MIDS W261 Fall 2016 Homework Week 1

Shih Yu Chang
W261-2
sychang@ischool.berkeley.edu
Sep. 3, 2016

I come from Mountain View, CA and am working at Oracle as sfotware enginner whorking for cloud networking. I began my MIDS from 2016 Spring and expect to graduate at 2017 Summer. I wish to learn applying ML knolwedge in analyzing Big Data.

HW1.0.0.

Define big data. Provide an example of a big data problem in your domain of expertise.

"Big Data" is a domain that involves working with data at volumes, velocities and varieties that are exceed the boundaries of "traditional" vertically scaled solutions and systems. In dealing with Big Data, people require new tools and techniques, e.g., using parallel processing and coordinating large networking.

We can apply Big Data at traffic control in big city. Some cities in China have seen their economy develop rapidly in recent years. As a result, local traffic has become much heavier, leading to an increase in vehicle accidents and traffic violations.The city government needed better ways to monitor and manage local traffic to provide better transportation services to the public. Taking a data-driven approach to the problem, the local traffic department installed more than thoudands digital monitoring devices in the city's key checkpoints. These devices capture images and video data continuously. The traffic department now faces a terabyte of data each month. The increasing amount of traffic data now poses challenges in the city's ability to effectively manage traffic. Big Data technique can be applied here to store, retrival and anlyze data more efficiently.

HW1.0.3. Bias Variance

What is bias-variance decomposition in the context machine learning? How is it used in machine learning?

The bias–variance decomposition is a method of analyzing a learning algorithm's expected generalization error with respect to a particular problem as a sum of three components, the bias, variance, and a quantity called the irreducible error, resulting from noise in the problem itself.

This tradeoff applies to all forms of supervised based machine learning: classification, function fitting (regression),and structured output learning. It has also been invoked to explain the effectiveness of heuristics in AI learning.

1. HW1.1 WordCount using a single thread

Back to Table of Contents

Write a program called alice_words.py that creates a text file named alice_words.txt containing an alphabetical listing of all the words, and the number of times each occurs, in the text version of Alice's Adventures in Wonderland. (You can obtain a free plain text version of the book, along with many others, from here The first 10 lines of your output file should look something like this (the counts are not totally precise):

In [44]:
```
# check where is the current directory and change if necessary using someth
ing like: %cd W261MasterDir
!pwd
```

```
/home/shihyu/HW1
```

In [2]:
```
#let's organize our homeworks into subfolders by week at least (and possibl
y by problem)
!mkdir HW1
```

```
mkdir: cannot create directory ‘HW1’: File exists
```

In [3]:
```
# notice the use of % in the following magic command
%cd HW1
```

```
/home/shihyu/HW1
```

In [4]:
```
!curl 'http://www.gutenberg.org/cache/epub/11/pg11.txt' -o alicesTExtFilena
me.txt
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Cur
rent
                                 Dload  Upload   Total   Spent    Left  Spe
ed
100  163k  100  163k    0     0   158k      0  0:00:01  0:00:01 --:--:--  1
99k
```

