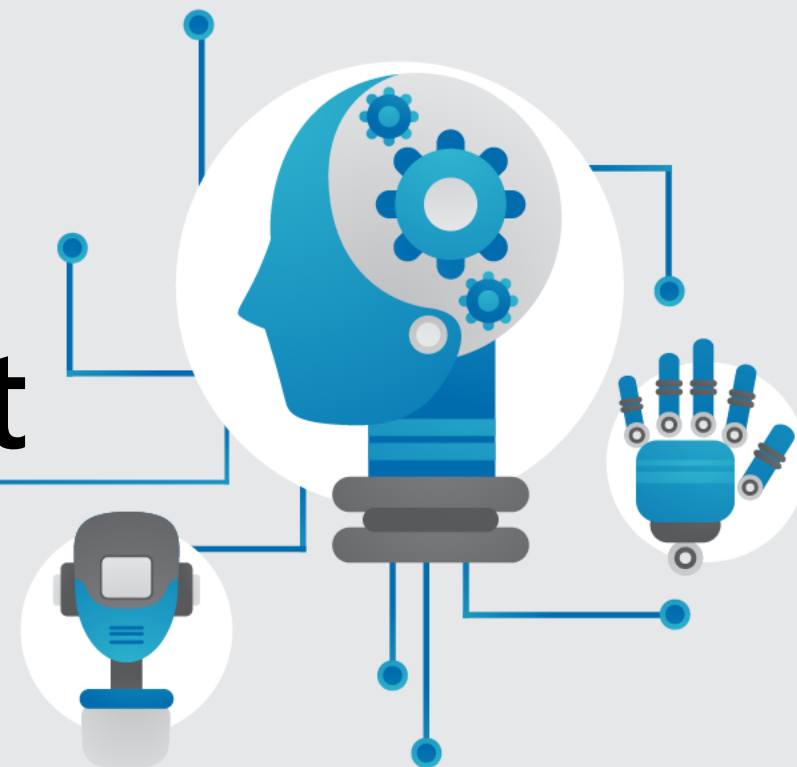


Scikit-learn Support Vector Classifier






Support Vector Classifier介紹

機器學習實務



- › `sklearn.svm.SVC`是支援向量機 (support vector machine)
SVM演算法用於分類的實作

 [Install](#) [User Guide](#) [API](#) [Examples](#) [More](#) Go

[Prev](#) [Up](#) [Next](#)

scikit-learn 0.22.2
[Other versions](#)

Please [cite us](#) if you use the software.

[sklearn.svm.SVC](#)
Examples using [sklearn.svm.SVC](#)

sklearn.svm.SVC

```
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)
```

[\[source\]](#)

C-Support Vector Classification.

The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using `sklearn.svm.LinearSVC` or `sklearn.linear_model.SGDClassifier` instead, possibly after a `sklearn.kernel_approximation.Nystroem` transformer.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how `gamma`, `coef0` and `degree` affect each other, see the corresponding section in the narrative documentation: [Kernel functions](#).

[Read more in the User Guide.](#)



Support Vector Classifier 參數說明

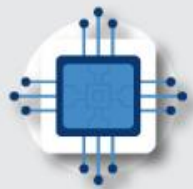
機器學習實務



› `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)`

› 常用重要參數

- C
- kernel
- degree 、 gamma 、 coef0
- class_weight
- decision_function_shape

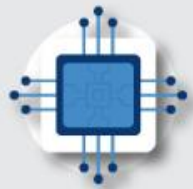


參數 C



› C : float, optional (default=1.0)

- ✓ C是正規化(Regularization)參數，正規化的強度與C成反比。
- ✓ C必須是正值
- ✓ C愈大，即對分錯樣本的懲罰程度愈大，因此在訓練樣本中準確率愈高，但是泛化能力降低，也就是對測試數據的分類準確率降低。模型容易過度學習。
- ✓ 懲罰 (penalty) 是平方L2 (squared L2) 。



參數 kernel



› **kernel** : string, optional (default= 'rbf')

