Knowledge Discovery and Data Mining

Lab 7 SVM

Xuan Song
Songx@sustech.edu.cn

Topics

(1) Implement Linear-SVM with scikit-learn
(2) Implement RBF SVM with scikit-learn

Implementing Linear-SVM with Scikit-Learn

- sklearn.svm.SVC or

*class sklearn.svm.**SVC**(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None)*

- sklearn.svm.LinearSVC

*class sklearn.svm.LinearSVC(penalty='l2', loss='squared_hinge', *, dual=True, tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000)*

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Implementing Linear-SVM with Scikit-Learn

- Sample data: bank note authentication data set
  - Attribute Information:
    - 1. variance of Wavelet Transformed image (continuous)
      2. skewness of Wavelet Transformed image (continuous)
      3. curtosis of Wavelet Transformed image (continuous)
      4. entropy of image (continuous)
  - Class:
    - 0: unauthentic
      1: authentic

Attribute | Class

Variance	Skewness	Curtosis	Entropy	Class
3.6216	8.6661	-2.8073	-0.44699	0
4.5459	8.1674	-2.4586	-1.4621	0
3.866	-2.6383	1.9242	0.10645	0
3.4566	9.5228	-4.0112	-3.5944	0
0.32924	-4.4552	4.5718	-0.9888	0
4.3684	9.6718	-3.9606	-3.1625	0
3.5912	3.0129	0.72888	0.56421	0
2.0922	-6.81	8.4636	-0.60216	0
3.2032	5.7588	-0.75345	-0.61251	0
1.5356	9.1772	-2.2718	-0.73535	0
1.2247	8.7779	-2.2135	-0.80647	0
3.9899	-2.7066	2.3946	0.86291	0

Our task is to predict whether a bank currency note is authentic or not based upon four attributes of the note.

Implementing Linear-SVM with Scikit-Learn

1. Load data from csv files.
2. Data cleaning.
3. Get X and Y, and standardize X.

```
from sklearn.preprocessing import StandardScaler
X = StandardScaler().fit_transform(X)
```

4. Build a SVM model with scikit-learn.

```
from sklearn.svm import SVC
clf = SVC(kernel='linear')
```

5. Fit the model.

```
clf.fit(X_train, y_train)
```

https://www.kdnuggets.com/2020/04/data-transformation-standardization-normalization.html

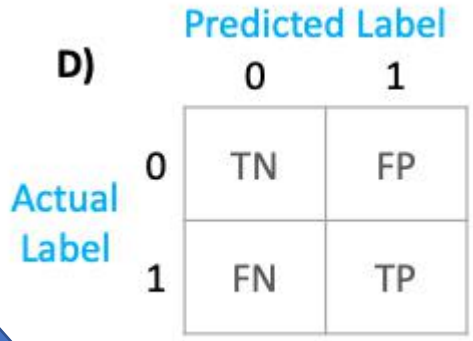
Implementing Linear-SVM with Scikit-Learn

6. Prediction.

```
y_pred = clf.predict(X_test)
```

7. Evaluation the model.

```
## accuracy##
clf.score(X_test, y_test)

## confusion_matrix (混淆矩阵) & F1 score  ##
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))

from sklearn import metrics
precision = metrics.precision_score(y_test, y_pred)
recall = metrics.recall_score(y_test, y_pred)
f1 = metrics.f1_score(y_test, y_pred)
```

