Homework 1

completed by Alejandro J. Rojas

HW1.0.0.

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

The term big data is asoociated to datasets that cannot be processed, stored and transformed using traditional applications and tools because of high volume, high velocity and high variety. By high volume, we mean datasets that not only require high storage capacity, usually beyond 1T, but also datasets that are too big to have a decent processing and thoughput time. By high velocity, we mean data that requires real-time processing with throughput speeds that can be bursty. High variety includes data that comes from different formats, some structured some that are not, that all need to be ingested and transformed to be processed. Big data is simply changing the way we used to collect and analyze data at the time that it opens opportunity to increase the scale, scope and intimacy of the analyses that we are now able to do.

The social web is a leading source of big data applications given our ability to log almost anything that the user does when interacting to an application. In my field, I've seen how online videos are increasingly the way users consume media. A video, per se, is an unstructured data item and its interactions are usually captured by leading social media platforms like Facebook, Twitter and Youtube in the form of JSON, a semi unstructured format that can capture user interactions such as likes, shares and comments. Across the internet, the amount of videos being upload and downstream is exploding making it a challenge to measure real-time, the media consuming habits of our target users. Big data can help in providing insights from all of this information so that we can better predict the taste of users visiting our site properties to serve them content they like.

HW1.0.1.

In 500 words (English or pseudo code or a combination) describe how to estimate the bias, the variance, the irreducible error for a test dataset T when using polynomial regression models of degree 1, 2,3, 4,5 are considered. How would you select a model?

For any dataset T that contains n independent variables (x1, x2, ..xn) and one dependent variable y_true, we can observe the following:

If we try to estimate y as a function of x:

```
y_pred = f(x)
```

The estimate of our function will produce an error shown as:

```
error = y_true - y_pred
```

The source of this error can be divided into three types:

```
bias
variance
irreducible error
```

Bias error is introduced by us when we try to simplify the dynamics that we observe in the data, for instace by using a linear function to estimate y.

As we try to better fit the underlying data, we can try implementing nonlinear functions.

As the order of the polynomial regression increases, our function f(x) will more closely match the underlying portion of the dataset T and consequently we reduced our bias error.

However, if we randomly applied our high-ordered polynomial f(x) to another portion of dataset T, we will find that our error will increase because we introduced variance error by overfitting the prior dataset.

So as a rule of thumb, we can say that

as the degree of the predictive polynomial function f(x) increases:

```
bias error is reduced
variance error is increased
```

the trick is to find the optimal point where the sum of these two errors are at the minimum.

Even at that point, our function(x) will still show some error that will be irreducible because it comes from imprecisions in the way data was collected or other type of noise present in the dataset T.

HW1.1.

Read through the provided control script (pNaiveBayes.sh) and all of its comments. When you are comfortable with their purpose and function, respond to the remaining homework questions below. A simple cell in the notebook with a print statmement with a "done" string will suffice here. (dont forget to include the Question Number and the quesition in the cell as a multiline comment!)

<----------End of HW1.1----------

HW1.2.

Provide a mapper/reducer pair that, when executed by pNaiveBayes.sh will determine the number of occurrences of a single, user-specified word. Examine the word “assistance” and report your results.

Map

In [2]:
```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Alejandro J. Rojas
## Description: mapper code for HW1.2-1.5

import sys
import re
count = 0
records = 0
words = 0

## collect user input
filename = sys.argv[1]
findwords = re.split(" ",sys.argv[2].lower())

with open (filename, "r") as myfile:
    for line in myfile.readlines():
        record = re.split(r'\t+', line)
        records = records + 1
        for i in range (len(record)):
            bagofwords = re.split(" ",record[i]) ### Break each ema
il records into words
            for word in bagofwords:
                words = words + 1
                for keyword in findwords:
                    if keyword in word:
                        count = count + 1       ### Add one the cou
nt of found words


##print '# of Records analized',records
##print '# of Words analized', words
##print '# of Ocurrences', count
print count
```

```
Overwriting mapper.py
```