In [74]:
```
#display the first few lines
!head alicesTExtFilename.txt
```

In [6]:
```
#example of a regular expression to detect words in a string.
import re
line = """ 0017.2000-01-17.beck 0        global risk management operations
        " congratulations, sally!!!  kk  ----------------------forwarded by
 kathy kokas/corp/enron on 01/17/2000  08:08 pm---------------------------
 from: rick causey 01/17/2000 06:04 pm  sent by: enron announcements  to: a
ll enron worldwide  cc:  subject: global risk management operations  recogn
izing enron , s increasing worldwide presence in the wholesale energy  busi
ness and the need to insure outstanding internal controls for all of our  r
isk management activities, regardless of location, a global risk management
  operations function has been created under the direction of sally w. beck
,  vice president. in this role, sally will report to rick causey, executiv
e  vice president and chief accounting officer.  sally , s responsibilities
 with regard to global risk management operations  will mirror those of oth
er recently created enron global functions. in this  role, sally will work
closely with all enron geographic regions and wholesale  companies to insur
e that each entity receives individualized regional support  while also foc
using on the following global responsibilities:  1. enhance communication a
mong risk management operations professionals.  2. assure the proliferation
 of best operational practices around the globe.  3. facilitate the allocat
ion of human resources.  4. provide training for risk management operations
 personnel.  5. coordinate user requirements for shared operational systems
.  6. oversee the creation of a global internal control audit plan for risk
  management activities.  7. establish procedures for opening new risk mana
gement operations offices  and create key benchmarks for measuring on-going
 risk controls.  each regional operations team will continue its direct rep
orting relationship  within its business unit, and will collaborate with sa
lly in the delivery of  these critical items. the houston-based risk manage
ment operations team under  sue frusco , s leadership, which currently supp
orts risk management activities  for south america and australia, will also
 report directly to sally.  sally retains her role as vice president of ene
rgy operations for enron  north america, reporting to the ena office of the
 chairman. she has been in  her current role over energy operations since 1
997, where she manages risk  consolidation and reporting, risk management a
dministration, physical product  delivery, confirmations and cash managemen
t for ena , s physical commodity  trading, energy derivatives trading and f
inancial products trading.  sally has been with enron since 1992, when she
joined the company as a  manager in global credit. prior to joining enron,
sally had four years  experience as a commercial banker and spent seven yea
rs as a registered  securities principal with a regional investment banking
 firm. she also owned  and managed a retail business for several years.  pl
ease join me in supporting sally in this additional coordination role for
global risk management operations."""
re.findall(r'[a-z]+', line.lower()) [0:10]
```

Out[6]:
```
['beck',
 'global',
 'risk',
 'management',
 'operations',
 'congratulations',
 'sally',
 'kk',
 'forwarded',
 'by']
```

Dictionaries are a good way to keep track of word counts

wordCounts={} defaultdict are slightly more effectice way of doing word counting

One way to do word counting but not best. A defaultdict is like a regular dictionary, except that when you try to look up a key it doesn't contain, it first adds a value for it using a zero-argument function you provided when you created it. In order to use defaultdicts, you have to import them

In [7]:
```
# Here is an example of wordcounting with a defaultdict (dictionary structu
re with a nice
# default behaviours when a key does not exist in the dictionary
import re
from collections import defaultdict

line = """ 0017.2000-01-17.beck 0        global risk management operations
        " congratulations, sally!!!  kk  ----------------------forwarded by
 kathy kokas/corp/enron on 01/17/2000  08:08 pm---------------------------
 from: rick causey 01/17/2000 06:04 pm  sent by: enron announcements  to: a
ll enron worldwide  cc:  subject: global risk management operations  recogn
izing enron , s increasing worldwide presence in the wholesale energy  busi
ness and the need to insure outstanding internal controls for all of our  r
isk management activities, regardless of location, a global risk management
  operations function has been created under the direction of sally w. beck
,  vice president. in this role, sally will report to rick causey, executiv
e  vice president and chief accounting officer.  sally , s responsibilities
 with regard to global risk management operations  will mirror those of oth
er recently created enron global functions. in this  role, sally will work
closely with all enron geographic regions and wholesale  companies to insur
e that each entity receives individualized regional support  while also foc
using on the following global responsibilities:  1. enhance communication a
mong risk management operations professionals.  2. assure the proliferation
 of best operational practices around the globe.  3. facilitate the allocat
ion of human resources.  4. provide training for risk management operations
 personnel.  5. coordinate user requirements for shared operational systems
.  6. oversee the creation of a global internal control audit plan for risk
  management activities.  7. establish procedures for opening new risk mana
gement operations offices  and create key benchmarks for measuring on-going
 risk controls.  each regional operations team will continue its direct rep
orting relationship  within its business unit, and will collaborate with sa
lly in the delivery of  these critical items. the houston-based risk manage
ment operations team under  sue frusco , s leadership, which currently supp
orts risk management activities  for south america and australia, will also
 report directly to sally.  sally retains her role as vice president of ene
rgy operations for enron  north america, reporting to the ena office of the
 chairman. she has been in  her current role over energy operations since 1
997, where she manages risk  consolidation and reporting, risk management a
dministration, physical product  delivery, confirmations and cash managemen
t for ena , s physical commodity  trading, energy derivatives trading and f
inancial products trading.  sally has been with enron since 1992, when she
joined the company as a  manager in global credit. prior to joining enron,
sally had four years  experience as a commercial banker and spent seven yea
rs as a registered  securities principal with a regional investment banking
 firm. she also owned  and managed a retail business for several years.  pl
ease join me in supporting sally in this additional coordination role for
global risk management operations."""
wordCounts=defaultdict(int)
for word in re.findall(r'[a-z]+', line.lower()):
    #if word in ["a"]:
        #print word,"\n"
    wordCounts[word] += 1
for key in sorted(wordCounts)[0:10]:
    print (key, wordCounts[key])
```

```
a 7
accounting 1
activities 3
additional 1
administration 1
all 3
allocation 1
also 3
america 2
among 1
```

