In [12]:
```
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
```

In [13]:
```
train_dataset_fp = './2006_final.csv'
train_df = pd.read_csv(train_dataset_fp)
train_df.head()
```

Out[13]:

	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 [14]:
```
# event_col is the header in the df that represents the 'Event / St
atus' 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, validati
on dataset, and testing dataset
train_data = dataframe_to_deepsurv_ds(train_df, event_col = 'fail_1
', time_col= 'dur')
```

In [15]:
```
hyperparams = {
    'L2_reg': 10.0,
    'batch_norm': True,
    'dropout': 0.4,
    'hidden_layers_sizes': [100, 100],
    'learning_rate': 1e00, #1
    'lr_decay': 0.001,
    'momentum': 0.9,
    'n_in': train_data['x'].shape[1],
    'standardize': True
}
```

In [16]:

```
# Create an instance of DeepSurv using the hyperparams defined abov
e
model = deep_surv.DeepSurv(**hyperparams)

# DeepSurv can now leverage TensorBoard to monitor training and val
idation
# This section of code is optional. If you don't want to use the te
nsorboard 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://lasa
gne.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 paramet
er to the function
metrics = model.train(train_data, n_epochs=n_epochs, logger=logger,
update_fn=update_fn)
```

```
2019-07-09 18:53:02,101 - Training step 0/2000    |
| - loss: 27.5550 - ci: 0.4754
2019-07-09 18:53:02,101 - Training step 0/2000    |
| - loss: 27.5550 - ci: 0.4754
2019-07-09 18:53:02,101 - Training step 0/2000    |
| - loss: 27.5550 - ci: 0.4754
2019-07-09 18:54:20,043 - Training step 250/2000  |***
| - loss: nan - ci: 0.5000
2019-07-09 18:54:20,043 - Training step 250/2000  |***
| - loss: nan - ci: 0.5000
2019-07-09 18:54:20,043 - Training step 250/2000  |***
| - loss: nan - ci: 0.5000
2019-07-09 18:55:37,023 - Training step 500/2000  |******
| - loss: nan - ci: 0.5000
2019-07-09 18:55:37,023 - Training step 500/2000  |******
| - loss: nan - ci: 0.5000
2019-07-09 18:55:37,023 - Training step 500/2000  |******
| - loss: nan - ci: 0.5000
2019-07-09 18:57:08,327 - Training step 750/2000  |*********
| - loss: nan - ci: 0.5000
2019-07-09 18:57:08,327 - Training step 750/2000  |*********
| - loss: nan - ci: 0.5000
2019-07-09 18:57:08,327 - Training step 750/2000  |*********
| - loss: nan - ci: 0.5000
2019-07-09 18:59:18,786 - Training step 1000/2000 |************
| - loss: nan - ci: 0.5000
2019-07-09 18:59:18,786 - Training step 1000/2000 |************
| - loss: nan - ci: 0.5000
2019-07-09 18:59:18,786 - Training step 1000/2000 |************
| - loss: nan - ci: 0.5000
2019-07-09 19:00:54,564 - Training step 1250/2000 |***************
| - loss: nan - ci: 0.5000
2019-07-09 19:00:54,564 - Training step 1250/2000 |***************
| - loss: nan - ci: 0.5000
2019-07-09 19:00:54,564 - Training step 1250/2000 |***************
| - loss: nan - ci: 0.5000
2019-07-09 19:02:27,534 - Training step 1500/2000 |***************
***       | - loss: nan - ci: 0.5000
2019-07-09 19:02:27,534 - Training step 1500/2000 |***************
***       | - loss: nan - ci: 0.5000
2019-07-09 19:02:27,534 - Training step 1500/2000 |***************
***       | - loss: nan - ci: 0.5000
2019-07-09 19:04:02,221 - Training step 1750/2000 |***************
******    | - loss: nan - ci: 0.5000
2019-07-09 19:04:02,221 - Training step 1750/2000 |***************
******    | - loss: nan - ci: 0.5000
2019-07-09 19:04:02,221 - Training step 1750/2000 |***************
******    | - loss: nan - ci: 0.5000
2019-07-09 19:05:36,640 - Finished Training with 2000 iterations i
n 755.03s
2019-07-09 19:05:36,640 - Finished Training with 2000 iterations i
n 755.03s
2019-07-09 19:05:36,640 - Finished Training with 2000 iterations i
n 755.03s
```

In [17]:
```
# 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, 0.5)
```