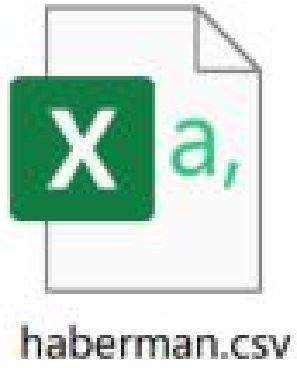



```
[[151  2]
[ 1  121]]
```

https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

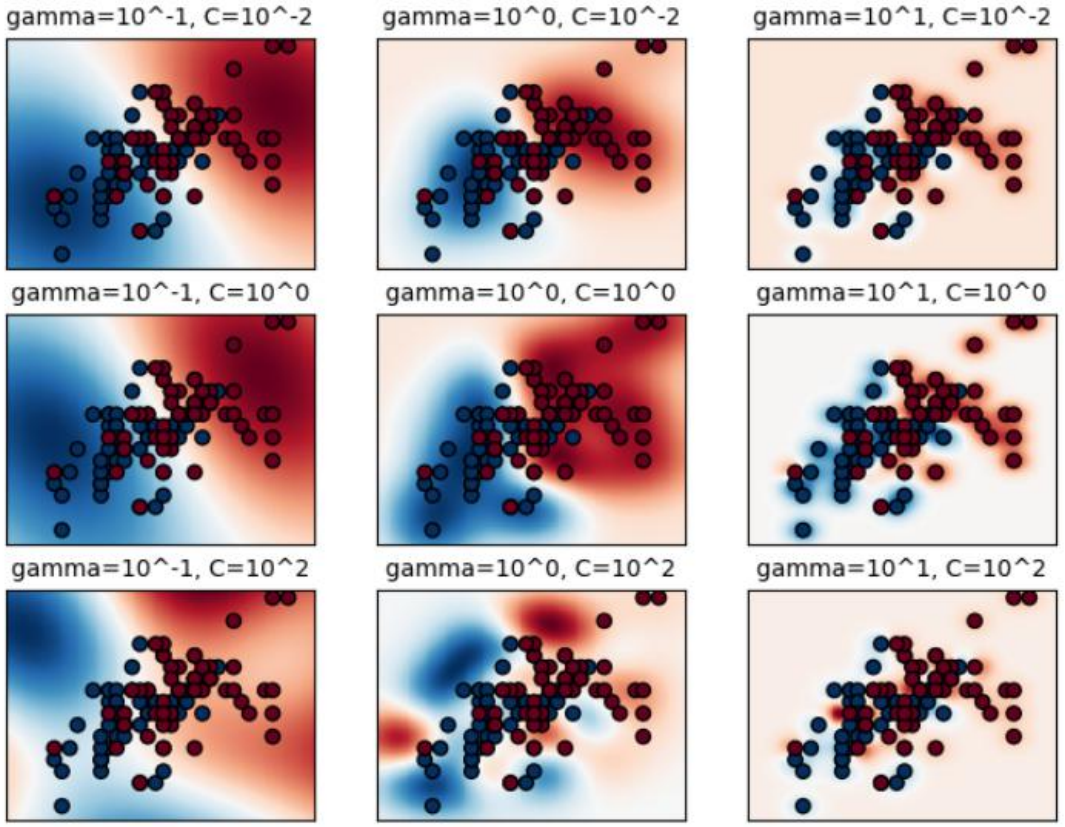
Task1:

- Implementing Linear-SVM based on the given dataset.
  - Haberman’s Survival Data Set
    - Description:
      - Attribute:
        - 1. Age of patient at time of operation (numerical)
          2. Patient's year of operation (year - 1900, numerical)
          3. Number of positive axillary nodes detected (numerical)
      - Class: Survival status
        - -- 1 = the patient survived 5 years or longer
          -- 2 = the patient died within 5 year



Implementing RBF SVM with Scikit-Learn

*class sklearn.svm.**SVC**(*, **C=1.0**, kernel='rbf', degree=3, **gamma='scale'**, coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None)*



https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Implementing RBF SVM with Scikit-Learn

- sklearn.model_selection.GridSearchCV

*class sklearn.model_selection.GridSearchCV(**estimator**, **param_grid**, *, scoring=None, n_jobs=None, refit=True, **cv**=None, verbose=0, pre_dispatch='2*n_jobs', error_score=nan, return_train_score=False)*

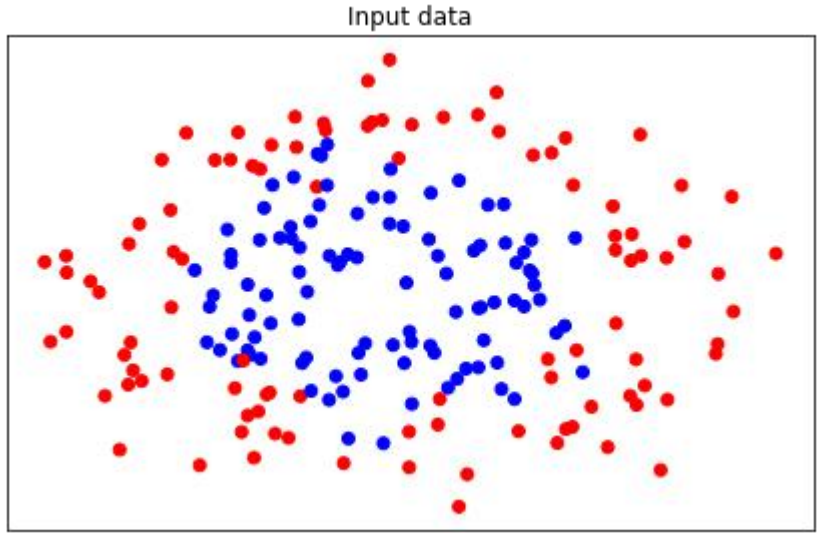
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
https://machinelearningmastery.com/k-fold-cross-validation/