In [79]:
```
#HW1.1.1 How many times does the word alice occur in the book?
import re
'''
import csv
from collections import Counter
counter = Counter()
with open("alicesTExtFilename.txt") as f:
    reader = csv.reader(f)
    for row in reader:
        counter.update(row)
print(counter['it'])


%%%%%

file=open("alicesTExtFilename.txt","r+")
wordcount={}
for word in file.read().split():
    if word not in wordcount:
        wordcount[word] = 1
    else:
        wordcount[word] += 1
print (word,wordcount)
file.close();
'''
wanted = "alice"
cnt = Counter()
words = re.findall('\w+', open('alicesTExtFilename.txt').read().lower())
for word in words:
    if word in wanted:
        cnt[word] += 1
print(cnt)


# There are 403 Alice in this textbook
```

```
Counter({'a': 690, 'i': 543, 'alice': 403, 'e': 29, 'c': 6})
```

1. HW1.2 Command Line Map Reduce Framework

Back to Table of Contents

Read through the provided mapreduce shell script (pWordCount.sh) provided below and all of its comments. When you are comfortable with their purpose and function, respond to the remaining homework questions below. Run the shell without any arguments.

In [51]:
```
%%writefile pWordCount.sh
#!/bin/bash
## pWordCount.sh
## Author: James G. Shanahan
## Usage: pWordCount.sh m wordlist testFile.txt
## Input:
##       m = number of processes (maps), e.g., 4
##       wordlist = a space-separated list of words in quotes, e.g., "the a
nd of"
##       inputFile = a text input file
##
## Instructions: Read this script and its comments closely.
##               Do your best to understand the purpose of each command,
##               and focus on how arguments are supplied to mapper.py/reduc
er.py,
##               as this will determine how the python scripts take input.
##               When you are comfortable with the unix code below,
##               answer the questions on the LMS for HW1 about the starter
code.



if [ $# -eq 0 ]
  then
    echo "No arguments supplied"
    echo "To run use"
    echo "     pWordCount.sh m wordlist inputFile"
    echo "Input:"
    echo "      number of processes (maps), e.g., 4"
    echo "      wordlist = a space-separated list of words in quotes, e.g.,
 'the and of'"
    echo "      inputFile = a text input file"
    exit
fi


## collect user input
m=$1 ## the number of parallel processes (maps) to run

wordlist=$2 ## if set to "*", then all words are used

## a text file
data="alicesTExtFilename.txt"

## 'wc' determines the number of lines in the data
## 'perl -pe' regex strips the piped wc output to a number
linesindata=`wc -l $data | perl -pe 's/^.*?(\d+).*?$/$1/'`

## determine the lines per chunk for the desired number of processes
linesinchunk=`echo "$linesindata/$m+1" | bc`

## split the original file into chunks by line
split -l $linesinchunk $data $data.chunk.

## assign python mappers (mapper.py) to the chunks of data
## and emit their output to temporary files
for datachunk in $data.chunk.*; do
    ## feed word list to the python mapper here and redirect STDOUT to a te
mporary file on disk
    ####
    ####
    ./mapper.py $datachunk "$wordlist" > $datachunk.counts &
    ####
```

```
Overwriting pWordCount.sh
```