In [3]:
```
!chmod +x mapper.py
```

Reduce

In [4]:
```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Alejandro J. Rojas
## Description: reducer code for HW1.2

import sys
import re
sum = 0

## collect user input
filenames = sys.argv[1:]
for file in filenames:
    with open (file, "r") as myfile:
        for line in myfile.readlines():
            if line.strip():
                sum = sum + int(line)            ### Add counts pr
esent on all mapper produced files

print sum
```

```
Overwriting reducer.py
```

In [5]:
```
!chmod +x reducer.py
```

Write script to file

In [6]:
```
%%writefile pNaiveBayes.sh
## pNaiveBayes.sh
## Author: Jake Ryland Williams
## Usage: pNaiveBayes.sh m wordlist
## Input:
##       m = number of processes (maps), e.g., 4
##       wordlist = a space-separated list of words in quotes,
e.g., "the and of"
##
## Instructions: Read this script and its comments closely.
##               Do your best to understand the purpose of each com
mand,
##               and focus on how arguments are supplied to mappe
r.py/reducer.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.

## collect user input
m=$1 ## the number of parallel processes (maps) to run
wordlist=$2 ## if set to "*", then all words are used

## a test set data of 100 messages
data="enronemail_1h.txt"

## the full set of data (33746 messages)
# data="enronemail.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 processe
s
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 temporary file on disk
    ####
    ####
    ./mapper.py $datachunk "$wordlist" > $datachunk.counts &
    ####
    ####
done
## wait for the mappers to finish their work
```

```
wait

## 'ls' makes a list of the temporary count files
## 'perl -pe' regex replaces line breaks with spaces
countfiles=`\ls $data.chunk.*.counts | perl -pe 's/\n/ /'`

## feed the list of countfiles to the python reducer and redirect S
TDOUT to disk
####
####
./reducer.py $countfiles > $data.output
####
####
numOfInstances=$(cat $data.output)
echo "found [$numOfInstances] [$wordlist]" ## Report  how many were
found
## clean up the data chunks and temporary count files
\rm $data.chunk.*
```

```
Overwriting pNaiveBayes.sh
```

In [7]:
```
!chmod a+x pNaiveBayes.sh
```

Run file

Usage: usage: pNaiveBayes.sh m wordlist

In [8]:
```
!./pNaiveBayes.sh 5 "assistance"
```

```
found [10] [assistance]
```

<----------End of HW1.2----------

HW1.3.

Provide a mapper/reducer pair that, when executed by pNaiveBayes.sh will classify the email messages by a single, user-specified word using the multinomial Naive Bayes Formulation. Examine the word “assistance” and report your results.

Map

In [23]:
```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Alejandro J. Rojas
## Description: mapper code for HW1.3

import sys
import re



########## Collect user input  ###############
filename = sys.argv[1]
findwords = re.split(" ",sys.argv[2].lower())




with open (filename, "r") as myfile:

    for line in myfile.readlines():
        record = re.split(r'\t+', line)                    ### Each
email is a record with 4 components
                                                           ### 1) I
D 2) Spam Truth 3) Subject 4) Content
        if len(record)==4:                                 ### Take
only complete records

            ########## Variables to collect and measure #########
            records = 0                                    ### Each
record corresponds to a unique email
            words = 0                                      ### Word
s written in all emails incluidng Subject
            spam_records, spam_words, spam_count = 0,0,0   ### Spam
email count, words in spam email, user-specified word count
            ham_records, ham_words, ham_count = 0, 0, 0    ### Same
as above but for not spam emails


            records += 1                                   ### add
one the the total sum of emails
            if int(record[1]) == 1:                        ### If t
he email is labeled as spam
                spam_records += 1                          ### add
one to the email spam count
                for i in range (2,len(record)):            ### Star
ting from Subject to the Content
                    bagofwords = re.split(" ",record[i])   ### Coll
ect all words present on each email
                    for word in bagofwords:                ### For
each word
                        words += 1                         ### add
one to the total sum of words
```

```
                        spam_words += 1                     ### add
one to the total sum of spam words
                        for keyword in findwords:           ### for
each word specified by user
                            if keyword in word:             ### If t
here's a match then
                                spam_count += 1             ### add
one to the user specified word count as spam

            else:                                           ### If e
mail is not labeled as spam
                ham_records +=1                             ### add
one to the email ham count
                for i in range (2,len(record)):             ### Star
ting from Subject to the Content
                    bagofwords = re.split(" ",record[i])    ### Coll
ect all words present on each email
                    for word in bagofwords:                 ### For
each word
                        words += 1                          ### add
one to the total sum of words
                        ham_words += 1                      ### add
one to the total sum of ham words
                        for keyword in findwords:           ### for
each word specified by user
                            if keyword in word:             ### If t
here's a match then
                                ham_count += 1              ### add
one to the user specified word count as ham

            record_id = record[0]
            truth = record[1]
            print spam_count, " ", spam_words, " ",  spam_records,
" ",  \
                  ham_count, " ", ham_words, " ",  ham_records, "
", \
                  words, " ", records, " ",  record_id, " ", truth
```

```
Overwriting mapper.py
```

