W261 MIDS Machine Learning at Scale

End of Term exam

Week 14

Fall, 2016
Student Name: [Jason, Sanchez]

Student ID: 26989981

Email: jason.sanchez@ischool.berkeley.edu

Exam Schedule (All times are in California Time)

4:00 PM - 6:00 PM

Instructions for exam

1. Please acknowledge receipt of exam by sending a quick reply to the me
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): http://goo.gl/forms/ggNYfRXz0t (http://goo.gl/forms/ggNYfRXz0t)
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, Midday, of week 16
6. Please submit your solutions and notebook via the following form: http://goo.gl/forms/ggNYfRXz0t (http://goo.gl/forms/ggNYfRXz0t)

This is an open book exam meaning you can consult webpages and textbooks, class notes, slides etc. but you can not each other or any other person/group. Please complete this exam by yourself within the time limit.

1. 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

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

Answer: D

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

Answer: C

ET:3

In the following (and also referring to HW12: http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/1wb2rdqbet54y1h/MIDS-MLS-Project-Criteo-CTR.ipynb (http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/1wb2rdqbet54y1h/MIDS-MLS-Project-Criteo-CTR.ipynb)) we have hashed the three sample points using numBuckets=4 and numBuckets=100. Complete the three statements below about these hashed features summarized in the following table using each answer once.

Name	**Raw Features**	**4 Buckets**	**100 Buckets**
sampleOne	[(0, 'mouse'), (1, 'black')]	{2: 1.0, 3: 1.0}	{14: 1.0, 31: 1.0}
sampleTwo	[(0, 'cat'), (1, 'tabby'), (2, 'mouse')]	{0: 2.0, 2: 1.0}	{40: 1.0, 16: 1.0, 62: 1.0}
sampleThree	[(0, 'bear'), (1, 'black'), (2, 'salmon')	{0: 1.0, 1: 1.0, 2: 1.0}	{72: 1.0, 5: 1.0, 14: 1.0}

With 100 buckets, sampleOne and sampleThree both contain index 14 due to __.

(a) A hash collision
(b) Underlying properties of the data
(c) The fact that used 100 buckets
(d) none of the above

Answer: ? B (Only answer not used from below)

ET:4

In the following (and also referring to HW12: http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/1wb2rdqbet54y1h/MIDS-MLS-Project-Criteo-CTR.ipynb (http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/1wb2rdqbet54y1h/MIDS-MLS-Project-Criteo-CTR.ipynb)) we have hashed the three sample points using numBuckets=4 and numBuckets=100. Complete the three statements below about these hashed features summarized in the following table using each answer once.

Name	**Raw Features**	**4 Buckets**	**100 Buckets**
sampleOne	[(0, 'mouse'), (1, 'black')]	{2: 1.0, 3: 1.0}	{14: 1.0, 31: 1.0}
sampleTwo	[(0, 'cat'), (1, 'tabby'), (2, 'mouse')]	{0: 2.0, 2: 1.0}	{40: 1.0, 16: 1.0, 62: 1.0}
sampleThree	[(0, 'bear'), (1, 'black'), (2, 'salmon')	{0: 1.0, 1: 1.0, 2: 1.0}	{72: 1.0, 5: 1.0, 14: 1.0}

It is likely that sampleTwo has two indices with 4 buckets, but three indices with 100 buckets due to __.

(a) A hash collision
(b) Underlying properties of the data
(c) The fact that we go from 4 to 100 buckets
(d) none of the above

Answer: C

ET:5

In the following (and also referring to HW12: http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/1wb2rdqbet54y1h/MIDS-MLS-Project-Criteo-CTR.ipynb (http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/1wb2rdqbet54y1h/MIDS-MLS-Project-Criteo-CTR.ipynb)) we have hashed the three sample points using numBuckets=4 and numBuckets=100. Complete the three statements below about these hashed features summarized in the following table using each answer once.

Name	Raw Features	4 Buckets	100 Buckets
sampleOne	[(0, 'mouse'), (1, 'black')]	{2: 1.0, 3: 1.0}	{14: 1.0, 31: 1.0}
sampleTwo	[(0, 'cat'), (1, 'tabby'), (2, 'mouse')]	{0: 2.0, 2: 1.0}	{40: 1.0, 16: 1.0, 62: 1.0}
sampleThree	[(0, 'bear'), (1, 'black'), (2, 'salmon')	{0: 1.0, 1: 1.0, 2: 1.0}	{72: 1.0, 5: 1.0, 14: 1.0}

With 4 buckets, sampleTwo and sampleThree both contain index 0 due to __.

