Machine Learning HW6 - support vector machine

library

scikit learn

PCA

SVM (C-SVC)

GridSearchCV

pyplot

code

https://github.com/lobZter/HW_in_NCTU/blob/master/Machine_Learning/HW6/SVM.ipynb

What I have done?

First, I try using sklearn.svm.SVC() with no argument.

panalty parameter C: 1.0

kernel: rbf

gamma: 1/n_feature And have a not bad score 95.32% accuracy!

By default, parameters are:

```
In [4]: model = svm.SVC()
        model.fit(X_train_np, T_train_np)

Out[4]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
          decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
          max_iter=-1, probability=False, random_state=None, shrinking=True,
          tol=0.001, verbose=False)

In [7]: model.score(X_test, T_test)

Out[7]: 0.95320000000000005
```

Use grid search sklearn.model_selection.GridSearchCV() to find best parameters

Visualize the result by using PCA to project data onto 2D space

Take me so much time waiting…

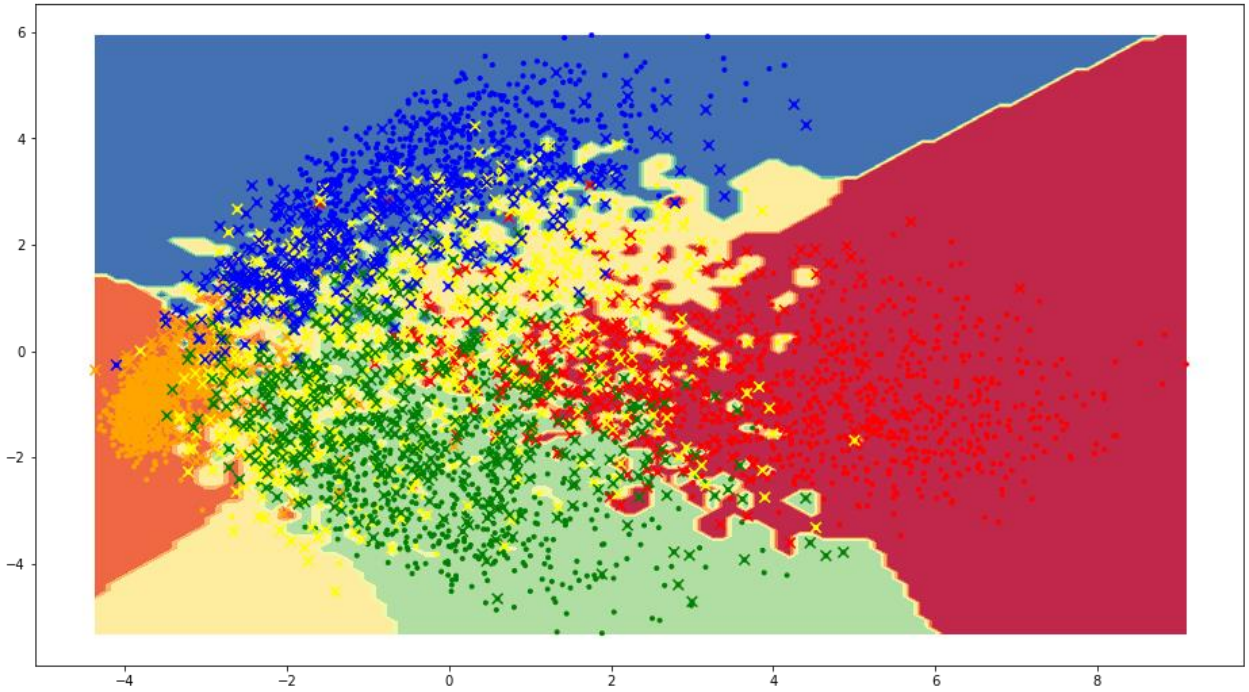
```
In [*]: model_cv = GridSearchCV(svm.SVC(), param_grid, cv=5)
        model_cv.fit(X_train_np, T_train_np)

In [24]: model_cv.best_params_

Out[24]: {'C': 2.0, 'gamma': 0.0004, 'kernel': 'rbf'}
```

pca_2d = PCA(n_components=2, copy=True) X_train_2d = pca_2d.fit_transform(X_train_np) x_2d, y_2d = zip(*X_train_2d)

