Elective Course
Course Code: CS4103
Autumn 2025-26

Lecture #45

Artificial Intelligence for Data Science

Week-13:
MACHINE LEARNING (Part XIII)
Exploring Support Vector Machines (SVMs) using Python (Scikit-learn)

Course Instructor:
Dr. Monidipa Das
Assistant Professor
Department of Computational and Data Sciences
Indian Institute of Science Education and Research Kolkata, India 741246

Support Vector Classifier from 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) #
```

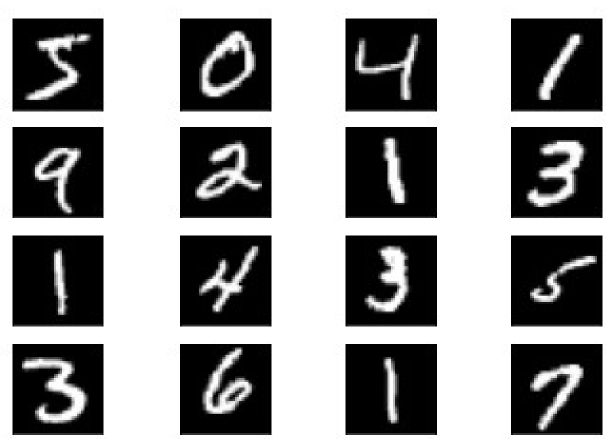
```
from sklearn import svm
# or
from sklearn.svm import SVC
```

SVM for Handwritten-digit Classification: Dataset Loading

```
import matplotlib.pyplot as plt
import pickle
import numpy as np

with open("F:/CS4103/code/mnist.pkl","rb") as fh:
    train_set, validation_set, test_set = pickle.load(fh,encoding='latin1')

train_imgs,train_labels = train_set[0], train_set[1]
valid_imgs, valid_labels = validation_set[0], validation_set[1]
test_imgs, test_labels = test_set[0], test_set[1]


image_size = 28
no_of_different_labels = 10
image_pixels = image_size * image_size
```


SVM for Handwritten-digit Classification: Data Visualization

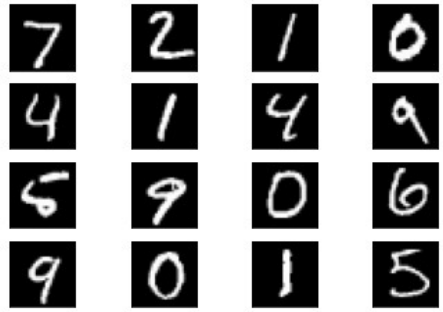
```
fig, axes = plt.subplots(4, 4)

# Flatten the axes array for easy
iteration
axes = axes.flatten()

#Plot a few training sample images
for i, ax in enumerate(axes):
    ax.imshow(train_imgs[i].reshape(
         image_size, image_size),
         cmap=plt.cm.gray)
    ax.set_xticks(())
    ax.set_yticks(())
```



SVM for Handwritten-digit Classification: Building Classifier

5

```python
from sklearn import svm

param_C = 5
param_gamma = 0.05

classifier = svm.SVC(C=param_C,gamma=param_gamma,ver
#Train the classifier
classifier.fit(train_imgs, train_labels)
fig, axes = plt.subplots(4, 4)
# Flatten the axes array for easy iteration
axes = axes.flatten()
#Plotting a few test samples
for i, ax in enumerate(axes):
    ax.imshow(test_imgs[i].reshape(image_size, image_size), cmap=plt.cm.gray)
    ax.set_xticks(())
    ax.set_yticks(())

#Predict for the test images
predicted = classifier.predict(test_imgs)
```



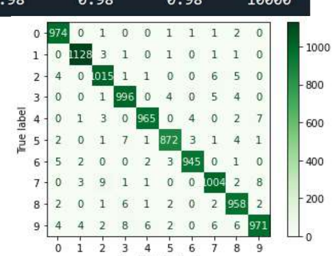
SVM for Handwritten-digit Classification: Evaluation

6

