LSTM-FCN AA Classifier Results

Mike Schaid

```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

```
from comet_ml import API, APIExperiment
from dotenv import load_dotenv
import os
from pathlib import Path
import matplotlib.pyplot as plt
import seaborn as sns
import polars as pl
import numpy as np
from great_tables import GT
import polars.selectors as cs
from transphorm.experiments.aa_classifiers.aa_lstmfcn_bayes_5_day import load_data
```

```
from sktime.classification.deep_learning.lstmfcn import LSTMFCNClassifier
```

```
load_dotenv()
key = os.getenv("COMET_API_KEY")
figures_path = Path('/Users/mds8301/Library/CloudStorage/OneDrive-NorthwesternUniversity/g
api = API(key)
version = "1.0.0"
exp_key = 'eaf5e79cc3044f37b7949b495d5dfd9d'
exp = api.get_experiment_by_key(exp_key)
```

```
# api = API()
# api.download_registry_model("mschaid", "aa_classifier_lopez_2024", version="1.0.0", outp
```

Model Parameters

I think these can go in supplementry

```
model_params = exp.get_parameters_summary()
params = (pl.DataFrame(model_params)
          .select(['name', 'valueCurrent'])
          .rename({'name': 'parameter', 'valueCurrent':'value'})
          .with_columns(pl.col('parameter').str.replace("_", ' ', n = -1))
          .to_pandas()
)
params.to_html(figures_path/'params.html', index = False)
params
```

	parameter	value
0	adam amsgrad	false
1	adam beta 1	0.9
2	adam beta 2	0.999
3	adam clipnorm	null
4	adam clipvalue	null
5	adam ema momentum	0.99
6	adam ema overwrite frequency	null
7	adam epsilon	1.0E-7
8	adam global clipnorm	null
9	adam gradient accumulation steps	null
10	adam learning rate	0.0010000000474974513
11	adam loss scale factor	null
12	adam use ema	false
13	adam weight decay	null
14	attention	false
15	batch size	128
16	callbacks	null
17	categories	auto
18	drop	null
19	dropout	0.2
20	dtype	<class 'numpy.float64'>

	parameter	value
21	epochs	2000
22	feature name combiner	concat
23	filter sizes	[128,256,128]
24	handle unknown	error
25	kernel sizes	[16,10,6]
26	lstm size	10
27	max categories	null
28	min frequency	null
29	n epochs	2000
30	Optimizer	adam
31	random state	42
32	sparse output	false
33	verbose	0

Validation Metrics

- show model parameters with accuracy being the most easy to interpret

```
def get_validation_metrics(exp:APIExperiment)-> pl.DataFrame:

    all_metrics = exp.get_metrics_summary()
    validation_metrics = [m for m in all_metrics if any(s in m['name'] for s in ['train',
    metrics_df = (
        pl.DataFrame(validation_metrics)
        .select(['name', 'valueCurrent'])
        .with_columns(
            pl.col('name').str.contains('train').alias('is_train'),
            pl.col('name').str.replace("train_", '')
            .str.replace("test_", '')
            .str.replace('_', ' ',n = -1)
            .str.replace('weighted', '(Weighted)')
            .str.to_titlecase()
            .str.replace('Roc Auc', 'ROC-AUC')
            .str.replace('Balanced Accuracy', 'Accuracy (weighted)')
            .alias('Metric'),
            pl.col('valueCurrent').cast(pl.Float32).round(4).alias('Value'))
        .with_columns(pl.when(pl.col('is_train')==True)
                    .then(pl.lit('Train'))
                    .otherwise(pl.lit('Test'))
```

```
                    .alias('Dataset'))
            .drop(['is_train', 'name', 'valueCurrent'])