(a) A hash collision
(b) Underlying properties of the data
(c) The fact that we use 4 buckets
(d) none of the above

Answer: A

ET:6

When applying numerical machine learning approaches (and for non-numerical approaches if required) to big data problems which of the following steps are could be used during modeling and are recommended:

(a) Convert categorical features to numerical features via one-hot-encoding and store in a dense representation
(b) Transform categorical features using hashing regardless of how many unique categorical values exist in training and test data
(c) Use matrix factorization to remap your input vectors to latent concepts
(d) none of the above

Answer: D or A (should be sparse if many categories)

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

Answer: These are all ways to deal with missing data.

Issue with A is that you can't predict on new data with missing values. Issue with B is that you should probably set missing values to 0 (but 1 would work if you are using a tree-based method) Issue with C is that it violates the separation of training and testing data.

A is the most reasonable.

ET:8

In the Criteo project, we're trying to predict what:

(a) Revenue from click events
(b) Click-through vs not click event
(c) Probability of a purchase
(d) none of the above

Answer: 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

Answer: All are true. B is best.

ET:10

Many standard machine learning methods can be formulated as a convex optimization problem, i.e. the task of finding a minimizer of a convex function *f* that depends on a variable vector **w** (called weights in the code), which has *d* entries. Formally, we can write this as the optimization problem $\min_{w\in\mathbb{R}^d} f(w)$, where the objective function is of the form (JGS)

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

Here the vectors $x_i \in \mathbb{R}^d$ are the training data examples, for $1 \leq i \leq n$, and $y_i \in \mathbb{R}$ are their corresponding labels, which we want to predict. We call the method linear if *L(w;x,y)* can be expressed as a function of $\boldsymbol{w^T}x$ and y. Several of spark.mllib's classification and regression algorithms fall into this category.

The objective function **f** has two parts: the regularizer that controls the complexity of the model, and the loss that measures the error of the model on the training data. The loss function *L(w;.)* is typically a convex function in **w**. The fixed regularization parameter $\lambda\geq0\lambda\geq0$ (regParam in the code) defines the trade-off between the two goals of minimizing the loss (i.e., training error) and minimizing model complexity (i.e., to avoid overfitting).

When implementing Logistic (or linear) Regression with Regularization in Spark which of the following apply when using the above cost functions (mulitple options may apply):

(I) When lambda equals one, it provides the same result as standard logistic/linear regression
(II) 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
(III) When lambda equals zero, it provides the same result as standard logistic (linear) regression (IV) None of the above

Select the most correct from the following:

- (a) I, II, III
- (b) II, III
- (c) III
- (d) IV

Answer: B (Make sure II is true by looking up the code if time permits)

ET:11

In the context of ecommerce you have just deployed a new conversion rate prediction model to production. This model (aka treatment model) will challenge the control nodel (i.e., the current model) in AB Test manner to see if it can be produce better revenue. Here is the data that was taken from this live AB Test.

```
CONTROL MODEL (our new CTR model)
Impression ID   Revenue
1                $0.50
2                $0.50
3                $3.00
......
20000            $3.00
20001            $3.00
20002            $3.00
20003            $3.00
......
50,001           $3.00
.....
100,000          $4.00
```

All other impressions in this 100,000 sample resulted in zero transactions and therefore zero revenue.

```
TREATMENT MODEL (our new CTR model)
Impression ID   Revenue
1                $1.50
2                $0.50
3                $0.00
......
50,001           $3.00
.....
100,000          $4.00
```

All other impressions in this 100,000 sample resulted in zero transactions and therefore zero revenue.

P-values are a common way to determine the statistical significance of a test. The smaller it is, the more confident you can be that the test results are due to something other than random chance. A common p-value of .05 is a 5% significance level. Similarly, a p-value of .01 is a 1% significance level. A p-value of .20 is a 20% significance level. For this problem set the p-value to 0.01

Which of the following are true:

(a) Based on revenue there is no statistical significant difference between the Control and the Treatment at p-value of 0.05 for a one-sided t-test
(b) Based on transaction rates (tranactions that generated revenue versus not) there is no statistical significant difference between the Control and the Treatment at p-value of 0.05 for a one-sided t-test
(c) AB testing using differences in revenue for this problem is a useful means of determining if the Treatment conversion rate prediction model is better than the control model.
(d) none of the above

```
In [2]: import numpy as np
        from scipy.sparse import coo_matrix
        import scipy.stats

        control=[.5, .5, 3, 3, 3, 3, 3, 3, 4]
        control = [1 for _ in range(len(control))]
        control.extend([0]*(100000-len(control)))
        control=np.asarray(control)

        treatment=[ 1.5, 0, .5, 3, 4]
        treatment = [1 for _ in range(len(treatment))]
        treatment.extend([0]*(100000-len(treatment)))
        treatment=np.asarray(treatment)

        #nb 2-sided test
        scipy.stats.ttest_ind(control, treatment), scipy.stats.ttest_ind(control.
        ype(bool), treatment.astype(bool))

Out[2]: (Ttest_indResult(statistic=1.0690800954955095, pvalue=0.285034870091426
        25),
         Ttest_indResult(statistic=1.0690800954955095, pvalue=0.285034870091426
        25))
```

