hw3

May 10, 2025

[40]:
```
import numpy as np
from matplotlib import pyplot as plt  # import subplots, cm
import pandas as pd
from sklearn.model_selection import GridSearchCV, train_test_split, KFold
from sklearn.feature_selection import VarianceThreshold
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from ISLP import load_data, confusion_table
from sklearn.svm import SVC
from sklearn.metrics import RocCurveDisplay
from ISLP.svm import plot as plot_svm
```

[41]:
```
import warnings
warnings.filterwarnings('ignore')
```

[42]:
```
# load data
df = pd.read_csv('data/gene_data.tsv', sep='\t')
```

[43]:
```
df.head()
```

[43]:
```
   sampleID  X354_s_at  X37387_r_at  X510_g_at  X32274_r_at  X41129_at  \
0      1005   3.191091     6.569342   5.795215     8.577962   6.074154
1      1010   3.014668     6.515097   4.236670     8.298168   4.844150
2      3002   3.064113     6.608859   5.318619     7.942474   6.395045
3      4007   3.340386     6.383065   5.906107     7.888818   6.146822
4      4008   3.616074     6.333751   6.043892     8.327389   5.240616

   X32896_at  X37035_at  X41383_at  X33171_s_at  …  X37218_at  X39053_at  \
0   2.663559   8.202604   9.404608     3.365162  …   4.825550   5.355198
1   2.780224   7.208483   9.776749     3.560249  …   4.035656   4.936842
2   2.823451   8.137416   9.486059     3.464070  …   4.564490   5.627310
3   2.520204   8.459870   9.047895     3.312506  …   7.184797   6.420221
4   2.694700   7.740742   9.605306     3.785565  …   6.837805   6.221688

   X41490_at  X41020_at  X1895_at  X35283_at   X532_at  X34972_s_at  \
0   5.860435   4.722569  4.911335   5.883893  2.767831     5.247684
1   6.438530   4.860410  5.147960   5.981595  2.894704     5.191797
2   4.978981   4.690765  6.604224   6.251940  4.278275     5.194465
```

```
3   7.033036   4.724506  6.153798   5.584744  2.794748     5.175139
4   5.785544   4.988296  4.599638   5.763766  2.800937     5.093404

     X1179_at  y
0    8.825393  1
1    9.292181 -1
2    9.760507  1
3    8.991216 -1
4   10.009073 -1

[5 rows x 2002 columns]
```

[44]:
```
sum(df.isna().sum(axis=1))
# no missing values
```

[44]: 0

1 PCA

[45]:
```
# from sklearn.decomposition import PCA
# import matplotlib.pyplot as plt

# pca = PCA(n_components=2)
# X_pca = pca.fit_transform(scaled_X_train)

# plt.figure(figsize=(6, 5))
# plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_train, cmap='coolwarm', alpha=0.6)
# plt.title("PCA Projection of the Dataset")
# plt.xlabel("PC 1")
# plt.ylabel("PC 2")
# plt.show()
```

2 t-SNE

[46]:
```
# from sklearn.manifold import TSNE

# X_tsne = TSNE(n_components=2, perplexity=30).fit_transform(scaled_X_train)
# plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_train, cmap='coolwarm', alpha=0.6)
# plt.title("t-SNE Projection")
# plt.show()
```

3 Pick model

We have to decide which model to use. We have to have a look at the data to determine whether: 1. The data is linearly separable among classes (however, this is not feasible as we are in more than 3 dimensions) 2.

3.1 Fit SVC

For the simplest model, we have to tune the C (cost) hyperparameter. To this end we perform a 5-fold CV to find the better parameter among a series of values

[47]:
```
X = df.drop(['y', 'sampleID'], axis=1)
y = df['y']
```

[48]:
```
selector = VarianceThreshold(threshold=0.05)

selector.fit_transform(X, y)
valid_variables = selector.get_feature_names_out()
```

[49]:
```
X = df[valid_variables]
y = df['y']
```

Split into train test each with size equal to 50% of the dataset

[50]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5,␣
↪random_state=1)
```

[51]:
```
scaler = StandardScaler()
scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

[52]:
```
param_grid = {'kernel': ['linear'],
              'C': list(range(1, 10))}

linear_SVM = SVC(random_state=1)

grid = GridSearchCV(linear_SVM, param_grid=param_grid,cv=5)
```