    )
    return metrics_df
metrics = get_validation_metrics(exp)
agg_df = (metrics
 .sort('Metric')
 .to_pandas()
 .groupby(['Metric', 'Dataset']).mean()
#  .to_html(figures_path/'metrics.html')
)
agg_df
```

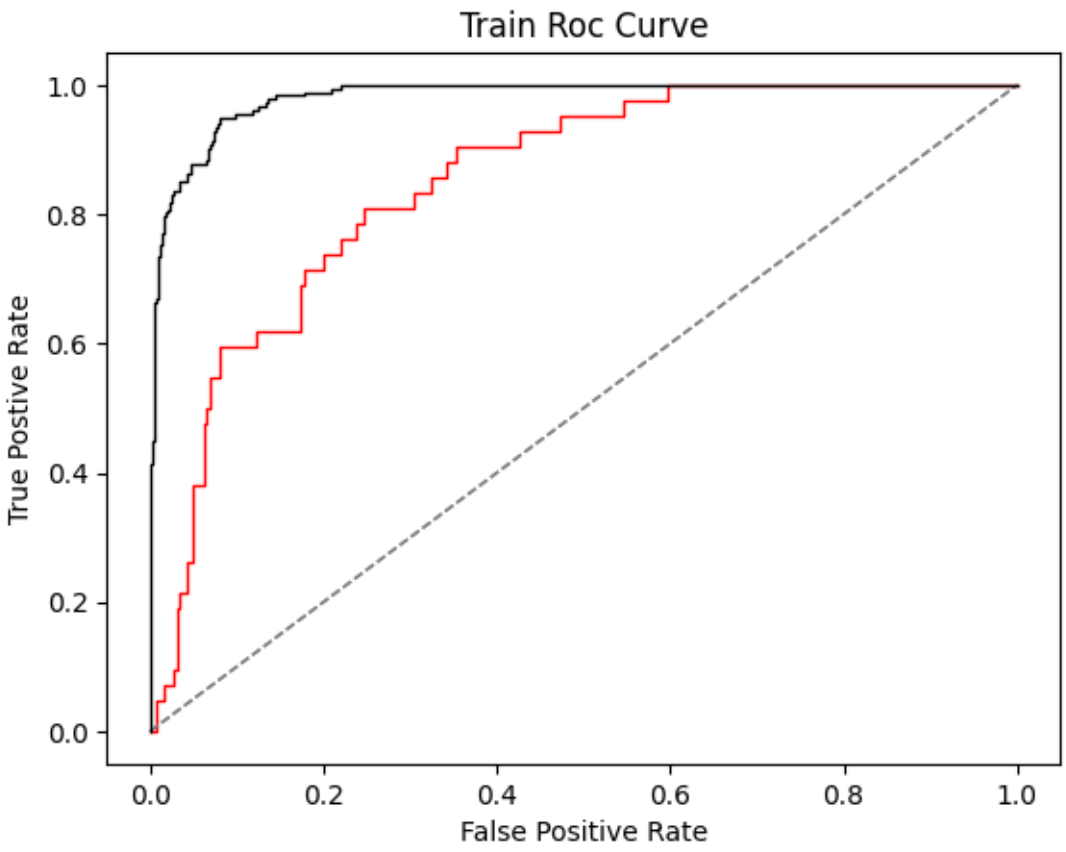
Metric	Dataset	Value
Accuracy	Test	0.8742
	Train	0.9426
Accuracy (weighted)	Test	0.7273
	Train	0.8798
F1 Score	Test	0.5366
	Train	0.8443
F1 Score (Weighted)	Test	0.8729
	Train	0.9406
Precision	Test	0.5500
	Train	0.9276
Precision (Weighted)	Test	0.8717
	Train	0.9420
ROC-AUC	Test	0.8512
	Train	0.9821
Recall	Test	0.5238
	Train	0.7747
Recall (Weighted)	Test	0.8742
	Train	0.9426

ROC Curves

- standard for showing model is better than random
- technically this model is overfitting to a degree but this is a result of the

- we can try to do more - ie over sample minority (escape) class or add more penality
- We can only show the test ROC curve to show it's better than random

```
curves =  exp.get_curves()
train = curves[0]
test = curves[1]
def plt_roc(curve, c= 'r'):
    x = curve['x']
    y = curve['y']
    title = curve['name']
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Postive Rate')
    plt.plot([0, 1], [0, 1], linestyle='--', lw=1, color='grey', label='Random classifier'
    plt.plot(x, y, lw = 1, color = c)
    plt.title(title.title())
# plt_roc(train)
plt_roc(test)
plt_roc(train, 'k')
```




```
path = Path('/Users/mds8301/Desktop/temp/over_day_5/eval_data/inference_results.npz')
results = np.load(path)
x_test = results['x_test']
y_test = results['y_test']
y_test_pred = results['y_test_pred']
```

```
def make_label_df(y_true, y_pred):

    label_df = (
        pl.DataFrame({'y_true': y_true, "y_pred": y_pred})
        .with_columns(
            pl.Series('trial', np.arange(0, y_true.shape[0])),
            pl.when(pl.col('y_true')==1.0).then(pl.lit('Escape')).otherwise(pl.lit('Avoid'
             pl.when(pl.col('y_pred')==1.0).then(pl.lit('Escape')).otherwise(pl.lit('Avoid
        )
```

```
        .drop(['y_true', 'y_pred'])
        )
    return label_df
true_labels = make_label_df(y_test, y_test_pred)
```

```
x_test_ds = x_test.reshape(-1, x_test.shape[-1])[:,::25]
```

```
time = np.linspace(-1,5, x_test_ds.shape[-1])
traces = (pl.DataFrame(x_test_ds.T)
          .with_columns(pl.Series('time', time))
.melt(id_vars =['time'], variable_name = 'trial')
.with_columns(pl.col('trial').str.replace('column_','').cast(pl.Int64))
)

all_df = traces.join(true_labels, on = 'trial')
all_df
```

```
/var/folders/_3/4x4mtlsd3n37vfrcjmsz1vcd8clmkl/T/ipykernel_31392/3266360539.py:4: Deprecatio
  .melt(id_vars =['time'], variable_name = 'trial')
```

time f64	trial i64	value f32	ground_truth str	predicted str
-1.0	0	-1.459656	”Avoid”	”Avoid”
-0.97541	0	-1.490074	”Avoid”	”Avoid”
-0.95082	0	-1.518508	”Avoid”	”Avoid”
-0.92623	0	-1.540291	”Avoid”	”Avoid”
-0.901639	0	-1.552294	”Avoid”	”Avoid”
…	…	…	…	…
4.901639	301	-0.478533	”Escape”	”Avoid”
4.92623	301	-0.505216	”Escape”	”Avoid”
4.95082	301	-0.524392	”Escape”	”Avoid”
4.97541	301	-0.535673	”Escape”	”Avoid”
5.0	301	-0.540014	”Escape”	”Avoid”

Visualizing true vs predicted lables

- indisict, but you can see the difference in the escape which I think is good

```
fig, ax = plt.subplot_mosaic(mosaic="""AB""")
sns.lineplot(all_df, x='time', y='value', hue='ground_truth', ax=ax['A'])
sns.lineplot(all_df, x='time', y='value', hue='predicted', ax=ax['B'], linestyle ='--')
sns.despine()
plt.tight_layout()
```