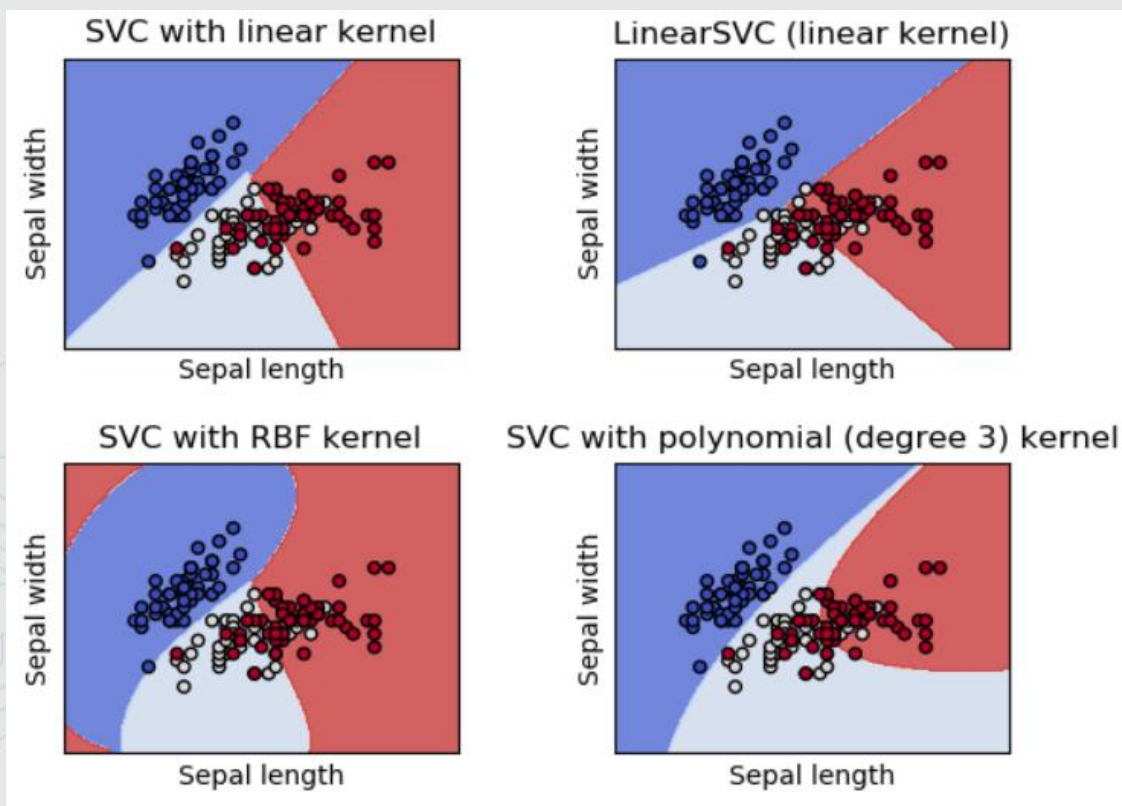
- ✓ 指定演算法要使用的核函數(kernel)類型，選項有'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'。
- ✓ 'linear' : 線性核函數
- ✓ 'poly' : 多項式核函數
- ✓ 'rbf' : 徑像核函數/高斯核函數
- ✓ 'sigmoid' : sigmoid核函數
- ✓ 'precomputed' : 核矩陣 (自訂矩陣)



參數 kernel



› kernel : string, optional (default= 'rbf')





參數 degree, gamma, coef0



› degree : int, optional (default=3)

- ✓ 多項式poly函式的維度，預設是3，選擇其他核函式時會被忽略。

› gamma : {'scale', 'auto'} or float, optional (default= 'scale')

- ✓ 'rbf'、'poly'和'sigmoid'的內核函數係數。
- ✓ 如果gamma = 'scale' (預設) ，
則使用 $1 / (n_features * X.var)$ 作為gamma值。
- ✓ 如果為'auto'，則使用 $1 / n_features$ 。

› coef0 : float, optional (default=0.0)

- ✓ 核函式的常數項。只對於核心函式'poly'和'sigmoid'有用。

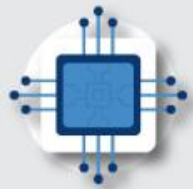


參數 `class_weight`



› `class_weight` : dict, list of dicts, 'balanced', 'balanced_subsample' or None, optional (default=None)

- ✓ 類別的權重，以字典格式表示{class_label : weight}
- ✓ 假設類別i的參數C，設定類別i的權重為 $\text{weight}[i] * C$
- ✓ 如果未設定，則所有類別的權重都設定為1
- ✓ 'balanced' 模式使用y的值來自動調整與輸入數據中的類別頻率成反比的權重，如下所示：
$$\text{n_samples} / (\text{n_classes} * \text{np.bincount}(y))$$