[53]:
```
grid.fit(scaled_X_train, y_train)
```

[53]:
```
GridSearchCV(cv=5, estimator=SVC(random_state=1),
             param_grid={'C': [1, 2, 3, 4, 5, 6, 7, 8, 9],
                         'kernel': ['linear']})
```

[54]:
```
best_linear_SVC = grid.best_estimator_

preds = best_linear_SVC.predict(scaled_X_test)

print(classification_report(y_true=y_test, y_pred=preds))
```

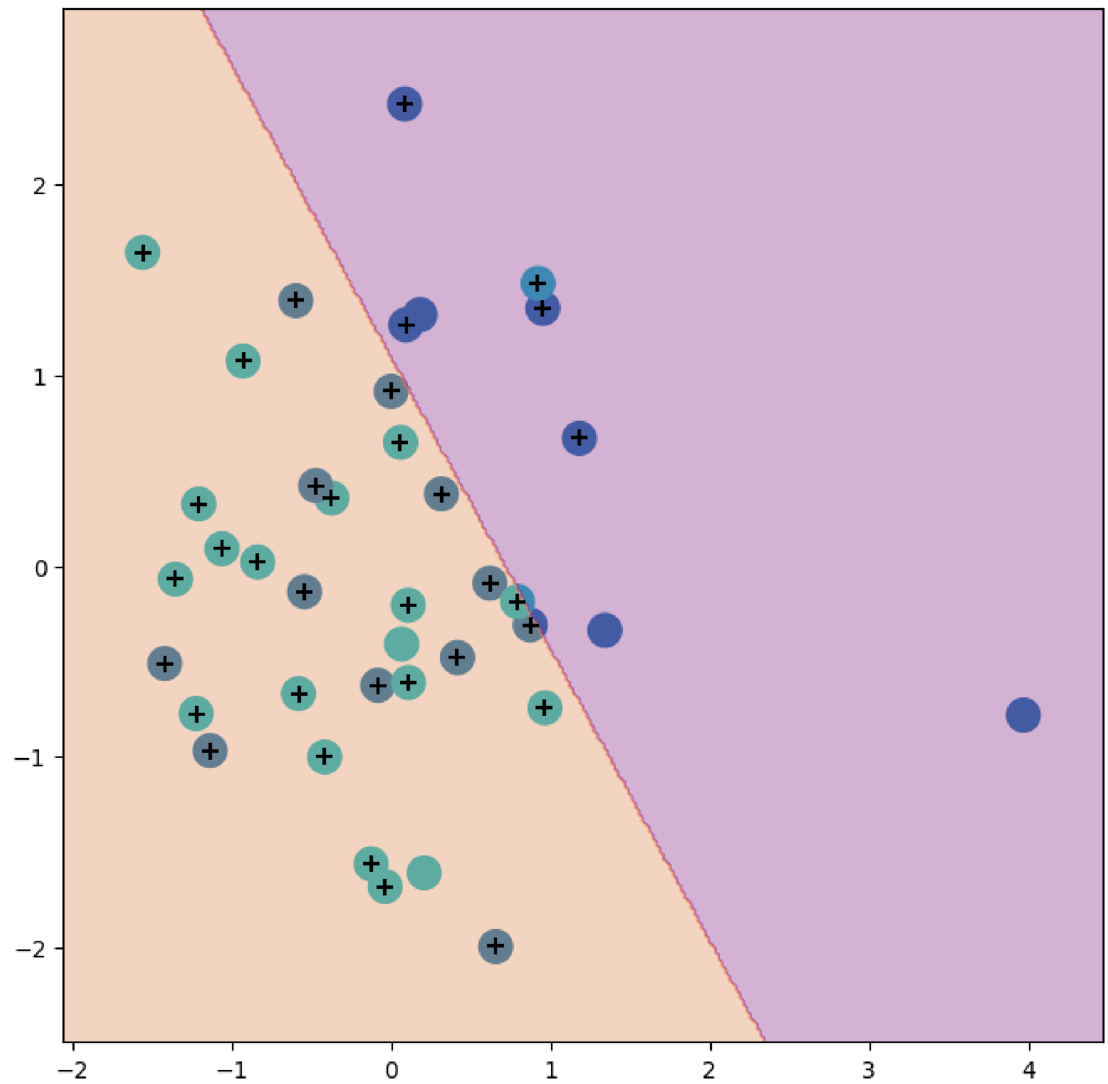
```
              precision    recall  f1-score   support

          -1       0.76      0.57      0.65        23
           1       0.57      0.76      0.65        17

    accuracy                           0.65        40
```

```
   macro avg       0.66      0.66      0.65        40
weighted avg       0.68      0.65      0.65        40
```