Scatter data points (use cross marker to identify support vectors)
plt.scatter(x_2d, y_2d, color=colors, marker='.') plt.scatter(x_SVs_2d, y_SVs_2d, color=color_SVs, marker='x', s=size)
Use Voronoi diagram generated from 1-nearset neighbor as visualization of decision boundary
resolution = 100 # 100x100 background pixels X2d_xmin, X2d_xmax = np.min(X_train_2d[:,0]), np.max(X_train_2d[:,0]) X2d_ymin, X2d_ymax = np.min(X_train_2d[:,1]), np.max(X_train_2d[:,1]) xx, yy = np.meshgrid(np.linspace(X2d_xmin, X2d_xmax, resolution), np.linspace(X2d_ymin, X2d_ymax, resolution)) # approximate Voronoi tesselation on resolution x resolution grid using 1-NN background_model = KNeighborsClassifier(n_neighbors=1).fit(X_train_2d, T_train_predict)
voronoiBackground = background_model.predict(np.c_[xx.ravel(), yy.ravel()])
voronoiBackground = voronoiBackground.reshape((resolution, resolution))
Visualization result



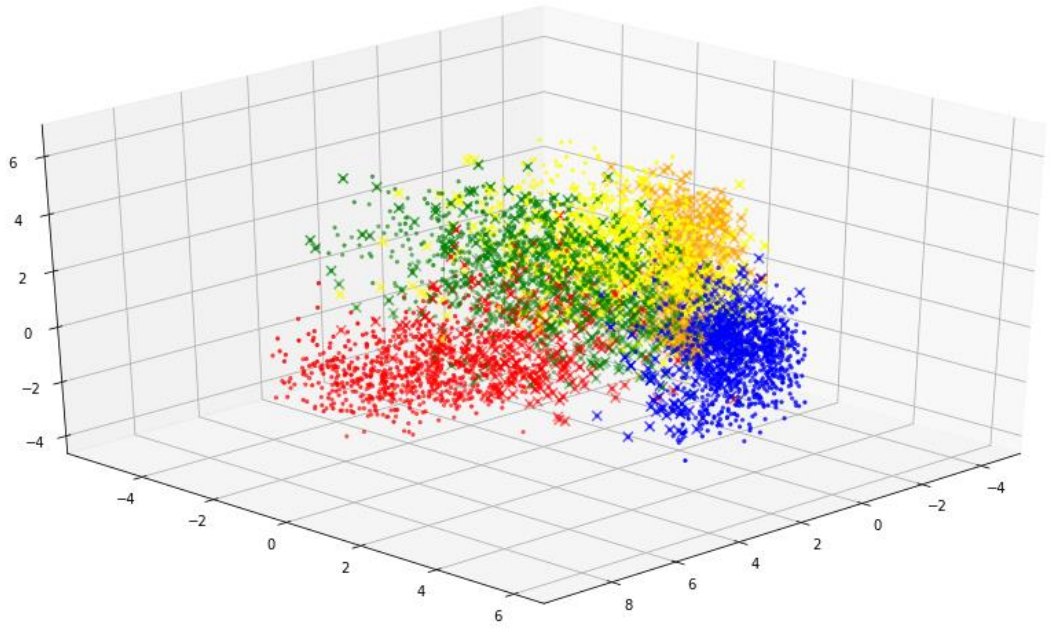
What I have learned?

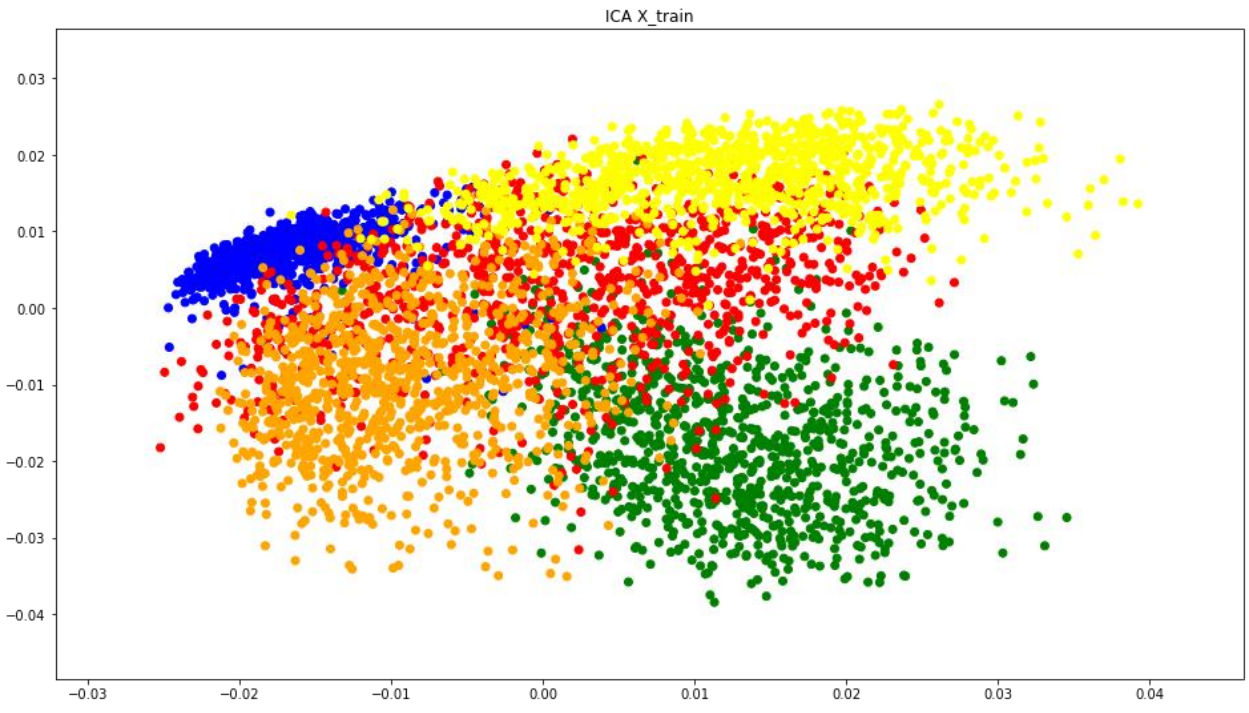
There is library tools that can do grid search, I no longer need to brute-forcely write loop to try all parameters.
Effect in different hyper-parameter such as panalty C and gamma used for rbf kernel
First time use 3D plot!