```python
from sklearn.metrics import
confusion_matrix,ConfusionMatrixDisplay,
classification_report,accuracy_score

print("Classification report\n",
classification_report(test_labels, predicted))

cm_pred = confusion_matrix(test_labels, predicted)

plt.figure()
disp =
ConfusionMatrixDisplay(confusion_matrix=cm_pred,
display_labels=np.unique(test_labels))
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()

print("Accuracy={}".format
(accuracy_score(test_labels, predicted)))
```

Classification report

	precision	recall	f1-score	support
0	0.98	0.99	0.99	980
1	0.99	0.99	0.99	1135
2	0.98	0.98	0.98	1032
3	0.98	0.99	0.98	1010
4	0.99	0.98	0.99	982
5	0.99	0.98	0.98	892
6	0.99	0.99	0.99	958
7	0.98	0.98	0.98	1028
8	0.97	0.98	0.98	974
9	0.98	0.96	0.97	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000



Accuracy=0.9828

SVM for Iris Classification: Dataset Loading

```python
import pandas as pd

#Load the dataset
dataset =
pd.read_csv('F:/CS4103/code/iris.csv')

#View dataset
print("First 6 rows of the dataset:")
print(dataset.head(6))

#Dataset column names
print("\nName of the columns in the dataset:")
print(list(dataset.columns))

input_features = list(dataset.columns[:-1])
class_col=dataset.columns[-1]
print("Name of the input feature columns:",input_features)
print("Name of the class/label column:",class_col)
```

```
First 6 rows of the dataset:
   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
5           5.4          3.9           1.7          0.4  Setosa
```

```
Name of the columns in the dataset:
['sepal.length', 'sepal.width',
'petal.length', 'petal.width', 'variety']
Name of the input feature columns:
['sepal.length', 'sepal.width',
'petal.length', 'petal.width']
Name of the class/label column: variety
```


SVM for Iris Classification: Exploring Dataset Details

```python
# Number of rows or instances and number of columns features
print("\nTotal number of data points/examples/instances:", dataset.shape[0])
print("Total number of input features:", dataset.shape[1]-1)

#Dataset description
print("\nDescription of the dataset:")
print(dataset.describe())

#Print number of classes
c=dataset[class_col].nunique()
print("\nNumber of classes(discrete labels):",c)

#Number of instances per class
print("\nSample count per class:")
print(dataset[class_col].value_counts())

#Check missing values in variables
dataset.isnull().sum()
```

```
Total number of data points/examples/instances: 150
Total number of input features: 4
```

```
Description of the dataset:
       sepal.length  sepal.width  petal.length  petal.width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

```
Number of classes (discrete labels): 3

Sample count per class:
Setosa        50
Versicolor    50
Virginica     50
Name: variety, dtype: int64
```

```
sepal.length    0
sepal.width     0
petal.length    0
petal.width     0
variety         0
dtype: int64
```

SVM for Iris Classification: Data Split

9

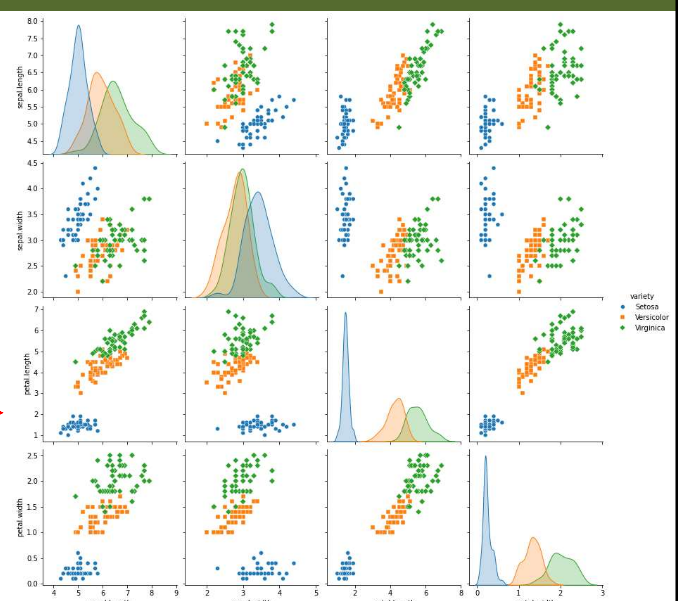
```
X = dataset[input_features].values
y = dataset[class_col].values

#split dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

#check the shape of X_train and X_test
print("Traning set shape:",X_train.shape,"\nTest set shape:",X_test.shape)
```

```
Traning set shape: (120, 4)
Test set shape: (30, 4)
```