Answer: D

ET:12

Given this graph expressed in the form of an adjacency list,

```
Node  adjacentNode:weightAssociatedWithEdge
N1    N6:10,  N2:2
N2    N3:1
N3    N4:1
N4    N5:1
N5    N6:1
N6    N7:1
N7    N8:1
N8    N9:1
```

Using the parallel breadth-first search algorithm for determining the shortest path from a single source, how many iterations are required to discover the shortest distances to all nodes from Node 1

A 7
B 8
C 13
D None of the above

Answer: A or B (probably B. Check lecture if time allows)

ET:13 Assume the Lagrangian for SOFT SVMs (unconstrained optimization) is as follows:

minimize [λ/2 * w'w + CΣi=1:n ξi + 1/n Σi=:n max(0, 1 - ξi – yi(w'xi – b)))]

(a) When λ is super small (e.g., 0.000001), then the above Langrangian will yield a Hard SVM

(b) In the context of support vector machines, linear kernels can be readily parallelized in map reduce frameworks such as Spark
(c) Sequential learning via algorithms such perceptron can take advantage of map-reduce frameworks and yield exactly the same results as a single core implementation with significant reductions in training time
(d) When λ is 1.0, then the above Langrangian will yield a Soft SVM

Answer: B (MLlib has a SVM implementation and C has the word *exactly* which is not true)

A and D are false --> C is the parameter to tune for hard vs soft margin.

ET:14 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 [7]:
```python
from pyspark import SparkContext

sc = SparkContext("local[*]")

rdd1 = sc.parallelize([(1, 2), (3, 4), (3, 6)])
rdd2 = sc.parallelize([(3, 9), (3, 6)])
rdd1.join(rdd2).collect()
```

Out[7]:
```
[(3, (4, 9)), (3, (4, 6)), (3, (6, 9)), (3, (6, 6))]
```

Answer: B

ET:15 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

Answer: D

ET:16

Use Spark and the following notebook to answer this question:

- 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)

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`. Calculate the MAPE for the mean model over the training set. Select the closest answer.

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

In [8]:
```
%%writefile beerSales.txt
Week    PRICE12PK       PRICE18PK       PRICE30PK       CASES12PK
CASES18PK       CASES30PK
1       19.98   14.10   15.19   223.5   439     55.00
2       19.98   18.65   15.19   215.0   98      66.75
3       19.98   18.65   13.87   227.5   70      242.00
4       19.98   18.65   12.83   244.5   52      488.50
5       19.98   18.65   13.16   313.5   64      308.75
6       19.98   18.65   15.19   279.0   72      111.75
7       19.98   18.65   13.92   238.0   47      252.50
8       20.10   18.73   14.42   315.5   85      221.25
9       20.12   18.75   13.83   217.0   59      245.25
10      20.13   18.75   14.50   209.5   63      148.50
11      20.14   18.75   13.87   227.0   57      229.75
12      20.12   18.75   13.64   216.5   54      312.00
13      20.12   13.87   14.31   169.0   404     96.75
14      20.13   14.27   13.85   178.0   380     123.25
15      20.14   18.76   14.20   301.5   65      200.50
16      20.14   18.77   13.64   266.5   40      359.75
17      20.13   13.87   14.33   182.5   456     113.50
18      20.13   14.14   13.14   159.0   176     136.50
19      20.13   18.76   13.81   285.5   61      225.50
20      20.13   18.72   15.19   360.0   91      122.25
21      20.13   18.76   13.13   263.0   59      443.75
22      19.18   18.76   13.63   443.5   83      322.75
23      14.78   18.74   15.19   1101.5  41      53.00
24      16.04   18.75   13.89   814.0   47      140.75
25      20.12   18.75   14.28   365.0   84      210.75
26      19.75   18.75   15.19   510.0   85      110.50
27      19.65   18.75   13.12   580.5   116     568.25
28      19.69   13.79   13.78   251.0   544     115.50
29      20.12   13.49   15.19   237.0   890     58.75
30      20.12   14.89   15.19   302.5   371     77.25
31      20.13   13.94   15.19   229.5   557     66.25
32      20.14   13.67   15.19   188.5   775     50.00
33      15.14   14.43   15.19   795.5   236     46.50
34      14.33   18.75   15.19   1556.5  43      65.75
35      16.24   18.22   13.14   807.5   63      252.75
36      19.93   14.06   13.45   243.0   469     179.00
37      21.06   14.43   13.00   201.5   335     226.25
38      21.19   19.48   13.60   294.0   75      288.50
39      21.23   15.15   14.46   220.5   461     114.25
40      20.12   13.79   14.94   255.5   817     70.00
41      14.73   14.31   15.19   920.5   200     47.75
42      14.57   19.50   15.19   730.0   32      98.75
43      15.94   13.85   15.19   262.5   460     77.00
44      20.70   14.23   13.43   209.5   751     160.50
45      19.57   19.31   14.37   283.0   70      143.50
46      19.60   19.29   15.19   262.5   80      133.00
47      19.94   13.76   15.19   310.0   523     68.75
48      21.28   13.45   15.19   278.5   741     81.75
49      14.56   15.13   15.19   741.5   130     56.25
50      14.39   19.43   15.19   1316.0  69      68.75
51      16.81   13.26   15.19   449.0   493     49.25
52      19.86   13.92   15.19   505.0   814     76.50
```

```
Overwriting beerSales.txt
```