In [24]:
```
!chmod +x mapper.py
```

Reduce

In [25]:
```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Alejandro J. Rojas
## Description: reducer code for HW1.3-1.4

import sys
import re
sum_spam_records, sum_spam_words, sum_spam_count = 0,0,0
sum_ham_records, sum_ham_words, sum_ham_count = 0,0,0
sum_records,sum_words = 0,0

## collect user input
filenames = sys.argv[1:]
for file in filenames:
    with open (file, "r") as myfile:
        for line in myfile.readlines():
            if line.strip():
                factors = re.split(" ", line)
                sum_spam_count += int(factors[0])           ## sum
up every time the word was found in a spam
                sum_spam_words += int(factors[3])           ## sum
up all words from spams
                sum_spam_records+= int(factors[6])          ## sum
up all emails labeled as spam
                sum_ham_count  += int(factors[9])           ## sum
up every time the word was found in a ham
                sum_ham_words += int(factors[12])           ## sum
up all words from hams
                sum_ham_records += int(factors[15])         ## sum
up all emails labeled as ham
                sum_words += int(factors[18])               ## sum
all words from all emails
                sum_records += int(factors[21])             ## sum
all emails


prior_spam = float(sum_spam_records)/float(sum_records)     ## prio
r prob of a spam email
prior_ham = float(sum_ham_records)/float(sum_records)       ## prio
r prob of a ham email
prob_word_spam = float(sum_spam_count)/float(sum_spam_words)## prob
of word given that email is spam
prob_word_ham = float(sum_ham_count)/float(sum_ham_words)   ## prob
of word given that email is ham

##check_prior = prior_spam + prior_ham                        ## ch
eck priors -> sum to 1
##check_words = float(sum_words)/float(sum_spam_words+sum_ham_word
s) ## check probabilities of a word -> sum to 1
##check_spam = prob_word_spam*float(sum_spam_words)/float(sum_spa
m_count) ## check spam counts -> sum to 1
##check_ham = prob_word_ham*float(sum_ham_words)/float(sum_ham_coun
```

```
t) ## check ham count -> sum to 1
sum_count = sum_spam_count+sum_ham_count

print "Summary of Data"
print '%4s'%sum_records ,'emails examined, containing %6s'%sum_word
s, 'words, we found %3s'%sum_count ,'matches.'

print '%30s' %'ID', '%10s' %'TRUTH', '%10s' %'CLASS'
for file in filenames:
    with open (file, "r") as myfile:
        for line in myfile.readlines():
            if line.strip():
                data = re.split(" ", line)
                record_id = data[24]
                y_true = data[27][0]
                count = int(data[0]) + int(data[9])
                p_spam = prior_spam*prob_word_spam**count
                p_ham = prior_ham*prob_word_ham**count
                if p_spam > p_ham:
                    y_pred = 1
                else:
                    y_pred = 0

                print  '%30s' %record_id, '%10s' %y_true, '%10s'
%y_pred
```

```
Overwriting reducer.py
```