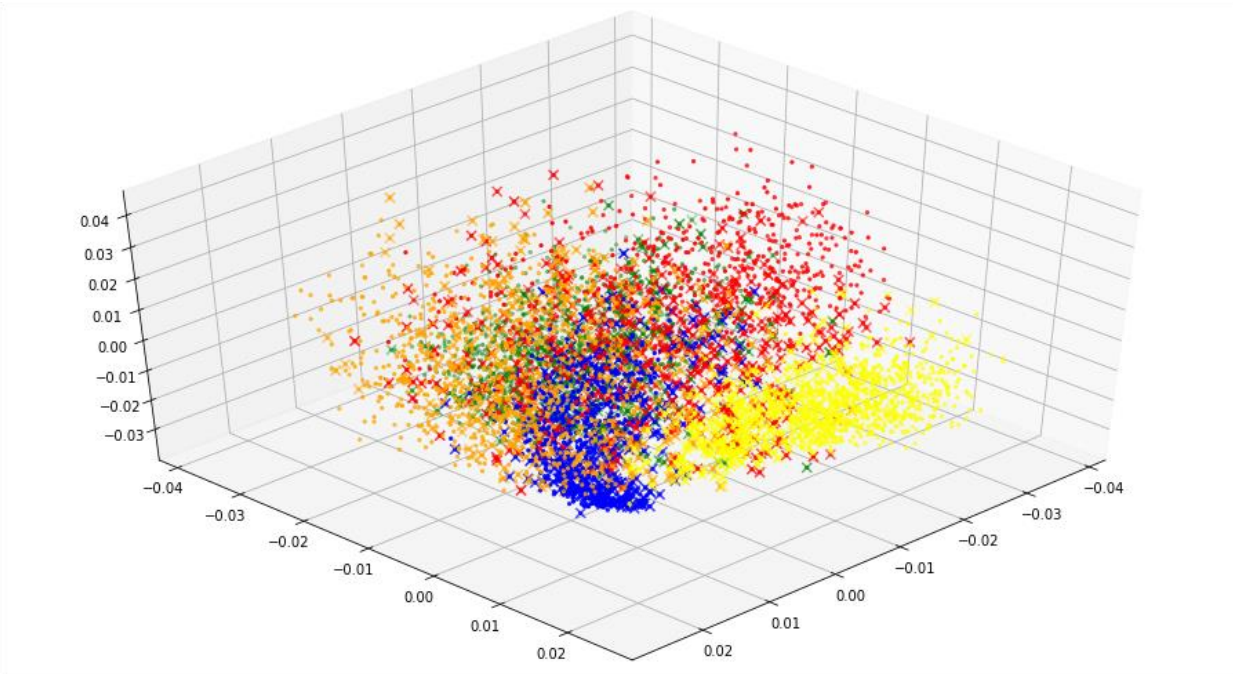
Something else

Projecting to 3D space is easier to observe support vectors

ICA compare to PCA






0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.03
−0.02
−0.01
0.00
0.01
0.02
−0.04
−0.03
−0.02
−0.01
0.00
0.01
0.02