MIDS-W261-HW-01-Sanchez

September 6, 2016

```
In [30]: %%javascript
         /*********************************************************************
         Known Mathjax Issue with Chrome - a rounding issue adds a border to the ri
         https://github.com/mathjax/MathJax/issues/1300
         A quick hack to fix this based on stackoverflow discussions:
         http://stackoverflow.com/questions/34277967/chrome-rendering-mathjax-equat
         *********************************************************************

         $('.math>span').css("border-left-color","transparent")
```

```
<IPython.core.display.Javascript object>
```

```
In [31]: %reload_ext autoreload
         %autoreload 2
```

1 MIDS - w261 Machine Learning At Scale

Course Lead: Dr James G. Shanahan (**email** Jimi via James.Shanahan *AT* gmail.com)

1.1 Assignment - HW1

Name: *Jason Sanchez* (26989981) **Class:** MIDS w261 (Section *Fall 2016 Group 2*)
Email: *jason.sanchez*@iSchool.Berkeley.edu
Week: 1

Due Time: HW is due the Tuesday of the following week by 8AM (West coast time). I.e., Tuesday, Sept 6, 2016 in the case of this homework.

Submit Date: Sept 6, 2016

2 Table of Contents

1 Instructions Back to Table of Contents

MIDS UC Berkeley, Machine Learning at Scale DATSCIW261 ASSIGNMENT #1

Version 2016-09-2

=== INSTRUCTIONS for SUBMISSIONS === Follow the instructions for submissions carefully.

https://docs.google.com/forms/d/1ZOr9RnIe_A06AcZDB6K1mJN4vrLeSmS2PD6Xm3eOiis/viewform?us

2.0.1 IMPORTANT

HW1 can be completed locally on your computer ### Documents: * IPython Notebook, published and viewable online. * PDF export of IPython Notebook.

2 Useful References Back to Table of Contents

- See lecture 1

HW Problems Back to Table of Contents

2.1 3. HW1.0

Back to Table of Contents

2.1.1 HW1.0.1. Self-Introduction

W1.0.0 Prepare your bio and include it in this HW submission. Please limit to 100 words. Count the words in your bio and print the length of your bio (in terms of words) in a separate cell.

Fill in the following information [Optional] * Your Location * When did you start MIDS and what is your target finish date * What you want to get out of w261?

Hi! My name is Jason Sanchez. I live in San Francisco. I started MIDS in Fall 2015 and expect to finish in Summer 2017. Why so long? I am a cofounder of a startup that is using data science to modernize the insurance industry.

After completing the first week of class, I am extremely hopeful that this class will define my Berkeley experience. This class seems difficult, but extremely worth the effort and I would like to feel comfortable tackling big data problems in production environments.

```
In [23]: !wc -w < bio
```

```
      86
```

2.1.2 HW1.0.2. Big data

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

Big data problems are ones that cannot be solved in a reasonable amount of time using one computer.

Processing trip data from telematics devices (i.e. systems that monitor driving behavior) in real time.

2.1.3 HW1.0.3. Bias Variance

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

bias: The difference in predictions between the trained model and the true function that generated the data (not including the irreducible noise).

variance: The variablility in the predictions of a model given different training data.

Simple models (i.e. models that produce simple decision boundaries) have high bias and adding a lot more data won't dramatically increase the accuracy of the model. Such models are said to underfit the data because they do not have the ability to use the data available to better infer the true decision boundary.

Complex models that produce highly complex decision boundaries tend to have high variance - especially in smaller datasets. This variance is reducible by adding more data. In fact, adding an infinite amount of data would drive the variance of any model down to zero (leaving only the bias).

The problem is we don't have an infinite amount of data and we must make a tradeoff between bias and variance. By plotting the cross validation error rates attained from moving from a simple model to a complex model (in terms of the hyperparameters of the model), we can see where the inflection point between bias and variance occurs.

When the model is simple, it misses a lot of the signal and thus the error is high. Increasing the complexity or flexibility of the model tends to increase the accuracy because more of the signal is being learned. Increasing the complexity of the model further can then lead to performance degradation because the model starts to learn patterns that only exist in the training dataset.

Therefore, in machine learning we must balance the bias and variance of a model given the size of the dataset available.