[55]:
```
y_train_reset = y_train.reset_index(drop=True)
fig, ax = plt.subplots(figsize=(8, 8))
plot_svm(scaled_X_train, y_train_reset, svm=best_linear_SVC, ax=ax)
```



[56]:
```
confusion_matrix(y_true=y_test, y_pred=preds)
```

[56]:
```
array([[13, 10],
       [ 4, 13]])
```

[57]:
```
(preds == y_test).mean()
```

[57]: np.float64(0.65)

4 Support Vector Machine

Now we try to fit an SVM

[58]:
```
svm_rbf = SVC(kernel="rbf", gamma=1, C=1)
svm_rbf.fit(scaled_X_train, y_train)
```

[58]: SVC(C=1, gamma=1)

[59]:
```
C_range = [0.1, 1, 10, 100, 1000]
gamma_range = [0.5, 1, 2, 3, 4]

params = {"C": C_range, "gamma": gamma_range}

kfold = KFold(5, random_state=0, shuffle=True)

grid = GridSearchCV(svm_rbf, param_grid=params, refit=True, cv=kfold,␣
 ↪scoring="accuracy")
```

[60]:
```
grid.fit(scaled_X_train, y_train)
```

[60]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=0, shuffle=True),
             estimator=SVC(C=1, gamma=1),
             param_grid={'C': [0.1, 1, 10, 100, 1000],
                         'gamma': [0.5, 1, 2, 3, 4]},
             scoring='accuracy')
```

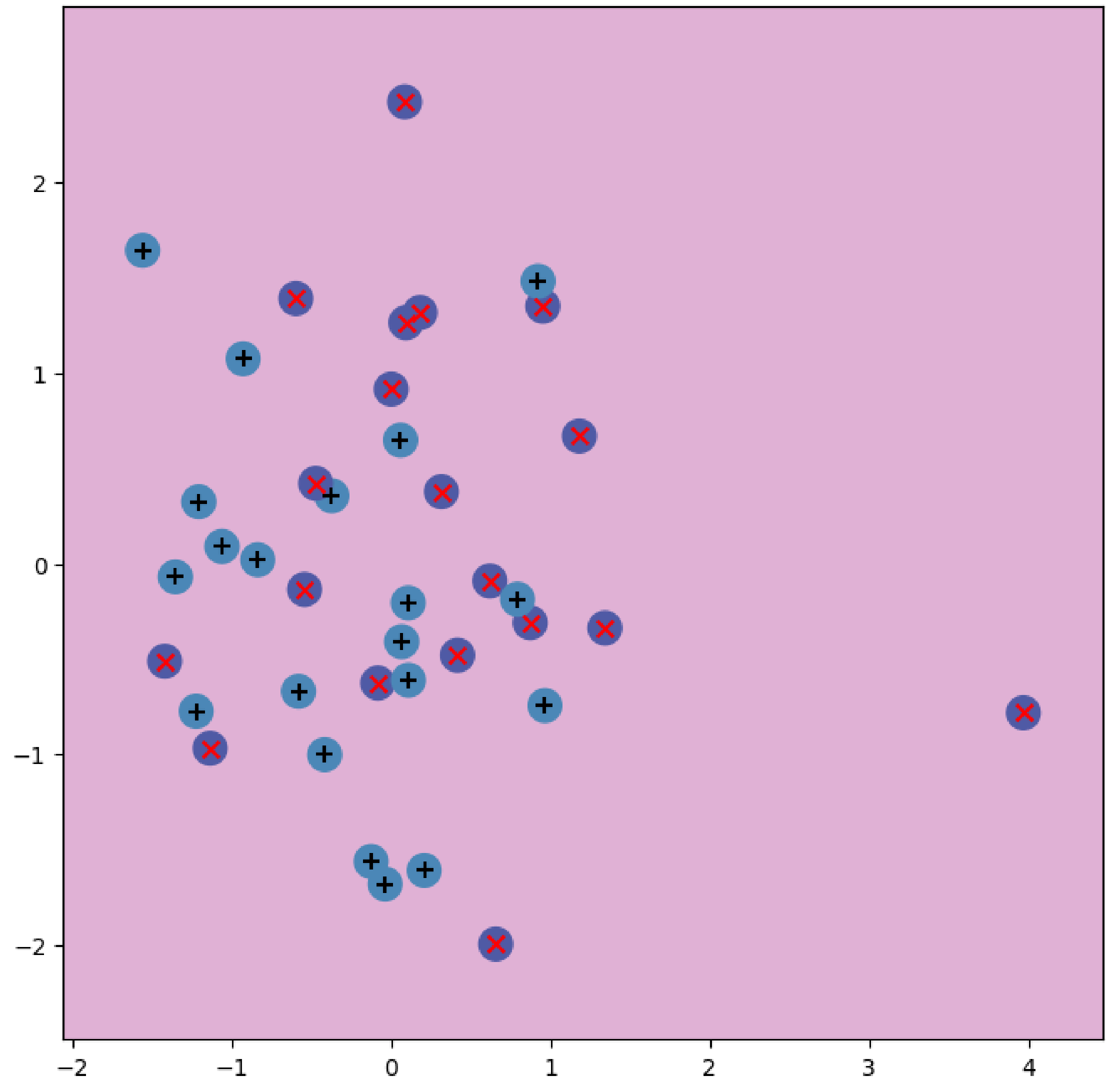
[61]:
```
grid.best_params_
```

[61]: {'C': 0.1, 'gamma': 0.5}

[62]:
```
best_rbf_svm = grid.best_estimator_
```

[63]:
```
# best_rbf_svm = SVC(kernel='rbf', C=1.0, gamma='scale')
# best_rbf_svm.fit(scaled_X_train, y_train)
```

[64]:
```
y_train_reset = y_train.reset_index(drop=True)
fig, ax = plt.subplots(figsize=(8, 8))
plot_svm(scaled_X_train, y_train_reset, best_rbf_svm, ax=ax)
```



[65]:
```
preds = best_rbf_svm.predict(scaled_X_test)