SVM for Iris Classification: Data Visualization

10

```
#visualize the distribution
of features and their
relationship with others

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure()
sns.pairplot(dataset, hue =
class_col, size=3, markers=["o",
"s", "D"])
plt.show()
```



SVM for Iris Classification: Building Classifier and Evaluation

11

```python
#import svm module
from sklearn import svm

param_C = 5
param_gamma = 0.05
classifier = svm.SVC(C=param_C,
gamma=param_gamma,verbose=True)
classifier.fit(X_train, y_train)

#Predict the value of the test
predicted = classifier.predict(X_test)

from sklearn.metrics import
confusion_matrix,ConfusionMatrixDisplay,
classification_report,accuracy_score
print("Classification report for classifier %s:\n%s\n"
      % (classifier, classification_report(y_test, predicted)))
print("Accuracy={}".format(accuracy_score(y_test, predicted)))
```

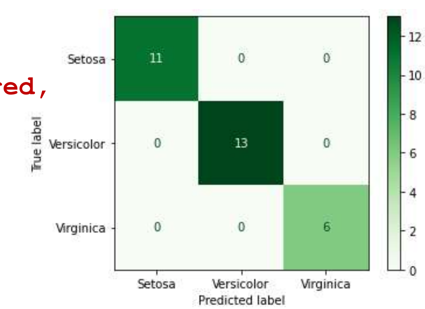
```
[LibSVM]Classification report for classifier SVC(C=5,
gamma=0.05, verbose=True):
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        11
  Versicolor       1.00      1.00      1.00        13
   Virginica       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

```
Accuracy=1.0
```

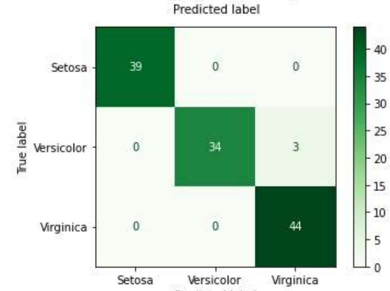

SVM for Iris Classification: Evaluation

12

```python
cm_pred = confusion_matrix(y_test, predicted)
plt.figure()
disp = ConfusionMatrixDisplay(confusion_matrix=cm_pred,
display_labels=np.unique(y_test))
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()
```




```python
predicted = classifier.predict(X_train)
cm_pred = confusion_matrix(y_train, predicted)
plt.figure()
disp = ConfusionMatrixDisplay(
confusion_matrix=cm_pred,
display_labels=np.unique(y_test))
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()
```



SVM for Iris Classification: Decision Boundary

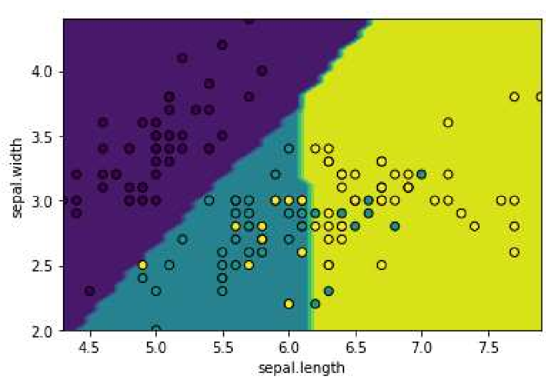
13

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.inspection import DecisionBoundaryDisplay
feature_1, feature_2 = np.meshgrid(
    np.linspace(X[:, 0].min(), X[:, 0].max()),
    np.linspace(X[:, 1].min(), X[:, 1].max())
)

grid = np.vstack([feature_1.ravel(), feature_2.ravel()]).T

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train1 = le.fit_transform(y_train)
y1 = le.fit_transform(y)

classifier = svm.SVC(C=param_C,gamma=param_gamma,verbose=True)
classifier.fit(X_train[:,[0,1]], y_train1)
```


