vis_risk_score_feature_distribution

June 6, 2024

[ ]:
```
import os
import sys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import torch
import glob
from pathlib import Path

sys.path.append(os.path.abspath(os.path.join('../..')))
from train_risk_regression_model_with_recon_task import get_dataset, get_model,␣
 ↪DL_single_run
```

[ ]:
```
## initialize dataset
task_name ="HYP_with_HF_event"
X,y =get_dataset(dataset_name = task_name)
```

```
/home/engs2522/project/LLM-ECG-Dual-Attention
input ecg shape (11575, 12, 608)
status, duration, eid (11575, 3)
```

[ ]:
```
## plot risk score distribution
from sklearn.model_selection import train_test_split, StratifiedKFold
kf = StratifiedKFold(n_splits=2, shuffle=True, random_state=42)
# Initialize lists to store predictions
val_c_index_list = []
y_status_list = y[:,0]
i =0
```

[ ]:
```
y_status_list = y[:,0]
seed_list = [42]
latent_code_dim =512
seed_y_score_list = []
risk_scores = []
for seed in seed_list:
    total_feature_score=[]
    y_total_test=[]
```

```
  test_indices_list=[]
  total_risk_score=[]
  for cval, (train_indices, test_indices) in enumerate(kf.split(X,␣
↪y_status_list)):
      x_train, y_train = X[train_indices], y[train_indices]
      x_test, y_test = X[test_indices], y[test_indices]
          ## find the best model path:
      project_root_path = Path(os.path.abspath("__file__")).parents[2]

      model_dir = os.path.join(project_root_path, f"result/
↪train_survival_net_{task_name}_0.5/
↪ECG_attention_pretrained_on_recon_ECG2Text_512/{seed}/cval_{cval}/")
      print(model_dir)
      best_model_path_list =glob.glob(model_dir+"best_model*_lr_*.pth")
      ## remove path with alpha
      if len(best_model_path_list)==0:
          raise ValueError("No model found")
      else:
          if len(best_model_path_list) >1:
              print (best_model_path_list)

              best_model_path_list = [x for x in best_model_path_list if␣
↪"alpha" not in x]
              c_index_list = [float((x.split("/")[-1]).split("_")[4]) for x␣
↪in best_model_path_list]
              highest_one = np.argmax(c_index_list)
              best_model_path  = best_model_path_list[highest_one]
          else:
              best_model_path = best_model_path_list[0]
      print(best_model_path)
      trainer, survival_model = DL_single_run(x_train, y_train,  model_name =␣
↪"ECG_attention",
                  batch_size = 200,
                  train_from_scratch=True,
                  freeze_encoder=False, test_only=True,
                  test_checkpoint_path =␣
↪best_model_path,latent_code_dim=latent_code_dim)
      survival_model.freeze()
      X_test= torch.from_numpy(x_test).float().to(survival_model.device)

      with torch.inference_mode():
          log_risk_score,_ = survival_model(X_test)
          ## get last layer hidden feature for visualization
          encoder_feature = survival_model.encoder(X_test)
          last_hidden = survival_model.downsteam_net.
↪get_features(encoder_feature)
```

```
            print ("last hidden shape", last_hidden.shape)


        total_feature_score.append(last_hidden.cpu().detach().numpy())
        total_risk_score.append(log_risk_score.cpu().detach().numpy())
        test_indices_list.append(test_indices)
    total_feature_score_flatten = np.concatenate(total_feature_score)
    test_indices_flatten = np.concatenate(test_indices_list)
    total_risk_score_flatten = np.concatenate(total_risk_score)
    ## sort the risk score back to the original order
    total_feature_score_sorted = total_feature_score_flatten[np.
↪argsort(test_indices_flatten)]
    total_risk_score_sorted = total_risk_score_flatten[np.
↪argsort(test_indices_flatten)]
```

```
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
```

```
/home/engs2522/project/LLM-ECG-Dual-Attention/result/train_survival_net_HYP_with
_HF_event_0.5/ECG_attention_pretrained_on_recon_ECG2Text_512/42/cval_0/
['/home/engs2522/project/LLM-ECG-Dual-Attention/result/train_survival_net_HYP_wi
th_HF_event_0.5/ECG_attention_pretrained_on_recon_ECG2Text_512/42/cval_0/best_mo
del_c_index_0.6033_lr_2.238721138568339e-05.pth', '/home/engs2522/project/LLM-
ECG-Dual-Attention/result/train_survival_net_HYP_with_HF_event_0.5/ECG_attention
_pretrained_on_recon_ECG2Text_512/42/cval_0/best_model_c_index_0.6032_lr_2.23872
1138568339e-05.pth']
/home/engs2522/project/LLM-ECG-Dual-Attention/result/train_survival_net_HYP_with
_HF_event_0.5/ECG_attention_pretrained_on_recon_ECG2Text_512/42/cval_0/best_mode
l_c_index_0.6033_lr_2.238721138568339e-05.pth
no linear layer
last hidden shape torch.Size([5788, 3])
/home/engs2522/project/LLM-ECG-Dual-Attention/result/train_survival_net_HYP_with
_HF_event_0.5/ECG_attention_pretrained_on_recon_ECG2Text_512/42/cval_1/
['/home/engs2522/project/LLM-ECG-Dual-Attention/result/train_survival_net_HYP_wi
th_HF_event_0.5/ECG_attention_pretrained_on_recon_ECG2Text_512/42/cval_1/best_mo
del_c_index_0.6195_lr_7.943282347242813e-07.pth', '/home/engs2522/project/LLM-
ECG-Dual-Attention/result/train_survival_net_HYP_with_HF_event_0.5/ECG_attention
_pretrained_on_recon_ECG2Text_512/42/cval_1/best_model_c_index_0.6178_lr_7.94328
2347242813e-07.pth']
/home/engs2522/project/LLM-ECG-Dual-Attention/result/train_survival_net_HYP_with
_HF_event_0.5/ECG_attention_pretrained_on_recon_ECG2Text_512/42/cval_1/best_mode
l_c_index_0.6195_lr_7.943282347242813e-07.pth
```

```
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
```

```
HPU available: False, using: 0 HPUs

no linear layer
last hidden shape torch.Size([5787, 3])
```