In [26]:
```
!chmod +x reducer.py
```

Write script to file

In [27]:
```
%%writefile pNaiveBayes.sh
## pNaiveBayes.sh
## Author: Jake Ryland Williams
## Usage: pNaiveBayes.sh m wordlist
## Input:
##       m = number of processes (maps), e.g., 4
##       wordlist = a space-separated list of words in quotes,
e.g., "the and of"
##
## Instructions: Read this script and its comments closely.
##               Do your best to understand the purpose of each com
mand,
##               and focus on how arguments are supplied to mappe
r.py/reducer.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.

## collect user input
m=$1 ## the number of parallel processes (maps) to run
wordlist=$2 ## if set to "*", then all words are used

## a test set data of 100 messages
data="enronemail_1h.txt"

## the full set of data (33746 messages)
# data="enronemail.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 processe
s
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 temporary file on disk
    ####
    ####
    ./mapper.py $datachunk "$wordlist" > $datachunk.counts &
    ####
    ####
done
## wait for the mappers to finish their work
```

```
wait

## 'ls' makes a list of the temporary count files
## 'perl -pe' regex replaces line breaks with spaces
countfiles=`\ls $data.chunk.*.counts | perl -pe 's/\n/ /'`

## feed the list of countfiles to the python reducer and redirect S
TDOUT to disk
####
####
./reducer.py $countfiles > $data.output
####
####
numOfInstances=$(cat $data.output)
echo "NB Classifier based on word(s): $wordlist" ## Print out words
echo "$numOfInstances" ## Print out output data
## clean up the data chunks and temporary count files
\rm $data.chunk.*
```

```
Overwriting pNaiveBayes.sh
```

Run file

In [28]:
```
!./pNaiveBayes.sh 5 "assistance"
```

```
NB Classifier based on word(s): assistance
Summary of Data
  98 emails examined, containing  35352 words, we found   9 matche
s.
```

ID	TRUTH	CLASS
0001.1999-12-10.farmer	0	0
0001.1999-12-10.kaminski	0	0
0001.2000-01-17.beck	0	0
0001.2001-02-07.kitchen	0	0
0001.2001-04-02.williams	0	0
0002.1999-12-13.farmer	0	0
0002.2001-02-07.kitchen	0	0
0002.2001-05-25.SA_and_HP	1	0
0002.2003-12-18.GP	1	0
0002.2004-08-01.BG	1	1
0003.1999-12-10.kaminski	0	0
0003.1999-12-14.farmer	0	0
0003.2000-01-17.beck	0	0
0003.2001-02-08.kitchen	0	0
0003.2003-12-18.GP	1	0
0003.2004-08-01.BG	1	0
0004.1999-12-10.kaminski	0	1
0004.1999-12-14.farmer	0	0
0004.2001-04-02.williams	0	0
0004.2001-06-12.SA_and_HP	1	0
0004.2004-08-01.BG	1	0
0005.1999-12-12.kaminski	0	1
0005.1999-12-14.farmer	0	0
0005.2000-06-06.lokay	0	0
0005.2001-02-08.kitchen	0	0
0005.2001-06-23.SA_and_HP	1	0
0005.2003-12-18.GP	1	0
0006.1999-12-13.kaminski	0	0
0006.2001-02-08.kitchen	0	0
0006.2001-04-03.williams	0	0
0006.2001-06-25.SA_and_HP	1	0
0006.2003-12-18.GP	1	0
0006.2004-08-01.BG	1	0
0007.1999-12-13.kaminski	0	0
0007.1999-12-14.farmer	0	0
0007.2000-01-17.beck	0	0
0007.2001-02-09.kitchen	0	0
0007.2003-12-18.GP	1	0
0007.2004-08-01.BG	1	0
0008.2001-02-09.kitchen	0	0
0008.2001-06-12.SA_and_HP	1	0
0008.2001-06-25.SA_and_HP	1	0
0008.2003-12-18.GP	1	0
0008.2004-08-01.BG	1	0
0009.1999-12-13.kaminski	0	0
0009.1999-12-14.farmer	0	0
0009.2000-06-07.lokay	0	0
0009.2001-02-09.kitchen	0	0