SVM for Iris Classification: Decision Boundary

14

```
y_pred = np.reshape(classifier.predict(grid), feature_1.shape)
display = DecisionBoundaryDisplay(
    xx0=feature_1, xx1=feature_2, response=y_pred)
display.plot()

display.ax_.scatter(
    X[:, 0], X[:, 1], c=y1, edgecolor="black"
)
plt.xlabel(input_features[0])
plt.ylabel(input_features[1])
plt.show()
```



SVM for Iris Classification: Decision Boundary

15

```
from sklearn.inspection import DecisionBoundaryDisplay
def plot_training_data_with_decision_boundary(
    kernel, ax=None, long_title=True, support_vectors=True):
    # Train the SVC
    clf = svm.SVC(kernel=kernel, gamma=2).fit(X[:,0:2], y1)

    # Settings for plotting
    if ax is None:
        _, ax = plt.subplots(figsize=(4, 3))
    x_min, x_max, y_min, y_max = min(X[:,0])-1,max(X[:,0])+1,min(X[:,1])-
1,max(X[:,1])+1
    ax.set(xlim=(x_min, x_max), ylim=(y_min, y_max))

    # Plot decision boundary and margins
    common_params = {"estimator": clf, "X": X[:,0:2], "ax": ax}
    DecisionBoundaryDisplay.from_estimator(**common_params,
response_method="auto",plot_method="pcolormesh",
        alpha=0.3)
```

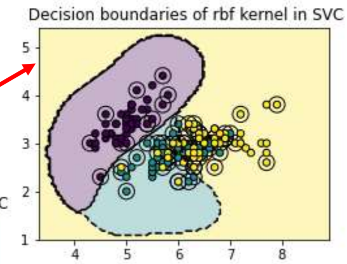

SVM for Iris Classification: Decision Boundary

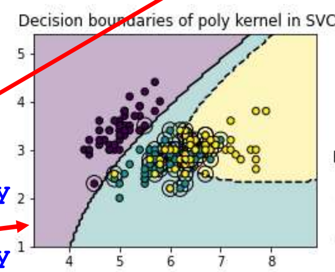
16

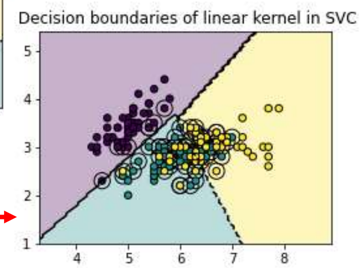
```
 DecisionBoundaryDisplay.from_estimator(
        **common_params,
        response_method="auto",
        plot_method="contour",
        levels=[-1, 0, 1],
        colors=["k", "k", "k"],
        linestyles=["--", "-", "--"])

    if support_vectors:
        # Plot bigger circles around samples that serve as support vectors
        ax.scatter(
            clf.support_vectors_[:, 0],
            clf.support_vectors_[:, 1],
            s=150,
            facecolors="none", edgecolors="k")
```

SVM for Iris Classification: Decision Boundary


```
# Plot samples by color and add legend
    ax.scatter(X[:, 0], X[:, 1], c=y1,
s=30, edgecolors="k")
    if long_title:
        ax.set_title(f" Decision
boundaries of {kernel} kernel in SVC")
    else:
        ax.set_title(kernel)
    if ax is None:
        plt.show()

plot_training_data_with_decision_boundary
('rbf')
plot_training_data_with_decision_boundary
('poly')
plot_training_data_with_decision_boundary
('linear')
```







Questions?