In [52]:
```
!cat pWordCount.sh
```

```
#!/bin/bash
## pWordCount.sh
## Author: James G. Shanahan
## Usage: pWordCount.sh m wordlist testFile.txt
## Input:
##       m = number of processes (maps), e.g., 4
##       wordlist = a space-separated list of words in quotes, e.g., "the a
nd of"
##       inputFile = a text input file
##
## Instructions: Read this script and its comments closely.
##               Do your best to understand the purpose of each command,
##               and focus on how arguments are supplied to mapper.py/reduc
er.py,
##               as this will determine how the python scripts take input.
##               When you are comfortable with the unix code below,
##               answer the questions on the LMS for HW1 about the starter
code.



if [ $# -eq 0 ]
  then
    echo "No arguments supplied"
    echo "To run use"
    echo "     pWordCount.sh m wordlist inputFile"
    echo "Input:"
    echo "      number of processes (maps), e.g., 4"
    echo "      wordlist = a space-separated list of words in quotes, e.g.,
 'the and of'"
    echo "      inputFile = a text input file"
    exit
fi


## collect user input
m=$1 ## the number of parallel processes (maps) to run

wordlist=$2 ## if set to "*", then all words are used

## a text file
data="alicesTExtFilename.txt"

## 'wc' determines the number of lines in the data
## 'perl -pe' regex strips the piped wc output to a number
linesindata=`wc -l $data | perl -pe 's/^.*?(\d+).*?$/$1/'`

## determine the lines per chunk for the desired number of processes
linesinchunk=`echo "$linesindata/$m+1" | bc`

## split the original file into chunks by line
split -l $linesinchunk $data $data.chunk.

## assign python mappers (mapper.py) to the chunks of data
## and emit their output to temporary files
for datachunk in $data.chunk.*; do
    ## feed word list to the python mapper here and redirect STDOUT to a te
mporary file on disk
    ####
    ####
    ./mapper.py $datachunk "$wordlist" > $datachunk.counts &
    ####
    ####
```

In [53]:
```
!chmod a+x pWordCount.sh
#run the shell without any arguments
!./pWordCount.sh 4 "Alice"
```

In [12]:
```
##3. HW1.3 WordCount via Command Line Map Reduce Framework ##

## MAP
```

In [54]:
```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Given a file and list of words, read lines and count occurrences of word
s


import sys
import re

## Lines in the file have 4 fields:
## ID \t SPAM \t SUBJECT \t CONTENT \n
WORD_RE = re.compile(r"[\w']+")

filename = sys.argv[1]

## Words in the word list are space delimited
wordlist = sys.argv[2].lower().split(' ')
counts = {}


with open (filename, "rU") as myfile:
    for text in myfile.readlines():
        for word in WORD_RE.findall(text):
            if word.lower() in wordlist:
                try:
                    counts[word.lower()] += 1
                except:
                    counts[word.lower()] = 1

for word in counts:
    sys.stdout.write('{0}\t{1}\n'.format(word, counts[word]))
```

```
Overwriting mapper.py
```

In [55]:
```
# Set the execution permissions of the Python script
!chmod a+x mapper.py
```

In [15]:
```
## Reduce
```

In [56]:
```
%%writefile reducer.py
#!/usr/bin/python2
import sys
counts = {}

for intermediate_file in sys.argv:
    with open(intermediate_file, 'rU') as infile:
        # intermediate files are word <tab> count per line
        for line in infile.readlines():
            word_count = line.split('\t')
            if len(word_count) == 2:
                try:
                    counts[word_count[0]] += int(word_count[1])
                except KeyError:
                    counts[word_count[0]] = int(word_count[1])
for word in counts:
    sys.stdout.write('{0}\t{1}\n'.format(word, counts[word]))
```

```
Overwriting reducer.py
```

In [59]:
```
# Set the execution permissions of the Python script
!chmod a+x reducer.py

# Set the execution permissions of the pWordCount.sh bash shell script
!chmod a+x pWordCount.sh

!./pWordCount.sh 4 "Alice the we"
!cat *.output

### For example, I input "Alice", "the",  "we", it sorted as alphabetical o
rder
```

```
alice   385
the     1797
we      28
```