0009.2003-12-18.GP	1	0
0010.1999-12-14.farmer	0	0
0010.1999-12-14.kaminski	0	0
0010.2001-02-09.kitchen	0	0
0010.2001-06-28.SA_and_HP	1	1
0010.2003-12-18.GP	1	0
0010.2004-08-01.BG	1	0
0011.1999-12-14.farmer	0	0
0011.2001-06-28.SA_and_HP	1	0
0011.2001-06-29.SA_and_HP	1	0
0011.2003-12-18.GP	1	0
0011.2004-08-01.BG	1	0
0012.1999-12-14.farmer	0	0
0012.1999-12-14.kaminski	0	0
0012.2000-01-17.beck	0	0
0012.2000-06-08.lokay	0	0
0012.2001-02-09.kitchen	0	0
0012.2003-12-19.GP	1	0
0013.1999-12-14.farmer	0	0
0013.1999-12-14.kaminski	0	0
0013.2001-04-03.williams	0	0
0013.2001-06-30.SA_and_HP	1	0
0013.2004-08-01.BG	1	1
0014.1999-12-14.kaminski	0	0
0014.1999-12-15.farmer	0	0
0014.2001-02-12.kitchen	0	0
0014.2001-07-04.SA_and_HP	1	0
0014.2003-12-19.GP	1	0
0014.2004-08-01.BG	1	0
0015.1999-12-14.kaminski	0	0
0015.1999-12-15.farmer	0	0
0015.2000-06-09.lokay	0	0
0015.2001-02-12.kitchen	0	0
0015.2001-07-05.SA_and_HP	1	0
0015.2003-12-19.GP	1	0
0016.1999-12-15.farmer	0	0
0016.2001-02-12.kitchen	0	0
0016.2001-07-05.SA_and_HP	1	0
0016.2001-07-06.SA_and_HP	1	0
0016.2003-12-19.GP	1	0
0016.2004-08-01.BG	1	0
0017.1999-12-14.kaminski	0	0
0017.2000-01-17.beck	0	0
0017.2001-04-03.williams	0	0
0017.2003-12-18.GP	1	0
0017.2004-08-01.BG	1	0
0017.2004-08-02.BG	1	0
0018.1999-12-14.kaminski	0	0
0018.2001-07-13.SA_and_HP	1	1
0018.2003-12-18.GP	1	1

<----------End of HW1.3----------->

HW1.4.

Provide a mapper/reducer pair that, when executed by pNaiveBayes.sh will classify the email messages by a list of one or more user-specified words. Examine the words “assistance”, “valium”, and “enlargementWithATypo” and report your results

Run file

In [29]:
```
!./pNaiveBayes.sh 5 "assistance valium enlargementWithATypo"
```

```
NB Classifier based on word(s): assistance valium enlargementWithA
Typo
Summary of Data
  98 emails examined, containing  35352 words, we found  12 matche
s.
```