In [23]:
```
cases18 = sc.textFile("beerSales.txt") \
            .filter(lambda x: not x.startswith("Week")) \
            .map(lambda x: int(x.split("\t")[5])) \
            .cache()

mean_model = cases18.mean()
mape = cases18.map(lambda x: 100*abs(x-mean_model)/x).mean()
mape
```

Out[23]:
```
200.4442233299054
```

Answer: a

ET:17

Use Spark and the following notebook to answer this question:

- 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)

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 `CASES18PK`. Select the closest answer to your calculated MAPE.

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

In [24]:
```
from math import log

logcases18 = cases18.map(lambda x: log(x)).cache()

mean_model = logcases18.mean()
mape = logcases18.map(lambda x: 100*abs(x-mean_model)/x).mean()
mape
```

Out[24]:
```
19.58493428963071
```

Answer: C

ET:18

Use Spark and the following notebook to answer this question:

- 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)

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%

```
In [72]: from pyspark.mllib.regression import LabeledPoint, LinearRegressionWithS
         GD
         from math import log

         data = sc.textFile("beerSales.txt") \
                  .map(lambda x: x.split("\t")) \
                  .cache()

         header = data.filter(lambda x: x[0] == "Week").collect()[0]

         def make_dataset(x, header):
             x_num = [float(val) for val in x]
             return dict(zip(header, x_num))

         X = data.filter(lambda x: x[0] != "Week") \
                 .map(lambda x: make_dataset(x, header)) \
                 .map(lambda x: LabeledPoint(log(x["CASES18PK"]),
                                             (log(x["PRICE12PK"]),
                                              log(x["PRICE18PK"]),
                                              log(x["PRICE30PK"]))))

         model = LinearRegressionWithSGD.train(X, step=.01, intercept=True, itera
         tions=10000, miniBatchFraction=1, convergenceTol=.0001)

         print(model.weights, model.intercept)

         predictions = X.map(lambda p: (p.label, model.predict(p.features))) \
                        .cache()

         predictions.map(lambda x: 100*abs(x[0] - x[1])/x[0]).mean()
```

```
/usr/local/spark/python/pyspark/mllib/regression.py:281: UserWarning: D
eprecated in 2.0.0. Use ml.regression.LinearRegression.
  warnings.warn("Deprecated in 2.0.0. Use ml.regression.LinearRegressio
n.")

[0.489930128379,0.428743439835,0.441889801716] 1.1644897697659025
```

```
Out[72]: 20.34860159122119
```

After working on trying to get this to match an available answer and finding the parameters just wouldn't converge, I figured I would test it against the scikit-learn implementation.

```
In [50]: from sklearn.linear_model import LinearRegression
         import pandas as pd
         import numpy as np

         X_for_sk = sc.textFile("beerSales.txt") \
                      .map(lambda x: x.split("\t")) \
                      .filter(lambda x: x[0] != "Week") \
                      .map(lambda x: [float(val) for val in x])

         header = ['Week',
                   'PRICE12PK',
                   'PRICE18PK',
                   'PRICE30PK',
                   'CASES12PK',
                   'CASES18PK',
                   'CASES30PK']

         X_data = np.log(pd.DataFrame(X_for_sk.collect(), columns=header))
         y = X_data.pop("CASES18PK")
         X_data.drop(["CASES12PK", "CASES30PK", "Week"], axis=1, inplace=True)

         skmodel = LinearRegression()
         skmodel.fit(X_data, y)
         y_hat = skmodel.predict(X_data)

         (100*(y - y_hat).abs()/y).mean()
```

```
Out[50]: 4.2150866635659234
```

Answer: A (I would roll by own linear model if I had more time)>

ET:19

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 to answer this question:

- 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)

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

Answer: A (You would also have to add .value to the broadcasted variable references)