03_4_CountEncoding

January 28, 2021

[ ]:
```
# The MIT License (MIT)

# Copyright (c) 2020, NVIDIA CORPORATION.

# Permission is hereby granted, free of charge, to any person obtaining a copy␣
 ↪of
# this software and associated documentation files (the "Software"), to deal in
# the Software without restriction, including without limitation the rights to
# use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies␣
 ↪of
# the Software, and to permit persons to whom the Software is furnished to do␣
 ↪so,
# subject to the following conditions:

# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,␣
 ↪FITNESS
# FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
# COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
# IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE
```

1 Tutorial: Feature Engineering for Recommender Systems

2 3. Feature Engineering - Categorical

2.1 3.4. Count Encoding

[1]:
```
import IPython

import pandas as pd
import cudf
import numpy as np
```

```
import cupy
import matplotlib.pyplot as plt

df_train = cudf.read_parquet('./data/train.parquet')
df_valid = cudf.read_parquet('./data/valid.parquet')
df_test = cudf.read_parquet('./data/test.parquet')

df_train['brand'] = df_train['brand'].fillna('UNKNOWN')
df_valid['brand'] = df_valid['brand'].fillna('UNKNOWN')
df_test['brand'] = df_test['brand'].fillna('UNKNOWN')
df_train['cat_2'] = df_train['cat_2'].fillna('UNKNOWN')
df_valid['cat_2'] = df_valid['cat_2'].fillna('UNKNOWN')
df_test['cat_2'] = df_test['cat_2'].fillna('UNKNOWN')
```

[2]:
```
df_train.head()
```

[2]:

	event_time	event_type	product_id	brand	price	user_id	user_session	target	cat_0	cat_1	cat_2	cat_3	timestamp	ts_hour	ts_minute	ts_weekday	ts_day	ts_month	ts_year
0	2019-12-01 00:00:28 UTC	cart	17800342	zeta	66.90	550465671	22650a62-2d9c-4151-9f41-2674ec6d32d5	0	computers	desktop	UNKNOWN	<NA>	2019-12-01 00:00:28	0	0	6	1	12	2019
1	2019-12-01 00:00:39 UTC	cart	3701309	polaris	89.32	543733099	a65116f4-ac53-4a41-ad68-6606788e674c	0	appliances	environment	vacuum	<NA>	2019-12-01 00:00:39	0	0	6	1	12	2019
2	2019-12-01 00:00:40 UTC	cart	3701309	polaris	89.32	543733099	a65116f4-ac53-4a41-ad68-6606788e674c	0	appliances	environment	vacuum	<NA>	2019-12-01 00:00:40	0	0	6	1	12	2019
3	2019-12-01 00:00:41 UTC	cart	3701309	polaris	89.32	543733099	a65116f4-ac53-4a41-ad68-6606788e674c	0	appliances	environment	vacuum	<NA>	2019-12-01 00:00:41	0	0	6	1	12	2019
4	2019-12-01 00:01:56 UTC	cart	1004767	samsung	235.60	579970209	c6946211-ce70-4228-95ce-fd7fccdde63c	0	construction	tools	light	<NA>	2019-12-01 00:01:56	0	1	6	1	12	2019

[3]:
```
cat = 'product_id'
```

2.2 Theory

Count Encoding (CE) calculates the frequency from one or more categorical features given the training dataset. For example we can consider *Count Encoding* as the populiarity of an item or activity of an user.

[4]:
```
ce = df_train[cat].value_counts()
```

[5]:
```
ce
```

[5]:
```
1004767      317711
1005115      251189
1004856      227432
4804056      224545
1005100      180072
              …
100143590         1
100143856         1
100143867         1
100144046         1
100144443         1
Name: product_id, Length: 164453, dtype: int32
```

[6]:
```
ce = ce.reset_index()
```

[7]:
```
ce.columns = [cat, 'CE_' + cat]
df_train.merge(ce, how='left', left_on=cat, right_on=cat)
```