print(classification_report(y_true=y_test, y_pred=preds))
```

```
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00        23
           1       0.42      1.00      0.60        17

    accuracy                           0.42        40
   macro avg       0.21      0.50      0.30        40
weighted avg       0.18      0.42      0.25        40
```

[66]:
```
confusion_matrix(y_true=y_test, y_pred=preds)
```

[66]:
```
array([[ 0, 23],
       [ 0, 17]])
```

[67]:
```
(preds == y_test).mean()
```

[67]:
```
np.float64(0.425)
```

5 Poly SVM

[68]:
```
svm_poly = SVC(kernel="poly", gamma=1, C=1)
 #svm_poly.fit(scaled_X_train, y_train)
```

[69]:
```
C_range = [1, 10, 100, 1000]
degree_range = [2, 3, 4, 10]
gamma_range = [0.5, 1, 2, 3, 4]

params = {"C": C_range, "degree": degree_range, "gamma": gamma_range}

kfold = KFold(5, random_state=0, shuffle=True)

grid = GridSearchCV(svm_poly, param_grid=params, refit=True, cv=kfold,␣
 ↪scoring="accuracy")
```

[70]:
```
grid.fit(scaled_X_train, y_train)
```

[70]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=0, shuffle=True),
             estimator=SVC(C=1, gamma=1, kernel='poly'),
             param_grid={'C': [1, 10, 100, 1000], 'degree': [2, 3, 4, 10],
                         'gamma': [0.5, 1, 2, 3, 4]},
             scoring='accuracy')
```

[71]:
```
best_poly_svm = grid.best_estimator_
```

[72]:
```
best_poly_svm = SVC(kernel='poly', gamma=0.001, degree=8, C=0.1,␣
 ↪class_weight='balanced')
best_poly_svm.fit(scaled_X_train, y_train)
```

[72]:
```
SVC(C=0.1, class_weight='balanced', degree=8, gamma=0.001, kernel='poly')
```

[73]:
```
# y_train_reset = y_train.reset_index(drop=True)
# fig, ax = plt.subplots(figsize=(8, 8))
# plot_svm(scaled_X_train, y_train_reset, best_poly_svm, ax=ax)
```