參數說明 decision_function_shape



› decision_function_shape : 'ovo', 'ovr', default= 'ovr'

- 最終決策函數是要選擇ovr，或者選擇ovo。
- **ovr(one-vs-rest) : 型態為(n_samples, n_classes)**
訓練時依次把某一類分為一類，剩下都分為另一類，
這樣n個類別就需要有n個SVM，預測分類時取預測值最大的那一類。
- **ovo(one-vs-one) : 型態為(n_samples, n_classes * (n_classes - 1) / 2)**
其做法是在任意兩類樣本之間設計一個SVM，
因此k個類別的樣本就需要設計k(k-1)/2個SVM。
當對一個未知樣本進行分類時，
最後得票最多的類別即為該未知樣本的類別。
Libsvm中的多類分類就是根據這個方法實現的。



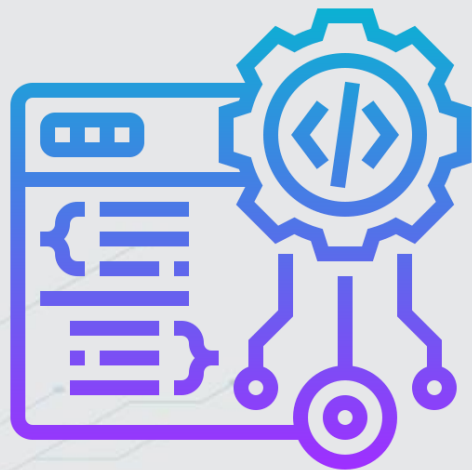
Support Vector Classifier 函式說明

機器學習實務



› Support Vector Classifier 常用函式

- fit
- predict
- score





訓練 (fit)



› 指令 `fit(self, x, y, sample_weight=None)`

› 參數

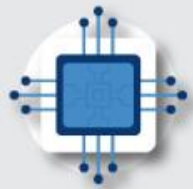
- x : 訓練輸入樣本
- y : 目標值 (分類中的類標籤)

› 回傳 : 訓練後的support vector classifier模型物件

› 說明 : 根據給定的訓練數據訓練SVM模型。

› 範例程式

```
from sklearn.svm import SVC  
  
svc = SVC()  
  
svc.fit(x_train, y_train)
```



預測 (predict)



› 指令 `predict(self, x)`

› 參數

- `x`: 輸入樣本

› 回傳：`x`中樣本的類別標籤

› 說明：對`x`中的樣本執行分類

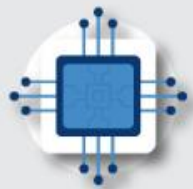
› 範例程式

```
from sklearn.svm import SVC
```

```
svc = SVC()
```

```
svc.fit(x_train, y_train)
```

```
predictions = svc.predict(x_test)
```



評分 (score)



› 指令 `score(self, x, y, sample_weight=None)`

› 參數

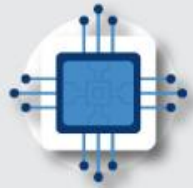
- x : 測試樣本
- y : 測試樣本的正确答案

› 回傳 : 測試樣本的平均準確度

› 說明 : 返回給定測試數據和標籤上的平均準確度

› 範例程式

```
from sklearn.svm import SVC  
svc = SVC()  
svc.fit(X_train, y_train)  
accuracy = svc.score(X_test, y_test)
```



程式範例 (IRIS)



› 程式碼

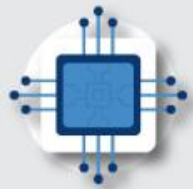
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn import datasets

# 載入資料
iris = datasets.load_iris()
X = iris.data[:, :2] # 只取前兩種特徵
Y = iris.target

# 建立 Support Vector Machine Classifier
svc = SVC()

# 進行訓練
svc.fit(X, Y)

# 繪製座標軸
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
h = .02 # 單位間隔
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
```

程式範例 (IRIS)



› 程式碼

進行預測

```
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
```

繪製預測結果

```
Z = Z.reshape(xx.shape)
```

```
plt.figure(1, figsize=(4, 3))
```

```
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)
```

```
plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=plt.cm.Paired)
```

```
plt.xlabel('Sepal length')
```

```
plt.ylabel('Sepal width')
```

```
plt.xlim(xx.min(), xx.max())
```

```
plt.ylim(yy.min(), yy.max())
```

```
plt.xticks()
```

```
plt.yticks()
```

```
plt.show()
```



程式範例 (IRIS)

機器學習實務



› 輸出結果