In [29]:
```
## Below is  HW1.4
```

In [60]:
```
%%writefile pWordCount_n.sh
#!/bin/bash
## pWordCount.sh
## Author: James G. Shanahan
## Usage: pWordCount.sh m wordlist testFile.txt
## Input:
##       m = number of processes (maps), e.g., 4
##       wordlist = a space-separated list of words in quotes, e.g., "the a
nd of"
##       inputFile = a text input file
##
## Instructions: Read this script and its comments closely.
##               Do your best to understand the purpose of each command,
##               and focus on how arguments are supplied to mapper.py/reduc
er.py,
##               as this will determine how the python scripts take input.
##               When you are comfortable with the unix code below,
##               answer the questions on the LMS for HW1 about the starter
code.



if [ $# -eq 0 ]
  then
    echo "No arguments supplied"
    echo "To run use"
    echo "     pWordCount.sh m wordlist inputFile"
    echo "Input:"
    echo "      number of processes (maps), e.g., 4"
    echo "      wordlist = a space-separated list of words in quotes, e.g.,
 'the and of'"
    echo "      inputFile = a text input file"
    exit
fi


## collect user input
m=$1 ## the number of parallel processes (maps) to run

wordlist=$2 ## if set to "*", then all words are used

## a text file
data="alicesTExtFilename.txt"

## 'wc' determines the number of lines in the data
## 'perl -pe' regex strips the piped wc output to a number
linesindata=`wc -l $data | perl -pe 's/^.*?(\d+).*?$/$1/'`

## determine the lines per chunk for the desired number of processes
linesinchunk=`echo "$linesindata/$m+1" | bc`

## split the original file into chunks by line
split -l $linesinchunk $data $data.chunk.

## assign python mappers (mapper.py) to the chunks of data
## and emit their output to temporary files
for datachunk in $data.chunk.*; do
    ## feed word list to the python mapper here and redirect STDOUT to a te
mporary file on disk
    ####
    ####
    ./mapper_n.py $datachunk "$wordlist" > $datachunk.counts &
    ####
```

```
Overwriting pWordCount_n.sh
```

In [61]:
```
!chmod a+x pWordCount_n.sh
```

In [62]:
```
%%writefile mapper_n.py
#!/usr/bin/python
## mapper.py
## Given a file and list of words, read lines and count occurrences of word
s


import sys
import re

## Lines in the file have 4 fields:
## ID \t SPAM \t SUBJECT \t CONTENT \n
WORD_RE = re.compile(r"[\w']+")

filename = sys.argv[1]

## Words in the word list are space delimited
wordlist = sys.argv[2].split(' ')
counts = {}


with open (filename, "rU") as myfile:
    for text in myfile.readlines():
        for word in WORD_RE.findall(text):
            if word in wordlist:
                try:
                    counts[word] += 1
                except:
                    counts[word] = 1

for word in counts:
    sys.stdout.write('{0}\t{1}\n'.format(word, counts[word]))
```

```
Overwriting mapper_n.py
```

In [63]:
```
# Set the execution permissions of the Python script
!chmod a+x mapper_n.py
```

In [64]:
```
%%writefile reducer_n.py
#!/usr/bin/python2
import sys
counts_U = {}
counts_L = {}

for intermediate_file in sys.argv:
    with open(intermediate_file, 'rU') as infile:
        # intermediate files are word <tab> count per line
        for line in infile.readlines():
            word_count = line.split('\t')
            # starting with an uppercase
            word = word_count[0]
            if len(word_count) == 2 and (word[0] == word[0].upper()):
                try:
                    counts_U[word_count[0]] += int(word_count[1])
                except KeyError:
                    counts_U[word_count[0]] = int(word_count[1])

            elif len(word_count) == 2:
                try:
                    counts_L[word_count[0]] += int(word_count[1])
                except KeyError:
                    counts_L[word_count[0]] = int(word_count[1])

for word in counts_U:
    sys.stdout.write('{0}\t{1}\n'.format(word, counts_U[word]))

for word in counts_L:
    sys.stdout.write('{0}\t{1}\n'.format(word, counts_L[word]))
```

```
Overwriting reducer_n.py
```