2.2 3. 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): Word Count ========== a 631 a-piece 1 abide 1 able 1 about 94 above 3 absence 1 absurd 2

```
In [24]: # check where is the current directory and change if necessary using somet
         !pwd
```

```
/Users/BlueOwl1/Desktop/w261
```

```
In [1]: ls
```

```
An Introduction to Information Retrieval.pdf
CountOfMonteCristo.txt
DivideAndConquer.ipynb
DivideAndConquer.pdf
LICENSE.txt
MIDS-W261-HW-01-Sanchez.ipynb
bio
mapper.py*
pGrepCount.sh*
reducer.py*
```

```
In [8]: !curl 'http://www.gutenberg.org/cache/epub/11/pg11.txt' -o alice.txt
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  163k  100  163k    0     0   175k      0 --:--:-- --:--:-- --:--:--  175k
```

```
In [9]: #display the first few lines
        !head alice.txt
```

```
In [5]: #example of a regular expression to detect words in a string.
        import re
        line = """ 0017.2000-01-17.beck        0         global risk management ope
        re.findall(r'[a-z]+', line.lower())[0:10]
```

```
Out[5]: ['beck',
         'global',
         'risk',
         'management',
         'operations',
         'congratulations',
         'sally',
         'kk',
         'forwarded',
         'by']
```

2.2.1 Dictionaries are a good way to keep track of word counts

wordCounts={}

2.2.2 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 [6]: # Here is an example of wordcounting with a defaultdict (dictionary structu
        # 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 ope
        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
```

2.2.3 HW1.1.1 How many times does the word alice occur in the book?

```
In [433]: with open("alice.txt", "r") as alice:
              print(len(re.findall(r'alice+', alice.read().lower())))
```

```
403
```

The above results doesn't make sense in light of the bottom result

```
In [435]: with open("alice.txt", "r") as alice:
              print(len(re.findall(r'[Aa]lice+', alice.read())))
```

```
398
```

```
In [440]: with open("alice.txt", "r") as alice:
              print(len(re.findall(r'alice+', alice.read())))
```

```
0
```

2.3 3. 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 [446]: %%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
          ##       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/red
          ##               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 starte


          usage()
          {
              echo ERROR: No arguments supplied
              echo
              echo To run use
              echo "     pWordCount.sh m wordlist inputFile"
              echo Input:
              echo "      number of processes/maps, EG, 4"
              echo "      wordlist = a space-separated list of words in quotes, e.g
              echo "      inputFile = a text input file"
          }

          if [ $# -eq 0 ] # I removed a hash after the $ sign
            then
              usage
```

```
    exit 1
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=$3

## Clean up remaining chunks from a previous run
rm -f $data.chunk.*

## '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 $data.chunk.*; do
    ## feed word list to the python mapper here and redirect STDOUT to a
    ####
    ####
    ./mapper.py  "$wordlist" <$datachunk > $datachunk.counts &
    ####
    ####
done
## wait for the mappers to finish their work
wait

###------------------------------------------------------------------------
# Sorting magic

hash_to_bucket ()
{
# Takes as input a string and n_buckets and assigns the string to one of
# This function is the bottle neck and takes forever to run
string=$1
n_buckets=$2

# Convert to checksum then take the mod of it
```

```
        echo -n $string | cksum | cut -f1 -d" " | awk -v n="$n_buckets" '{print $
        }

        sorted_file_prefix=sorted_pwordcount

        rm -f $sorted_file_prefix* *output

        # For each file with counts, create coallated files
        for file in $data.chunk.*.counts
            do
                cat $file | while read line
                    do
                        key=$(echo $line | cut -f1 -d" ")
                        hash_key=$(hash_to_bucket $key $m)
                        echo $line >> $sorted_file_prefix.$hash_key
                    done
            done

        # Sort these files
        for file in $sorted_file_prefix*
            do
                sort $file -o $file
            done
        #
        ###------------------------------------------------------------------------


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

        ## feed the list of countfiles to the python reducer and redirect STDOUT
        ####
        ####
        for file in $sorted_file_prefix*
            do
                <$file ./reducer.py >> $data.output
            done

        sort $data.output -o $data.output
        ####
        ####

        ## clean up the data chunks and temporary count files
        rm $data.chunk.*
```

```
Overwriting pWordCount.sh
```

```
In [447]: # Change the execution priviledges to make the shell script executable by
          !chmod a+x pWordCount.sh
```

2.3.1 Please feel free to adopt and modify the following mapper for your purpose¶

```
In [369]: %%writefile mapper.py
          #!/usr/bin/python
          import sys
          import re

          findword = sys.argv[1]
          findword = findword.lower()

          if findword != "*":
              findword = set(findword.split())

          for line in sys.stdin:
              # Turn the list into a collection of lowercase words
              for word in re.findall(r'[a-z]+', line.lower()):
                  if (word in findword) or (findword == "*"):
                      print("%s 1" % word)

Overwriting mapper.py


In [370]: !chmod +x mapper.py
```