[7]:
```
                       event_time event_type  product_id    brand   price  \
0         2019-12-01 12:27:02 UTC       cart    12700214  UNKNOWN   35.38
1         2019-12-01 12:27:02 UTC       cart    12700214  UNKNOWN   35.38
2         2019-12-01 12:27:02 UTC       cart    12700214  UNKNOWN   35.38
3         2019-12-01 12:27:02 UTC       cart    12700214  UNKNOWN   35.38
4         2019-12-01 12:27:02 UTC       cart    12700214  UNKNOWN   35.38
…                             …          …          …       …      …
11461352  2019-11-30 16:39:49 UTC   purchase     1004873  samsung  346.83
11461353  2019-11-30 16:39:55 UTC   purchase     1005115    apple  915.49
11461354  2019-11-30 16:39:56 UTC   purchase     1004870  samsung  282.89
11461355  2019-11-30 16:40:00 UTC   purchase     1005100  samsung  131.74
11461356  2019-11-30 16:40:04 UTC   purchase     1005253   xiaomi  261.01

            user_id                            user_session  target  \
0         580243411  0cbf5e06-a782-4c74-8002-acf282026d82       0
1         580243411  0cbf5e06-a782-4c74-8002-acf282026d82       0
2         580243411  0cbf5e06-a782-4c74-8002-acf282026d82       0
3         580243411  0cbf5e06-a782-4c74-8002-acf282026d82       0
4         580243411  0cbf5e06-a782-4c74-8002-acf282026d82       0
…               …                                     …      …
```

```
11461352  561528294  67086842-ef36-4dc9-9a30-2aad4a7b191e       1
11461353  573793765  1966b5dc-bf01-40a9-b5d8-b2a0797be941       1
11461354  562742398  4817e492-3278-414d-9f6d-6a4305edc55c       1
11461355  518629444  915b4d55-be00-4501-8eff-d40aec8de72c       1
11461356  521716329  9598b75d-9342-4b56-aab6-9d8690f09a7a       1

                cat_0       cat_1    cat_2 cat_3           timestamp  \
0                <NA>        <NA>  UNKNOWN  <NA>  2019-12-01 12:27:02
1                <NA>        <NA>  UNKNOWN  <NA>  2019-12-01 12:27:02
2                <NA>        <NA>  UNKNOWN  <NA>  2019-12-01 12:27:02
3                <NA>        <NA>  UNKNOWN  <NA>  2019-12-01 12:27:02
4                <NA>        <NA>  UNKNOWN  <NA>  2019-12-01 12:27:02
…                 …          …       …   …                 …
11461352  electronics  smartphone  UNKNOWN  <NA>  2019-11-30 16:39:49
11461353  electronics  smartphone  UNKNOWN  <NA>  2019-11-30 16:39:55
11461354  electronics  smartphone  UNKNOWN  <NA>  2019-11-30 16:39:56
11461355  electronics  smartphone  UNKNOWN  <NA>  2019-11-30 16:40:00
11461356  electronics  smartphone  UNKNOWN  <NA>  2019-11-30 16:40:04

          ts_hour  ts_minute  ts_weekday  ts_day  ts_month  ts_year  \
0              12         27           6       1        12     2019
1              12         27           6       1        12     2019
2              12         27           6       1        12     2019
3              12         27           6       1        12     2019
4              12         27           6       1        12     2019
…             …          …          …       …        …       …
11461352       16         39           5      30        11     2019
11461353       16         39           5      30        11     2019
11461354       16         39           5      30        11     2019
11461355       16         40           5      30        11     2019
11461356       16         40           5      30        11     2019

          CE_product_id
0                   881
1                   881
2                   881
3                   881
4                   881
…                    …
11461352          82398
11461353         251189
11461354         105730
11461355         180072
11461356          30286

[11461357 rows x 20 columns]
```

Similar, we can apply *Count Encoding* to a group of categorical features.

[8]:
```
ce = df_train[['cat_2', 'brand', 'target']].groupby(['cat_2', 'brand']).
↪agg(['count'])
```

[9]:
```
ce
```

[9]:
```
                    target
                     count
cat_2   brand
UNKNOWN UNKNOWN     521515
        a-case         367
        a-derma         57
        a-elita         22
        a-mega          25
…                      …
winch   tutti           31
        vichy           12
        viteks          14
        woodcraft       21
        yvesrocher      11

[11154 rows x 1 columns]
```

[10]:
```
ce = ce.reset_index()
ce.columns = ['cat_2', 'brand', 'CE_cat_2_brand']
df_train.merge(ce, how='left', left_on=['cat_2', 'brand'], right_on=['cat_2',␣
↪'brand'])
```

