

```
In [1]: import sys
sys.path.append('../deepsurv')
import deep_surv

from deepsurv_logger import DeepSurvLogger, TensorboardLogger
import utils
import viz

import numpy as np
import pandas as pd

import lasagne
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

```
/Users/timfingerhut/anaconda3/lib/python3.6/site-packages/h5py/__i
nit__.py:36: FutureWarning: Conversion of the second argument of i
ssubdtype from `float` to `np.floating` is deprecated. In future,
it will be treated as `np.float64 == np.dtype(float).type`.
    from ._conv import register_converters as _register_converters
```

```
In [2]: train_dataset_fp = './2006_final.csv'
train_df = pd.read_csv(train_dataset_fp)
train_df.head()
```

Out[2]:

	fail_1	dur	country1	country2	country4	country5	country6	country7	country8	count
0	1	30	1	0	0	0	0	0	0	
1	1	1	1	0	0	0	0	0	0	
2	1	60	1	0	0	0	0	0	0	
3	1	60	1	0	0	0	0	0	0	
4	1	45	1	0	0	0	0	0	0	

5 rows × 67 columns

```

In [3]: # event_col is the header in the df that represents the 'Event / Status' indicator
# time_col is the header in the df that represents the event time
def dataframe_to_deepsurv_ds(df, event_col = 'fail_1', time_col = 'dur'):
    # Extract the event and time columns as numpy arrays
    e = df[event_col].values.astype(np.int32)
    t = df[time_col].values.astype(np.float32)

    # Extract the patient's covariates as a numpy array
    x_df = df.drop([event_col, time_col], axis = 1)
    x = x_df.values.astype(np.float32)

    # Return the deep surv dataframe
    return {
        'x' : x,
        'e' : e,
        't' : t
    }

# If the headers of the csv change, you can replace the values of
# 'event_col' and 'time_col' with the names of the new headers
# You can also use this function on your training dataset, validation dataset, and testing dataset
train_data = dataframe_to_deepsurv_ds(train_df, event_col = 'fail_1', time_col = 'dur')

```

```

In [4]: hyperparams = {
    'L2_reg': 10.0,
    'batch_norm': True,
    'dropout': 0.4,
    'hidden_layers_sizes': [100, 100],
    'learning_rate': 3.3e-2,
    'lr_decay': 0.001,
    'momentum': 0.9,
    'n_in': train_data['x'].shape[1],
    'standardize': True
}

```

```
In [5]: # Create an instance of DeepSurv using the hyperparams defined above
model = deep_surv.DeepSurv(**hyperparams)

# DeepSurv can now leverage TensorBoard to monitor training and validation
# This section of code is optional. If you don't want to use the tensorboard logger
# Uncomment the below line, and comment out the other three lines:
# logger = None

experiment_name = 'test_experiment_tim'
logdir = './logs/tensorboard/'
logger = TensorboardLogger(experiment_name, logdir=logdir)

# Now we train the model
update_fn=lasagne.updates.nesterov_momentum # The type of optimizer to use. \
                                             # Check out http://lasagne.readthedocs.io/en/latest/modules/updates.html \
                                             # for other optimizers to use
n_epochs = 2000

# If you have validation data, you can add it as the second parameter to the function
metrics = model.train(train_data, n_epochs=n_epochs, logger=logger, update_fn=update_fn)
```

```
2019-07-10 14:28:16,929 - Training step 0/2000 |
| - loss: 29.4952 - ci: 0.4389
2019-07-10 14:29:55,203 - Training step 250/2000 | ***
| - loss: 6.7815 - ci: 0.7085
2019-07-10 14:31:31,550 - Training step 500/2000 | *****
| - loss: 6.7372 - ci: 0.7249
2019-07-10 14:33:15,260 - Training step 750/2000 | *****
| - loss: 6.7940 - ci: 0.7346
2019-07-10 14:34:52,526 - Training step 1000/2000 | *****
| - loss: 6.8138 - ci: 0.2592
2019-07-10 14:36:30,074 - Training step 1250/2000 | *****
| - loss: 6.7759 - ci: 0.4419
2019-07-10 14:38:07,394 - Training step 1500/2000 | *****
*** | - loss: 6.8324 - ci: 0.2463
2019-07-10 14:39:42,858 - Training step 1750/2000 | *****
***** | - loss: 6.7457 - ci: 0.7596
2019-07-10 14:41:20,638 - Finished Training with 2000 iterations in 784.07s
```

```
In [6]: # Print the final metrics
print('Train C-Index:', metrics['c-index'][-1])
# print('Valid C-Index: ', metrics['valid_c-index'][-1])

# Plot the training / validation curves
viz.plot_log(metrics)
```

Train C-Index: (1999, 0.7679167720784831)