Implementing RBF SVM with Scikit-Learn

- Dataset:

Attribute Class

	x1	x2	class
0	-1.56366	-0.9548	0
1	-1.20983	1.595268	0
2	-0.89427	1.601064	0
3	0.584755	-0.17071	1
4	0.162367	-0.52175	1
5	0.943815	-0.09296	1
6	1.408254	-0.33384	0
7	2.382925	0.373576	0
8	-0.547	1.758576	0
9	0.440602	-0.89595	1
10	-0.83652	-0.60722	1
11	0.695012	1.609827	0
12	1.560784	0.350477	0



Implementing RBF SVM with Scikit-Learn

1. Load data from csv files.
2. Data cleaning.
3. Get X and Y, and standardize X.
4. Find the best parameters for RBF SVM model.

```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(SVC(kernel='rbf'), param_grid={"C":[0.1, 1, 10], "gamma": [1, 0.1, 0.01]}, cv=5)
grid.fit(X, y)
print("The best parameters are %s with a score of %0.2f"
      % (grid.best_params_, grid.best_score_))
```

```
The best parameters are {'C': 1, 'gamma': 0.1} with a score of 0.90
```

Implementing RBF SVM with Scikit-Learn

5. Build a RBF SVM model with best parameters.

```
clf = SVC(kernel='rbf', C=grid.best_params_['C'], gamma=grid.best_params_['gamma'])
```

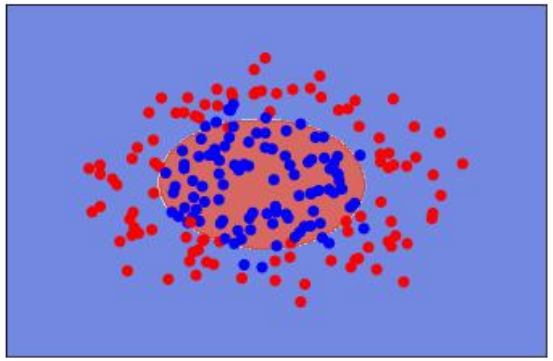
6. Fit the model.

```
clf.fit(X_train, y_train)
```

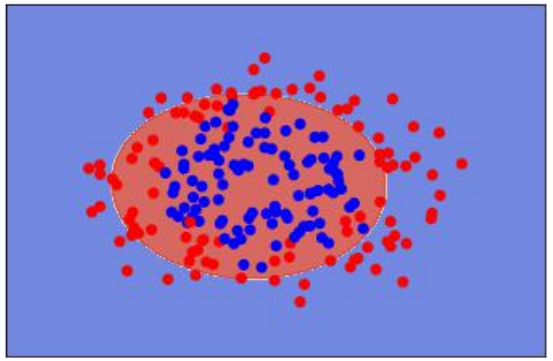
7. Prediction and evaluation.

```
y_pred = clf.predict(X_test)

#evaluation
from sklearn import metrics
accuracy = metrics.accuracy_score(y_test, y_pred)

precision = metrics.precision_score(y_test, y_pred)
recall = metrics.recall_score(y_test, y_pred)
f1 = metrics.f1_score(y_test, y_pred)
```

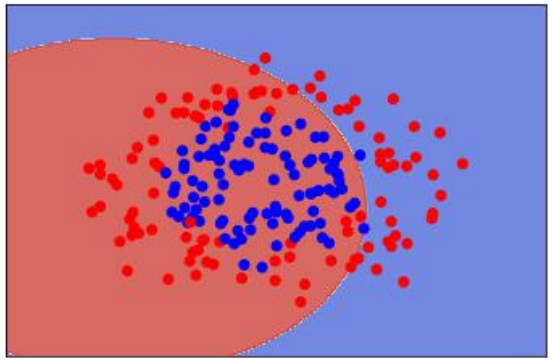
Hyperplanes with different hyperparameters