[10]:
```
                       event_time event_type  product_id     brand   price  \
0         2019-12-01 07:04:06 UTC       cart   100008496   respect   95.24
1         2019-12-01 07:04:07 UTC       cart     5100816    xiaomi   32.18
2         2019-12-01 07:04:09 UTC       cart    12704744    nokian  129.27
3         2019-12-01 07:04:09 UTC       cart     1005212   samsung  168.86
4         2019-12-01 07:04:10 UTC       cart     3601405      beko  180.16
…                             …          …           …       …      …
11461352  2019-11-30 17:06:47 UTC   purchase     1005115     apple  915.49
11461353  2019-11-30 17:06:50 UTC   purchase     1005161    xiaomi  191.56
11461354  2019-11-30 17:06:53 UTC   purchase    11100284  scarlett    6.67
11461355  2019-11-30 17:06:53 UTC   purchase     2900852  dauscher   60.49
11461356  2019-11-30 17:06:54 UTC   purchase    12711054     tunga   34.75

            user_id                            user_session  target  \
0         529320958  ba3ecb00-0a81-480e-a35a-caee3ccc4b6e       0
1         554214170  06737067-47b5-4219-8bc7-b9c1fc017e74       0
2         519798261  24472c68-b11e-4187-8603-cd71d14ddfba       0
3         554551310  51d86227-dc05-4784-9a70-c6d45da1a17f       0
```

```
4          513516750  fc2a04a8-8eaf-4727-8785-a6ae160ab9eb       0
…              …                                        …     …
11461352   579789462  b5411c53-e888-4fc7-9042-5b647659a5ab       1
11461353   531355728  36618eb8-4718-4abd-926f-b161f682d226       1
11461354   562924883  c05b8b4a-20a9-44a1-9a15-69bac0869c57       1
11461355   575788127  4efcc103-3d11-44fe-af39-cbccd523d8d1       1
11461356   518767884  5ee565b9-7d98-4839-8215-026ff6aa6111       1

                cat_0       cat_1      cat_2 cat_3            timestamp  \
0             apparel       shoes    UNKNOWN  <NA>  2019-12-01 07:04:06
1             apparel       shoes    UNKNOWN  <NA>  2019-12-01 07:04:07
2                <NA>        <NA>    UNKNOWN  <NA>  2019-12-01 07:04:09
3          construction       tools      light  <NA>  2019-12-01 07:04:09
4            appliances     kitchen     washer  <NA>  2019-12-01 07:04:10
…                 …         …        …   …                 …
11461352   electronics  smartphone    UNKNOWN  <NA>  2019-11-30 17:06:47
11461353   electronics  smartphone    UNKNOWN  <NA>  2019-11-30 17:06:50
11461354    appliances    personal     scales  <NA>  2019-11-30 17:06:53
11461355    appliances     kitchen  microwave  <NA>  2019-11-30 17:06:53
11461356          <NA>        <NA>    UNKNOWN  <NA>  2019-11-30 17:06:54

           ts_hour  ts_minute  ts_weekday  ts_day  ts_month  ts_year  \
0                7          4           6       1        12     2019
1                7          4           6       1        12     2019
2                7          4           6       1        12     2019
3                7          4           6       1        12     2019
4                7          4           6       1        12     2019
…              …         …          …      …        …       …
11461352        17          6           5      30        11     2019
11461353        17          6           5      30        11     2019
11461354        17          6           5      30        11     2019
11461355        17          6           5      30        11     2019
11461356        17          6           5      30        11     2019

           CE_cat_2_brand
0                   28562
1                  483359
2                   40060
3                 1212393
4                   21073
…                   …
11461352           880314
11461353           483359
11461354             1310
11461355             1051
11461356            23337
```

```
[11461357 rows x 20 columns]
```

Count Encoding creates a new feature, which can be used by the model for training. It groups categorical values based on the frequency together. For example,

users, which have only 1 interaction in the datasets, are encoded with 1. Instead of having 1 datapoint per user, now, the model can learn a behavior pattern of these users at once.

products, which have many interactions in the datasets, are encoded with a high number. The model can learn to see them as top sellers and treat them, accordingly. The advantage of Count Encoding is that the category values are grouped together based on behavior. Particularly in cases with only a few observation, a decision tree is not able to create a split and neural networks have only a few gradient descent updates for these values.