In [70]:
```
# Set the execution permissions of the Python script
!chmod a+x reducer_n.py

# Set the execution permissions of the pWordCount.sh bash shell script
!chmod a+x pWordCount_n.sh

!./pWordCount_n.sh 4 "Alice the we The We How dog"
print("Upper Case")
!cat *.outputU

print("Lower Case")
!cat *.outputL

### For example, I input "Alice", "the",  "we", "The", "We", "How", "dog"
### they are sorted as alphabetical order for Upper case begin (saved .outp
utU file),
### and Lower case begin (saved .outputL)
```

```
Upper Case
Alice   385
How     12
The     104
We      3
Lower Case
dog     2
the     1680
we      24
```

In [ ]:
```
## This example illustrates and compares the bias-variance decomposition
## of the expected mean squared error of a single estimator against
## a bagging ensemble. In regression, the expected mean squared error of
## an estimator can be decomposed in terms of bias, variance and noise.
## On average over datasets of the regression problem, the bias term
## measures the average amount by which the predictions of the estimator
## differ from the predictions of the best possible estimator for the
## problem (i.e., the Bayes model).
## Following generated figures show that the larger the variance,
## the more sensitive are the predictions for x to small changes
## in the training set. The bias term corresponds to the difference
## between the average prediction of the estimator (in cyan) and
## the best possible model (in dark blue).
```

In [ ]:
```
#####Below example came from scikit, but we modify a little bit as using
## 4-th power instead of square error measures. But, still can observe
## varaince and bias issue.

print(__doc__)



import numpy as np
import matplotlib.pyplot as plt

from sklearn.ensemble import BaggingRegressor
from sklearn.tree import DecisionTreeRegressor

# Settings
n_repeat = 50       # Number of iterations for computing expectations
n_train = 50        # Size of the training set
n_test = 1000       # Size of the test set
noise = 0.1         # Standard deviation of the noise
np.random.seed(0)

# Change this for exploring the bias-variance decomposition of other
# estimators. This should work well for estimators with high variance (e.g.
,
# decision trees or KNN), but poorly for estimators with low variance (e.g.
,
# linear models).
estimators = [("Tree", DecisionTreeRegressor()),
              ("Bagging(Tree)", BaggingRegressor(DecisionTreeRegressor()))]

n_estimators = len(estimators)

# Generate data
def f(x):
    x = x.ravel()

    return np.exp(-x ** 2) + 1.5 * np.exp(-(x - 2) ** 2)

def generate(n_samples, noise, n_repeat=1):
    X = np.random.rand(n_samples) * 10 - 5
    X = np.sort(X)

    if n_repeat == 1:
        y = f(X) + np.random.normal(0.0, noise, n_samples)
    else:
        y = np.zeros((n_samples, n_repeat))

        for i in range(n_repeat):
            y[:, i] = f(X) + np.random.normal(0.0, noise, n_samples)

    X = X.reshape((n_samples, 1))

    return X, y

X_train = []
y_train = []

for i in range(n_repeat):
    X, y = generate(n_samples=n_train, noise=noise)
    X_train.append(X)
    y_train.append(y)

X_test, y_test = generate(n_samples=n_test, noise=noise, n_repeat=n_repeat)
```

```
Automatically created module for IPython interactive environment
```

```
/home/shihyu/anaconda3/lib/python3.5/site-packages/matplotlib/font_manager.
py:273: UserWarning: Matplotlib is building the font cache using fc-list. T
his may take a moment.
  warnings.warn('Matplotlib is building the font cache using fc-list. This
may take a moment.')
/home/shihyu/anaconda3/lib/python3.5/site-packages/matplotlib/font_manager.
py:273: UserWarning: Matplotlib is building the font cache using fc-list. T
his may take a moment.
  warnings.warn('Matplotlib is building the font cache using fc-list. This
may take a moment.')
```

```
Tree: 0.0255 (error) = 0.0003 (bias^2)  + 0.0152 (var) + 0.0098 (noise)
Bagging(Tree): 0.0196 (error) = 0.0004 (bias^2)  + 0.0092 (var) + 0.0098 (n
oise)
```