2.3.2 Please feel free to adopt and modify the following reducer for your purpose¶

(i.e., there will be no need for a sort in reducer.py code; leverage mapreduce framework).

```
In [417]: %%writefile reducer.py
          #!/usr/bin/python
          import sys

          total = 1
          current_word = None

          for countStr in sys.stdin:
              word, count = countStr.split()
              if current_word == word:
                  total += int(count)
              elif current_word is None:
                  current_word = word
              else:
                  print(current_word, total)
                  current_word = word
                  total = 1
          print(current_word, total)
```

```
Overwriting reducer.py
```

```
In [418]: ! chmod +x reducer.py
```

2.3.3 Dont forget to add a sort component to your MapReduce framework and leverage the sort order in your reduceer (i.e., there will be no need for a sort in reducer.py).

I.e., insert code

```
In [441]: !chmod a+x mapper.py
          !chmod a+x reducer.py
          !chmod a+x pWordCount.sh
          ! ./pWordCount.sh

ERROR: No arguments supplied

To run use
     pWordCount.sh m wordlist inputFile
Input:
      number of processes/maps, EG, 4
      wordlist = a space-separated list of words in quotes, e.g., 'the and of'
      inputFile = a text input file
```

2.4 3. HW1.3 WordCount via Command Line Map Reduce Framework

Back to Table of Contents

Write the mapper.py/reducer.py combination to perform WordCount using the command line mapreeduce framework 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):

To do so, make sure of the following:

- That the mapper.py counts all occurrences of a single word
- In the pWordCount.sh, please insert a sort command between the mappers (after the for loop) and the reducer calls to collate the output key-value pair records by key from the mappers. E.g., sort -k1,1. Use "man sort" to learn more about Unix sorts.
- reducer.py sums the count value from the collated records for each word. There should be no sort in the reducer.py

Word Count ========== a 631 a-piece 1 abide 1 able 1 about 94 above 3 absence 1 absurd 2 Here, mapper.py will read in a portion (i.e., a single record corresponding to a row) of the email data, count the number of occurences of the word in questions and print/emit a count to the output stream. While the utility of the reducer responsible for reading in counts of the word and summarizing them before printing that summary to the output stream. See example the notebook See video section 1.12.1 1.12.1 Poor Man's MapReduce Using Command Line (Part 2) located at: https://learn.datascience.berkeley.edu/mod/page/view.php?id=10961

NOTE in your python notebook create a cell to save your mapper/reducer to disk using magic commands (see example here)

```
In [442]: ! ./pWordCount.sh 4 "*" alice.txt

In [450]: !head alice.txt.output
```

```
('a', 690)
('abide', 2)
('able', 1)
('about', 102)
('above', 3)
('absence', 1)
('absurd', 2)
('accept', 1)
('acceptance', 1)
('accepted', 2)
```

2.5 3. HW1.4

Back to Table of Contents

Change the mapper.py/reducer.py combination so that you get only the number of words starting with an uppercase letter, and the number of words starting with a lowercase letter for Alice in Wonderland available here. In other words, you need an output file with only 2 lines, one giving you the number of words staring with a lowercase ('a' to 'z'), and the other line indicating the number of words starting with an uppercase letter ('A' to 'Z'). In the pWordCount.sh, please insert a sort command between the mappers (after the for loop) and the reducer calls to collate the output key-value pair records by key from the mappers. E.g., sort -k1,1. Use "man sort" to learn more about Unix sorts.

```
In [451]: %%writefile mapper.py
          #!/usr/bin/python
          import sys
          import re

          for line in sys.stdin:
              # Turn the list into a collection of lowercase words
              for word in re.findall(r'[a-zA-Z]+', line):
                  if word.isupper():
                      print("Capital 1")
                  else:
                      print("Lower 1")
```

```
Overwriting mapper.py
```

```
In [452]: ! ./pWordCount.sh 4 "*" alice.txt

In [453]: ! cat alice.txt.output
```

```
('Capital', 1096)
('Lower', 29323)
```

2.6 3. HW1.5 Bias-Variance (This is an OPTIONAL HW)

Back to Table of Contents

Provide and example of bias variance in action for a similated function y = f(x). E.g., y = sin(x+x^2). Provide code, data, and graphs.

Using a bias-variance decomposition analsysis on your choosen problem, describe how you would decide which model to choose when you dont know the true function and how does this choice compares to the choice you made using the true function.

In [ ]:

Back to Table of Contents

——- END OF HOWEWORK ——