ID	TRUTH	CLASS
0001.1999-12-10.farmer	0	0
0001.1999-12-10.kaminski	0	0
0001.2000-01-17.beck	0	0
0001.2001-02-07.kitchen	0	0
0001.2001-04-02.williams	0	0
0002.1999-12-13.farmer	0	0
0002.2001-02-07.kitchen	0	0
0002.2001-05-25.SA_and_HP	1	0
0002.2003-12-18.GP	1	0
0002.2004-08-01.BG	1	1
0003.1999-12-10.kaminski	0	0
0003.1999-12-14.farmer	0	0
0003.2000-01-17.beck	0	0
0003.2001-02-08.kitchen	0	0
0003.2003-12-18.GP	1	0
0003.2004-08-01.BG	1	0
0004.1999-12-10.kaminski	0	1
0004.1999-12-14.farmer	0	0
0004.2001-04-02.williams	0	0
0004.2001-06-12.SA_and_HP	1	0
0004.2004-08-01.BG	1	0
0005.1999-12-12.kaminski	0	1
0005.1999-12-14.farmer	0	0
0005.2000-06-06.lokay	0	0
0005.2001-02-08.kitchen	0	0
0005.2001-06-23.SA_and_HP	1	0
0005.2003-12-18.GP	1	0
0006.1999-12-13.kaminski	0	0
0006.2001-02-08.kitchen	0	0
0006.2001-04-03.williams	0	0
0006.2001-06-25.SA_and_HP	1	0
0006.2003-12-18.GP	1	0
0006.2004-08-01.BG	1	0
0007.1999-12-13.kaminski	0	0
0007.1999-12-14.farmer	0	0
0007.2000-01-17.beck	0	0
0007.2001-02-09.kitchen	0	0
0007.2003-12-18.GP	1	0
0007.2004-08-01.BG	1	0
0008.2001-02-09.kitchen	0	0
0008.2001-06-12.SA_and_HP	1	0
0008.2001-06-25.SA_and_HP	1	0
0008.2003-12-18.GP	1	0
0008.2004-08-01.BG	1	0
0009.1999-12-13.kaminski	0	0
0009.1999-12-14.farmer	0	0
0009.2000-06-07.lokay	0	0

0009.2001-02-09.kitchen	0	0
0009.2003-12-18.GP	1	1
0010.1999-12-14.farmer	0	0
0010.1999-12-14.kaminski	0	0
0010.2001-02-09.kitchen	0	0
0010.2001-06-28.SA_and_HP	1	1
0010.2003-12-18.GP	1	0
0010.2004-08-01.BG	1	0
0011.1999-12-14.farmer	0	0
0011.2001-06-28.SA_and_HP	1	0
0011.2001-06-29.SA_and_HP	1	0
0011.2003-12-18.GP	1	0
0011.2004-08-01.BG	1	0
0012.1999-12-14.farmer	0	0
0012.1999-12-14.kaminski	0	0
0012.2000-01-17.beck	0	0
0012.2000-06-08.lokay	0	0
0012.2001-02-09.kitchen	0	0
0012.2003-12-19.GP	1	0
0013.1999-12-14.farmer	0	0
0013.1999-12-14.kaminski	0	0
0013.2001-04-03.williams	0	0
0013.2001-06-30.SA_and_HP	1	0
0013.2004-08-01.BG	1	1
0014.1999-12-14.kaminski	0	0
0014.1999-12-15.farmer	0	0
0014.2001-02-12.kitchen	0	0
0014.2001-07-04.SA_and_HP	1	0
0014.2003-12-19.GP	1	0
0014.2004-08-01.BG	1	0
0015.1999-12-14.kaminski	0	0
0015.1999-12-15.farmer	0	0
0015.2000-06-09.lokay	0	0
0015.2001-02-12.kitchen	0	0
0015.2001-07-05.SA_and_HP	1	0
0015.2003-12-19.GP	1	0
0016.1999-12-15.farmer	0	0
0016.2001-02-12.kitchen	0	0
0016.2001-07-05.SA_and_HP	1	0
0016.2001-07-06.SA_and_HP	1	0
0016.2003-12-19.GP	1	1
0016.2004-08-01.BG	1	0
0017.1999-12-14.kaminski	0	0
0017.2000-01-17.beck	0	0
0017.2001-04-03.williams	0	0
0017.2003-12-18.GP	1	0
0017.2004-08-01.BG	1	1
0017.2004-08-02.BG	1	0
0018.1999-12-14.kaminski	0	0
0018.2001-07-13.SA_and_HP	1	1
0018.2003-12-18.GP	1	1

<----------End of HW1.4----------->

<----------End of HW1----------->

In [ ]: