DAMLAS MidTerm Exam

11:00AM - 1:00PM(CT) August 22, 2016
End of Term Exam

Data Analytics and Machine Learning at Scale

Target, Minneapolis

Please insert your contact information here

Mark von Oven
Mark.Vonoven@target.com

Instructions for exam

1. Please submit your solutions and notebook via the following form:

   Exam Submission form (http://goo.gl/forms/iN4XIXQyiH1iAdyD3)

1. **Please acknowledge receipt of exam by sending a quick reply to the Jimi**
2. Review the submission form first to scope it out (it will take a 5-10 minutes to input your answers and other information into this form)
3. Please keep all your work and responses in ONE (1) notebook only (and submit via the submission form)
4. Please make sure that the NBViewer link for your Submission notebook works
5. Please do NOT discuss this exam with anyone (including your class mates) **until after Monday, September 5.**
6. This is an open book exam meaning you can consult webpages and textbooks, class notes, slides etc. but you can not consult each other or any other person/group. Please complete this exam by yourself within the time limit.
7. For markdown help in iPython Notebooks please see: https://sourceforge.net/p/ipython/discussion/markdown_syntax (https://sourceforge.net/p/ipython/discussion/markdown_syntax)

Exam questions begins here

In [1]:
```
import numpy as np
from __future__ import division

%reload_ext autoreload
%autoreload 2
```

In [2]:
```
import os
import sys #current as of 9/26/2015
import pyspark
from pyspark.sql import SQLContext
# We can give a name to our app (to find it in Spark WebUI) and configure execution mode
# In this case, it is local multicore execution with "local[*]"
app_name = "example-logs"
master = "local[*]"
conf = pyspark.SparkConf().setAppName(app_name).setMaster(master)
sc = pyspark.SparkContext(conf=conf)
sqlContext = SQLContext(sc)
print(sc)
print(sqlContext)
# Import some libraries to work with dates
import dateutil.parser
import dateutil.relativedelta as dateutil_rd
```

```
<pyspark.context.SparkContext object at 0x7f00f0153a58>
<pyspark.sql.context.SQLContext object at 0x7f00f076e128>
```

ET:1

Assume your tasked with modeling a REGRESSION problem. How do you determine which variables may be important?

1. If your data has unknown structure, start with: Tree-based methods
2. If statistical measures of importance are needed, start with: linear models (think Generalized linear models (GLMs)
3. If statistical measures of importance are not needed, start with: Regression with shrinkage (e.g., LASSO, Elastic net)
4. If statistical measures of importance are not needed, use : Stepwise regression

Select the single most correct response from the following:

- (a) 1
- (b) 2
- (c) 3
- (d) 1, 2, 3, 4

ET:1 - b

ET:2

Using one-hot-encoding, a categorical feature with four distinct values would be represented by how many features?

(a) 1 feature
(b) 2 features
(c) 4 features
(d) none of the above

ET:2 - b

ET:3

When it comes to decision tree prediction (in the case of either classification or regression problems) which of the the following steps are perfermed:

1. A decision tree takes as input an object or situation described by a set of attributes and returns a decision the predicted out value for the input.
2. A decision tree reaches its decision by performing a sequence of tests.
3. Each internal node in the tree corresponds to a test of the value of one of the properties, and the branches from the node are labeled with the possible values of the test.
4. Each leaf node in the tree specifies the value to be returned if that leaf is reached.
5. (a) 1
6. (b) 1, 2, 3, 4
7. (c) 3
8. (d) 1, 2, 3

ET:3 - b

ET:4

Using Claude Shannon's model of Entropy, compute the:

- Entropy of a fair coin
- Entropy of an unfair coint that comes up heads 80% of the time

Select the correct answer from the following:

- (a) 1 bit (entropy of a fair coin) and 0.8 bits (the biased coin)
- (b) 1 bit (entropy of a fair coin) and 1 bit (the biased coin)
- (c) 1 bit (entropy of a fair coin) and 0.2068 bits (the biased coin)
- (d) 1 bit (entropy of a fair coin) and 0.2 bits (the biased coin)

ET:4 - d

ET:5

Given the following gradient descent scenario indicate which of the statements below are correct:

- Given an objective function (also know as a cost fucntion) that we wish to minimize. This function has one decision variable w1.
- Suppose we reach a local minimum, i.e., w1 is already at a local minimum, what do you think one step of gradient descent will do?