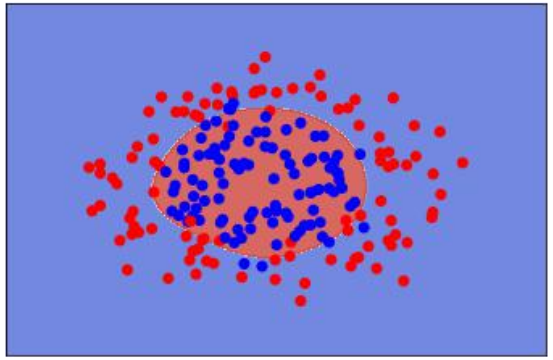
gamma=1 C=0.1



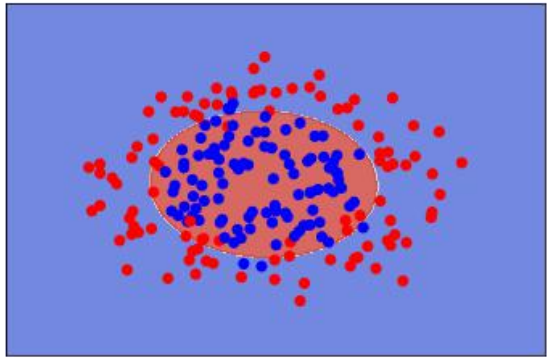
gamma=0.1 C=0.1



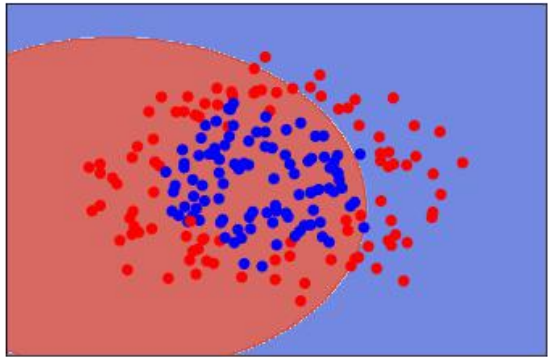
gamma=0.01 C=0.1



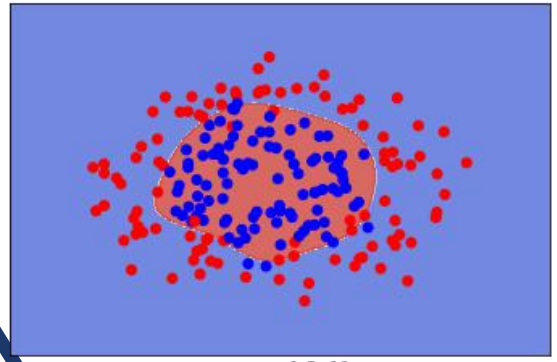
gamma=1 C=1



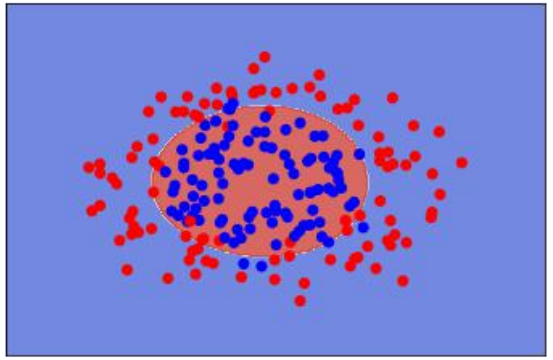
gamma=0.1 C=1



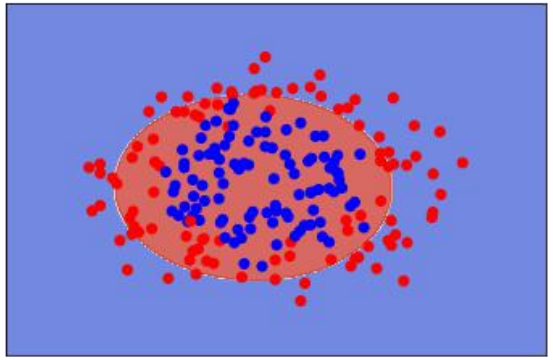
gamma=0.01 C=1



gamma=1 C=10



gamma=0.1 C=10



gamma=0.01 C=10

Task2:

- Implementing RBF SVM based on the given dataset.

Tasks

- Implementing Linear SVM with Scikit-Learn.
- Implementing RBF SVM with Scikit-Learn.

End of Lab7