[ ]:
```
## make a data frame with eid, status, time-to-event, predicted latent feature␣
 ↪axis 1, axis 2, axis 3, predicted risk score
list_of_elements =[y[:,2],y[:,0],y[:,1], total_feature_score_sorted[:
 ↪,0],total_feature_score_sorted[:,1],total_feature_score_sorted[:
 ↪,2],total_risk_score_sorted[:,0]]
```

[ ]:
```
np.array(list_of_elements)
```

[ ]:
```
array([[ 1.000270e+06,  1.000360e+06,  1.000409e+06,  1.001580e+06, …,
6.025087e+06,  6.025131e+06,  6.025343e+06,
         6.025351e+06],
       [ 0.000000e+00,  0.000000e+00,  0.000000e+00,  0.000000e+00, …,
0.000000e+00,  0.000000e+00,  0.000000e+00,
         0.000000e+00],
       [ 6.963333e+01,  6.430000e+01,  5.430000e+01,  8.810000e+01, …,
4.680000e+01,  5.703333e+01,  4.726667e+01,
         4.246667e+01],
       [-6.894987e-01, -6.894987e-01, -6.558711e-01,  1.589090e+00, …,
-2.623859e-01, -6.558711e-01, -6.558711e-01,
         2.062254e+00],
       [-7.474364e-01, -7.474364e-01,  3.810813e+00, -6.042092e-03, …,
-1.913464e-01, -6.732411e-01, -6.732411e-01,
        -7.474364e-01],
       [ 2.177607e+00, -1.258075e-01, -7.641597e-01, -8.189908e-02, …,
8.534212e-02,  2.027937e+00,  9.556093e-01,
        -2.296621e-01],
       [-1.665063e+00, -1.971956e-01,  4.334621e-01, -2.024217e+00, …,
-1.360261e-01, -9.575278e-01, -2.440594e-01,
        -8.429351e-01]])
```

[ ]:
```
df = pd.DataFrame(np.array(list_of_elements).
 ↪T,columns=["eid","status","time_to_event","axis1","axis2","axis3","risk_score"])
```

[ ]:
```
csv_dir = f"/Data/engs2522/ECGresult/train_survival_net_{task_name}/
 ↪ECG_attention_pretrained_on_recon_ECG2Text/risk_score_distribution.csv"
df.to_csv(csv_dir,index=False)
```