STATEMENTS

1. It turns out at the local optimum, your derivative will be equal to zero.
2. So the slope of the line tangent to the objective function at this point w1 will be equal to zero and thus this derivative term is equal to zero.
3. The gradient descent update at this local optimum is, w1 = w1 - alpha * 0; where alpha is the learning rate
4. If you're already at the local optimum it leaves w1 unchanged cause its updates as w1 equals w1. So if your parameters are already at a local minimum one step with gradient descent does absolutely nothing. It keeps your solution at the local optimum.

Select the single most correct answer from the following:

- (a) 1, 2, 3, 4
- (b) 2
- (c) 3,4
- (d) 2,3,4

ET:5 - b

ET:6

Given the following gradient descent scenario indicate which of the statements below are correct:

- Given a convex objective function (also known as a cost fucntion) that we wish to minimize. This function has one decision variable w1.
- Suppose we reach a local minimum, i.e., w1 is already at a local minimum, what do you think one step of gradient descent will do?
- use a fixed learning rate alpha; assume alpha is 1.

STATEMENTS
a. When the current estimate of the minimum of the objective function is further away from the mimimum the gradient is much bigger
b. When the current estimate of the minimum of the objective function is close to the mimimum, my derivative term is even smaller and so the magnitude of the update to w1 is even smaller.
c. As gradient descent runs, you will automatically take smaller and smaller steps. Until eventually you're taking very small steps, and you finally converge to the global minimum (it is a convex optimization after all).
d. Gradient descent will never converge will a fixed learning rate

ET:6 - a

ET:7

When dealing with numercial data which of the following are ways to deal with missing data:

(a) Delete records that have missing input values
(b) Standardize the data and set all missing values to 1 (one)
(c) Use K-nearest neighbours based on the test set to fill in missing values in the training set
(d) none of the above

ET:7 - c

ET:8

In a digital advertising problem where we are modeling the Pr(Click|context) and have no downstream knowledge of what purchases resulted from a click on the ad shown, which of the following approaches would lead to potentially more clicks on the ads shown?

(a) Model Click vs not click event using linear regression
(b) Model Click vs not click event using logistic regression
(c) Model probability of a purchase
(d) none of the above

ET:8 - b

ET:9

Which of the following are true about the purpose of a loss function?

(a) It's a way to penalize a model for incorrect predictions
(b) It precisely defines the optimization problem to be solved for a particular learning model
(c) Loss functions can be used for modeling both classification and regression problems
(d) none of the above

ET:9 - a, c

ET:10

When implementing Logistic (or linear) Regression with Regularization in Spark which of the following apply when using the following cost function:

$$minimize\left(f(w) := \lambda R(w) + \sum_{k=1}^{n} (w; w_i, y_i)\right)$$

(a) When lambda equals one, it provides the same result as standard logistic (linear) regression
(b) One only needs to modify the standard logistic (linear) regression (i.e., with no regularization term) by adding some code after the map-reduce (loss) gradient steps
(c) When lambda equals zero, it provides the same result as standard logistic (linear) regression
(d) One only needs to modify the standard logistic (linear) regression by modifying the mapper

ET:10 - c

ET:11 Given the following paired RDDs

RDD1 = {(1, 2), (3, 4), (3, 6)} RDD2 = {(3, 9) (3, 6)}

Using PySpark, write code to perform an inner join of these paired RDDs. What is the resulting RDD? Make your Spark available in your notebook:

a: [(3, (4, 9)), (3, (6, 9))]
b: [(3, (4, 9)), (3, (4, 6)), (3, (6, 9)), (3, (6, 6))]
c: [(3, (4, 9)), (3, (4, 6)), (3, (6, 9)), (3, (6, 9))]
d: None of the above

In [12]:
```
rdd1 =  sc.parallelize([(1, 2), (3, 4), (3, 6)])
rdd2 =  sc.parallelize([(3, 9), (3, 6)])

rddNEW = rdd1.join(rdd2)
for x in rddNEW.collect():
    print(x)
```

```
(3, (4, 9))
(3, (4, 6))
(3, (6, 9))
(3, (6, 6))
```

ET:11 - b

ET:12 You have been tasked to build a predictive model to forecast beer sales for a chain of stores.

After doing basic exploratory analysis on the data, what is the first thing you do regarding modeling?

(a) Construct a baseline model
(b) Determine a metric to evaluate your machine learnt models
(c) Split your data into training, validation and test subsets (or split using cross fold validatation)
(d) All of the of the above

ET:12 - b

ET:13

The beer sales data consists of 52 weeks of cases-sold and price-per-case data for 3 carton sizes of beer (12-packs, 18-packs, 30-packs) at a small chain of supermarkets.