Summary Count Encoding calculates frequency of categories

The model is trained based on these frequencies

Note In competition, we could count encode the categories for the datasets in different ways:

Count Encode the training dataset and apply it to the validation dataset

Count Encode the training dataset and Count Encode the validataion dataset, separatly

Merge the training dataset and validation dataset, Count Encode the concatenated dataset and apply to both datasets Our focus is on industry applications, therefore only the first process is a valid real-world solution. We may can collect statistics as a stream and update the characteristic of our dataset, but it is probably cleaner to increase the training frequency of our recommender models.

2.3 Practice

Now, it is your turn. Let's try to implement *Count Encoding* as a function. You can either use pandas, dask or cudf.

[11]:
```
col = 'user_id'
```

[12]:
```
### ToDo
```

[13]:
```
############### Solution ###############
```

[15]:
```
############### Solution End ###########
```

2.4 Optimization

Let's compare the runtime between pandas and cuDF. The implementation depends only on the DataFrame object (calling function of the object) and does not require any pd / cuDF function. Therefore, we can use the same implementation and just use pandas.DataFrame and cuDF.DataFrame.

We restart the kernel.

[16]:
```
app = IPython.Application.instance()
app.kernel.do_shutdown(True)
```

[16]:
```
{'status': 'ok', 'restart': True}
```

[2]:
```
import IPython

import pandas as pd
import cudf
import numpy as np
import cupy
import matplotlib.pyplot as plt

df_train = cudf.read_parquet('./data/train.parquet')
df_valid = cudf.read_parquet('./data/valid.parquet')

df_train['brand'] = df_train['brand'].fillna('UNKNOWN')
df_valid['brand'] = df_valid['brand'].fillna('UNKNOWN')
df_train['cat_2'] = df_train['cat_2'].fillna('UNKNOWN')
df_valid['cat_2'] = df_valid['cat_2'].fillna('UNKNOWN')
```

[3]:
```
df_train_pd = df_train.to_pandas()
df_valid_pd = df_valid.to_pandas()
```

[4]:
```
def count_encode(train, valid, col, gpu=True):
    """
        train:  train dataset
        valid:  validation dataset
        col:    column which will be count encoded (in the example RESOURCE)
    """
    # We keep the original order as cudf merge will not preserve the original␣
 ↪order
    if gpu:
        train['org_sorting'] = cupy.arange(len(train), dtype="int32")
    else:
        train['org_sorting'] = np.arange(len(train), dtype="int32")

    train_tmp = train[col].value_counts().reset_index()
    train_tmp.columns = [col,  'CE_' + col]
    df_tmp = train[[col, 'org_sorting']].merge(train_tmp, how='left',␣
 ↪left_on=col, right_on=col).sort_values('org_sorting')
    train['CE_' + col] = df_tmp['CE_' + col].fillna(0).values

    if gpu:
        valid['org_sorting'] = cupy.arange(len(valid), dtype="int32")
    else:
        valid['org_sorting'] = np.arange(len(valid), dtype="int32")
```

```
    df_tmp = valid[[col, 'org_sorting']].merge(train_tmp, how='left',␣
↪left_on=col, right_on=col).sort_values('org_sorting')
    valid['CE_' + col] = df_tmp['CE_' + col].fillna(0).values

    valid = valid.drop('org_sorting', axis=1)
    train = train.drop('org_sorting', axis=1)
    return(train, valid)
```

[5]:
```
%%time

df_train_pd, df_valid_pd = count_encode(df_train_pd, df_valid_pd, 'user_id',␣
↪gpu=False)
```

```
CPU times: user 6.91 s, sys: 2.77 s, total: 9.68 s
Wall time: 9.67 s
```

[6]:
```
%%time

df_train, df_valid = count_encode(df_train, df_valid, 'user_id', gpu=True)
```

```
CPU times: user 272 ms, sys: 272 ms, total: 544 ms
Wall time: 542 ms
```

In our experiments, we achieve a speed up of 15.8x.

Our implementation can be still improved. We will show a further optimized solution based on dask and dask_cudf.

We shutdown the kernel.

[7]:
```
app = IPython.Application.instance()
app.kernel.do_shutdown(False)
```

[7]:
```
{'status': 'ok', 'restart': False}
```