt@cs.dal.ca
• Meeting Times:
  • Lectures in LSC-COMMON AREA C244
  • TR 1305-1425
  • Lab in SIR JAMES DUNN 304:
    • W 1635-1725
• Office hours: Write email
• Course Website:
  • All materials including manuscripts will found here.
Evaluation Criteria

• Evaluation Criteria (CSCI4155)
  • 1. Assignments (50%)
    • Late assignments will be discounted by 10% per day.
    • Assignments must be submitted electronically on Brightspace.
    • No collaboration is permitted on the assignments.
    • All assignments will be checked with the Rubber Gasket plagiarism detection software.
  • 2. Midterm Exam (20%)
    • To be held during class (Oct 12).
  • 3. Final Exam (30%)
    • To be held during class (Nov 30).
    • Will cover all material in the course.

• Evaluation Criteria (CSCI6505)
  • 1. Assignments (30%)
    • Late assignments will be discounted by 10% per day.
    • Assignments must be submitted electronically on Brightspace.
    • No collaboration is permitted on the assignments.
    • All assignments will be checked with the Rubber Gasket plagiarism detection software.
  • 2. Presentation (20%)
    • To be held during tutorial time in the second half of the course.
  • 3. Midterm Exam (20%)
    • To be held during class (Oct 12).
  • 4. Final Exam (30%)
    • To be held during class (Nov 30).
    • Will cover all material in the course.
Submissions

• You must submit your assignments on Brightspace
  • [https://dal.brightspace.com](https://dal.brightspace.com)
  • Look for the course space for CSCI4155 or CSCI6505

• You will need Dal Net ID and password to log in
  • If you have any questions, contact the CS Help Desk in the Goldberg CS Building, or email them at cshelp@cs.dal.ca
NOT SURE IF PSEUDO CODE
OR PYTHON
Install Python, sklearn, tensorflow

• You need to install the Python programming environment (Version 3.5 or higher). Make sure your installation includes Numpy, Matplotlib, Spyder, sklearn, tensorflow, and Lea.
  • On Windows we recommend WinPython which should include everything except Lea.
  • On Macs we recommend Anaconda which includes all but tensorflow and Lea.
  • Please consult the CS helpdesk if you have problems with the installation (Goldberg CS Building, cshelp@cs.dal.ca).
Check installed modules and versions

• Check installed modules:
  import sys
  sys.version
  “module name” in sys.module
  Module name: time, numpy, scipy, matplotlib, pandas, sklearn, ggplot, bokeh, seaborn, altair, holoviews
  • Or import module

• Check Version:
  import module
  module.__version__
Control Flow and functions

• if/elif/else
  If conditional statement :
     do something

• for/range
  for i in range(n):
     do something

• while/break/continue
  while conditional statement :
     do something
   If sub-conditional statement :
      break/continue

• Conditional statements: ==,!=,>,<,<=, >=, and, or

• Defining function
  def function_name(arg1 , arg2, ...):
     Do something
     return result1, result2, ...
Basic data types in python

- integer, float, complex, Booleans, string
- Type(), %whos, ?,
- useful operations: =,+,-, *, /, %, **
- Containers:
  - Lists (list=[1,2,3] or list = ['a','b','c','d','e'])
    - 0 based indexing: list[-1], list[-2]
    - Slicing syntax: list[start:stop:stride] (start <= i < stop)
    - Discovering methods: list.<tab>
  - Arrays vs Matrixes (numpy)
    - dtype, ndim, size, shape, vstack(), hstack(), >, <, ==, :,transpose(), T, dot(),@, nonzero(), arange(), zeros(), ones(), random.rand(), unique(), reshape(), sort, squeeze, max, min, mean, std, sum, sqrt, exp, floor, ceil, single, int, float, randn, seed, savetxt, loadtxt, linalg, fft, ifft, linspace, meshgrid...
Other useful functions or modules

- itertools.combinations()
- time.clock()
- matplotlib.pyplot

  - The basic steps to creating plots with matplotlib are:
    1. Prepare data
    2. Create plot (figure)
    3. Plot (subplot, plot, bar, errorbar, hist, scatter, imshow, pcolor, ...)
    4. Customize plot (subplots_adjust, legend, axis, colorbar, annotate, set_xlim, set_ylim, title, xlabel, ylabel, set, ...)
    5. Save plot (savefig)
    6. Show plot (show)
NOT SURE IF I AM A GOOD PROGRAMMING

OR GOOD AT GOOGLING
Example 1:

*Write a program to calculate Y:*

\[
y_0 = x_0 \times a + b \\
y_1 = x_1 \times a + b \\
y_2 = x_2 \times a + b \\
y_3 = x_3 \times a + b \\
\vdots \\
y_{n-1} = x_{n-1} \times a + b
\]

Where \(a=4\), \(b=5\), \(n=10\)
a=3; b=4, n=10
params = np.array([a,b])
x=np.array( np.arange (1,n+1) ) or x=np.arange(1,n+1)
x=np.vstack( (x,np.ones( n ) ) )
y = np.dot(params,x ) or y = np.dot(x.T,params)
## Using python libraries

### Linear regression Using numpy
```python
import matplotlib.pyplot as plt
import numpy as np

hsize=np.array([937, 1150, 1170, 1290, 1275, 1410, 1550, 1730, 1910])
price=np.array([187, 222, 330, 310, 290, 440, 600, 550, 600])
slope, intercept = np.polyfit(hsize,price,1)

# Plot outputs
plt.scatter(hsize, price, label='Original data', color='black')
plt.plot(hsize, slope*hsize+intercept, label='Fitted line', color='blue', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.legend()
plt.title('Linear regression Using numpy')
plt.show()
```

### Linear regression Using scipy
```python
import matplotlib.pyplot as plt
from scipy import stats
import numpy as np

t = np.array([937, 1150, 1170, 1290, 1275, 1410, 1550, 1730, 1910])
price=np.array([187, 222, 330, 310, 290, 440, 600, 550, 600])

slope, intercept, r_value, p_value, std_err = stats.linregress(t, price)

# Plot outputs
plt.scatter(t, price, label='Original data', color='black')
plt.plot(t, slope*t+intercept, label='Fitted line', color='blue', linewidth=3)
plt.xticks()
plt.yticks()
plt.legend()
plt.title('Linear regression Using scipy')
plt.show()
```

### Linear regression Using sklearn
```python
import matplotlib.pyplot as plt
from sklearn import linear_model
import numpy as np

t = np.array([937, 1150, 1170, 1290, 1275, 1410, 1550, 1730, 1910])[:, np.newaxis]
price=np.array([187, 222, 330, 310, 290, 440, 600, 550, 600])[:, np.newaxis]

regr = linear_model.LinearRegression()
regr.fit(t, price)

# Plot outputs
plt.scatter(t, price, label='Original data', color='black')
plt.plot(t, regr.predict(t), label='Fitted line', color='blue', linewidth=3)
plt.xticks()
plt.yticks()
plt.legend()
plt.title('Linear regression Using sklearn')
plt.show()
```