Three additional rows of hypothetical price data for 12-,18-, 30-packs have been entered for purposes of forecasting from the models.

Use Spark and the following notebook, https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0 (https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0) to answer this question.

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a model for say a forecasting method in statistics, for example in trend estimation. It usually expresses accuracy as a percentage, and is defined by the formula:

MAPE = average over all examples (100*Abs(Actual - Predicted) / Actual))

Note when Actual is zero that test row is dropped from the evaluation.

Construct a mean model for target variable `CASES18PK` (for the purposes of this question, take the mean of the CASES18PK column as your model prediction). Calculate the MAPE for the mean model over the training set. Select the closest answer to your calculated MAPE.

(a) 200%
(b) 250%
(c) 20%
(d) 180%

In [ ]:
```
i=0
total = 0
for line in df.take(100):
    words = line.split()
    if i>0:
        total += float(words[2])
    i+=1

prediction = total/i

print('Model Prediction(Mean of PRICE18PK) is', prediction)

i=0
total = 0
for line in df.take(100):
    words = line.split()
    if i>0:
        example = 100*abs(float(words[2])-prediction) / float(words[2])
        print('100*abs(', words[2], '-', prediction, ') /', words[2], '=', example)
        total += example
    i+=1

answer = total/i

print('MAPE is', answer)
```

ET:13 - c

ET:14

Use Spark and the following notebook for the following questions:

- http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb (http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb)
- https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0 (https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0) to answer this question.

The target variable `CASES18PK` is skewed, so take the log of it (and make it more normally distributed) and compute the MAPE of the mean model for `LN_CASES18PK`. Select the closest answer to your calculated MAPE.

(a) 200%
(b) 30%
(c) 20%
(d) 10%

ET:14 - d

ET:15

Use Spark and the following notebook for the following questions:

- http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb (http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb)
- https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0 (https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0) to answer this question.

Build a linear regression model using the following variables:

Log(CASES18PK) ~ log(PRICE12PK) + log(PRICE18PK) + log(PRICE30PK)

Calculate MAPE over the test set and select the closest answer.

(a) 4.3%
(b) 4.6%
(c) 3.5%
(d) 3.9%

ET:15 - b

ET:16

Recall that Spark automatically sends all variables referenced in your closures to the worker nodes. While this is convenient, it can also be inefficient because (1) the default task launching mechanism is optimized for small task sizes, and (2) you might, in fact, use the same variable in multiple parallel operations, but Spark will send it separately for each operation. As an example, say that we wanted to write a Spark program that looks up countries by their call signs (e.g., the call sign for Ireland is EJZ) by prefix matching in an table. In the following the "signPrefixes" variable is essentially a table with two columns "Sign" and "Country Name". The goal is to join the following tables:

`signPrefixes` table with columns "Sign" and "Country Name"
`contactCounts` table with columns "Sign" and "count"

to yield a new table:

`countryContactCounts` with the following columns "Country Name" and "count"

Use Spark and the following notebook for the following questions:

- http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb (http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb)
- https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0 (https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0) to answer this question.

In [ ]:

```
#..... Other code...
#Country lookup code

## Helper functions for looking up the call signs

 def lookupCountry(sign, prefixes):
    pos = bisect.bisect_left(prefixes, sign)
    return prefixes[pos].split(",")[1]


def loadCallSignTable():
    f = open("callsign_tbl_sorted.txt", "r")
    return f.readlines()

## Lookup the locations of the call signs on the
## RDD contactCounts. We load a list of call sign
## prefixes to country code to support this lookup.
signPrefixes = loadCallSignTable()


def processSignCount(sign_count, signPrefixes):
    country = lookupCountry(sign_count[0], signPrefixes)
    count = sign_count[1]
    return (country, count)

countryContactCounts = (contactCounts
                        .map(lambda signCount: processSignCount(signCount, signPrefixes))
                        .reduceByKey((lambda x, y: x + y)))

countryContactCounts.saveAsTextFile(outputDir + "/countries.txt")
```

How can we modfify this code to make it more efficient? Choose one response only

(a) modify line 18 with `sc.broadcast(loadCallSignTable())`

(b) Use accumulators to store the counts for each country
(c) The code is already optimal
(d) none of the above

ET:16 - a

End of Exam