[ ]:
```
import plotly.express as px
fig = px.scatter_3d(df, x='axis1', y='axis2', z='axis3',
                    color='risk_score', opacity=0.7,size_max=10,symbol = "status",␣
 ↪size =␣
 ↪"time_to_event",hover_name="eid",hover_data=["risk_score","time_to_event","status"])
# tight layout
```

```
# fig.update_layout(margin=dict(l=0, r=0, b=0, t=0))
# fig.show()
fig.show()
```

[ ]:
```
## find the one with highest risk score
df["eid"] = df["eid"].apply(int)
highest_risk_score = np.argmax(df[df.status==1]["risk_score"].values)
print(df.iloc[highest_risk_score].eid)
## find the lowest risk score
lowest_risk_score = np.argmin(df["risk_score"].values)
print(df.iloc[lowest_risk_score].eid)
```

```
1010317.0
1835752.0
```

[ ]:
```
df[df.status>1e-6].sort_values(by="risk_score",ascending=False)
```

[ ]:

	eid	status	time_to_event	axis1	axis2	axis3	risk_score
1379	1611715	1.0	19.233333	-0.689499	2.414559	-0.689248	2.936992
3066	2333294	1.0	19.800000	-0.689499	2.364655	-0.689248	2.893193
3343	2458996	1.0	7.033333	-0.689499	2.163456	-0.689248	2.716610
4812	3100512	1.0	24.500000	-0.689499	2.093562	-0.689248	2.655268
6701	3891556	1.0	42.833333	-0.689499	1.561806	-0.689248	2.188572
…	…	…	…	…	…	…	…
5923	3562794	1.0	47.966667	2.402772	3.534876	-0.764160	-2.809075
3385	2477123	1.0	42.133333	1.646598	0.675648	1.209097	-3.015634
2366	2039854	1.0	36.166667	0.939843	-0.747436	3.891441	-3.178753
5898	3550238	1.0	27.900000	1.919151	1.598907	1.169094	-3.376613
9521	5134796	1.0	37.533333	2.930487	-0.118130	1.044550	-4.196670

[162 rows x 7 columns]

[ ]:
```
df[df.status==0].sort_values(by="risk_score",ascending=True)
```

[ ]:

	eid	status	time_to_event	axis1	axis2	axis3	risk_score
1900	1835752	0.0	88.100000	5.382341	1.913948	1.213789	-7.143068
3885	2695493	0.0	88.100000	4.750768	0.493179	1.732850	-6.665068
5167	3252795	0.0	88.100000	3.112210	-0.673241	4.162605	-6.407830
6303	3723211	0.0	88.100000	4.739233	1.600831	0.935798	-6.237706
4517	2979046	0.0	82.600000	4.438241	0.565057	1.511097	-6.190756
…	…	…	…	…	…	…	…
8349	4617425	0.0	59.366667	-0.460010	3.121449	-0.689248	3.498023
1583	1698612	0.0	88.100000	-0.689499	3.123237	-0.689248	3.558964
1476	1656130	0.0	58.633333	-0.689499	3.207524	-0.689248	3.632938
1025	1439717	0.0	37.366667	0.644826	4.011035	-0.689248	3.992931
2833	2231247	0.0	35.400000	-0.172391	3.892036	-0.689248	4.099918

```
[11413 rows x 7 columns]
```

[ ]:
```
df[df.eid ==5363922.0]
```

[ ]:

	eid	status	time_to_event	axis1	axis2	axis3	risk_score
702	5363922	1.0	84.0	-0.745112	-0.826321	-0.769051	1.597204

[ ]:
```
df[df.eid ==3069524.0]
```

[ ]:

	eid	status	time_to_event	axis1	axis2	axis3	risk_score
319	3069524	0.0	58.0	-0.566198	2.150224	2.604295	-2.287223

[ ]:
```
## plot the ECG data
from multi_modal_heart.ECG.ecg_utils import plot_overlapped_multi_lead_signals

ecg_high_risk  = X[702]
ecg_low_risk  = X[319]
plot_overlapped_multi_lead_signals(ecg_high_risk,ecg_low_risk,labels=["high␣
 ↪risk_","low risk_"],color_list=["tab:orange","tab:purple"])
```

[ ]: