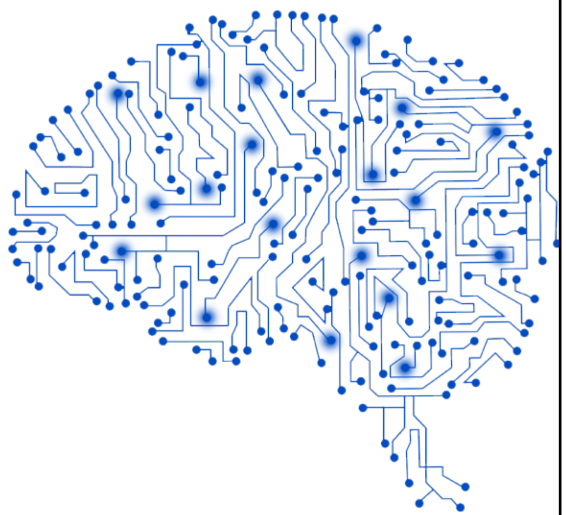
Python Programming
Machine Learning
Lecture 04

Min-Kuan Chang

GICE, EECS



線性分類模型

- 線性模型也常用在分類問題上面
- 同樣地，我們會採用下面的線性模型來試圖對資料做分類

$$\hat{y}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$$

  - 這通常叫做線性區分函示
- 當只有兩類資料需要做區分的時候，如果加設決策邊界在 $\hat{y}(\boldsymbol{x}) = 0$，則我們們會有下面的決策規則

$$\hat{y}(\boldsymbol{x}) \underset{C_2}{\overset{C_1}{\gtrless}} 0$$

- 有很多演算法都適用在現行模型上面。這些演算法主要區別如下：
  - 再用何種方式來評估模型與資料吻合的程度
  - 採用何種正規化的方法
- 最常見的兩種線性分模型演算法
  - logistic regression
    - linear_model.LogisticRegression
  - linear support vector machines (linear SVMs)
    - svm.LinearSVC (SVC stands for support vector classifier)

線性分類模型

Logistic Regression

線性分類模型

Logistic Regression

基礎

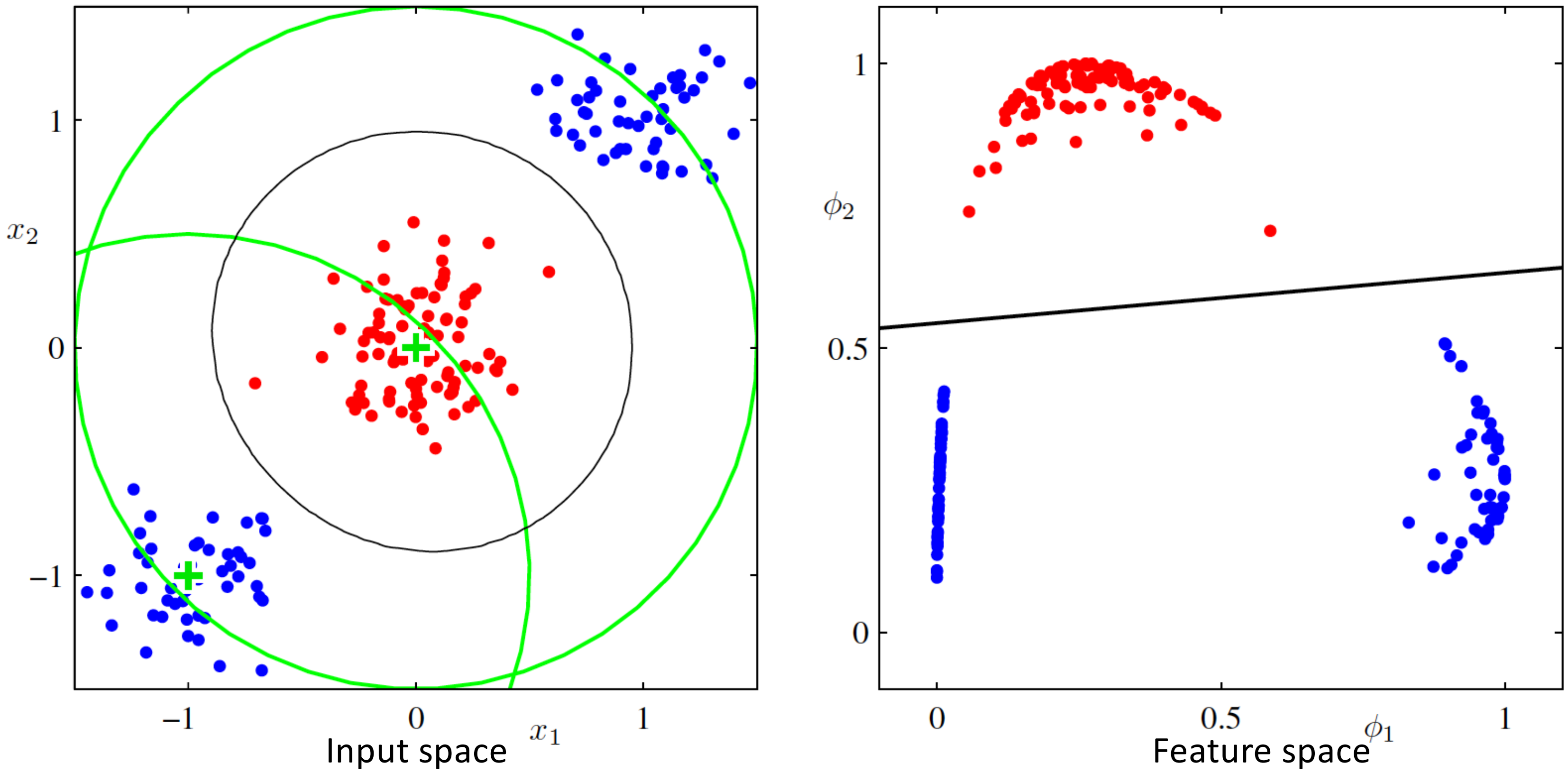
$$\boxed{p(C_k|\boldsymbol{X})} = \frac{\boxed{f_{X|C_k}(\boldsymbol{x})p(C_k)}}{f_X(\boldsymbol{x})}$$

Probabilistic generative model — **Two Fittings**

Probabilistic discriminant model — **One Fitting**

基底函式

Linear inseparability



Linear separability

Logistic regression

- 在 logistic regression 中，類別 C_1的事後機率為

$$p(C_1|\phi) = p(\phi) = \sigma(w^T\phi)$$

  - $\sigma(\cdot)$ 是 logistic sigmoid function

$$\sigma(a) = \frac{1}{1 + \exp\{-a\}}$$

- 對於M維度的特徵空間來說，這樣的模型則會有M可供調整的參數

線性分類模型

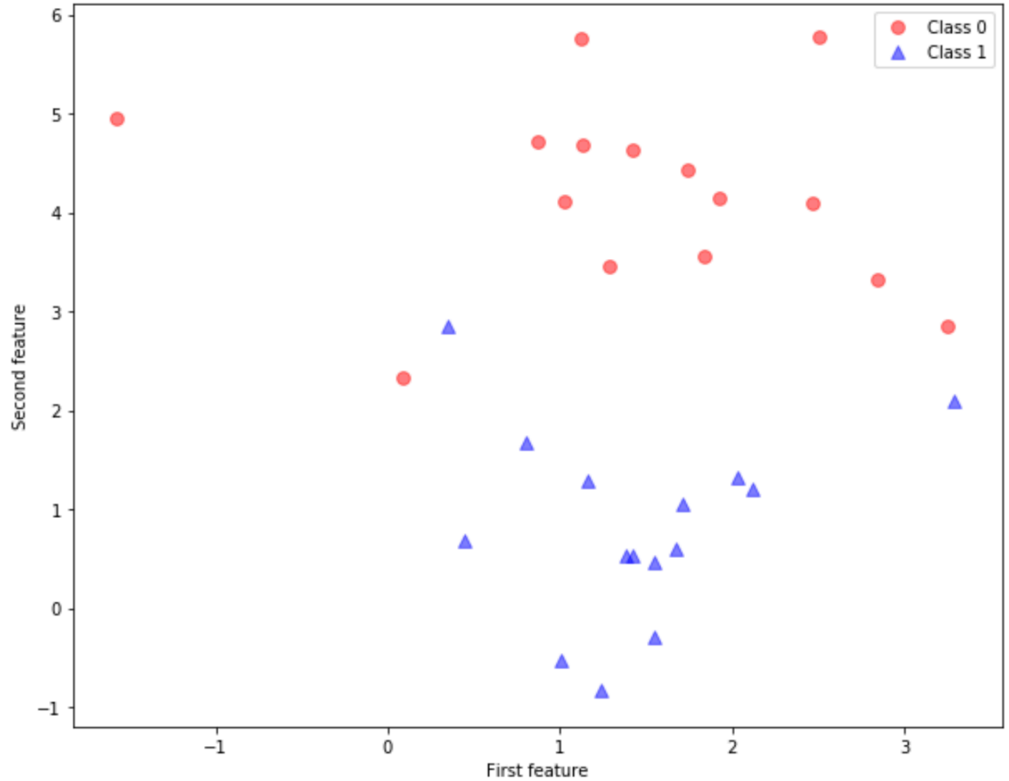
Logistic Regression

實現

步驟一: 資料的收集或產生

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

X, y = make_blobs(n_samples=30, centers=2, n_features=2,random_state=0)

plt.figure(figsize=(10,8))

plt.scatter(X[y==0, 0], X[y==0, 1], marker='o', c='red',s=60, alpha=0.5, label='Class 0')
plt.scatter(X[y==1, 0], X[y==1, 1], marker='^', c='blue',s=60, alpha=0.5, label='Class 1')

plt.xlabel('First feature')
plt.ylabel('Second feature')
plt.legend(loc='upper right')
plt.show()
```

步驟一: 資料的收集或產生



步驟二: 產生一個 LogisticRegression 實例

class sklearn.linear_model.LogisticRegression(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)*

- Logistic Regression (aka logit, MaxEnt) classifier
- **penalty**
  - *str, 'l1', 'l2', 'elasticnet' or 'none', optional (default='l2')*
  - 'newton-cg', 'sag' and 'lbfgs' solvers 只支援 L_2懲罰

步驟二: 產生一個 LogisticRegression 實例

class sklearn.linear_model.LogisticRegression(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)*

- **tol**
  - *float, optional (default=1e-4)*
  - 停止條件的容忍值
- **C**
  - *float, optional (default=1.0)*
  - 與正規化成反比的值
  - 值越小， 正規化影響越大

步驟二: 產生一個 LogisticRegression 實例

class sklearn.linear_model.LogisticRegression(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)*

- **random_state**
  - *int, RandomState instance or None, optional (default=None)*
  - 在solver == 'sag' or 'liblinear'才會使用

步驟二: 產生一個 LogisticRegression 實例

class sklearn.linear_model.LogisticRegression(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)*

- **solver**
  - *str, {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, optional (default='lbfgs')*
  - 'newton-cg', 'lbfgs', 'sag' and 'saga' 適用在 L_2 或沒有懲罰
  - 'liblinear' and 'saga'也適用在 L_1懲罰
  - 'saga'也適用在 'elasticnet'懲罰
  - 'liblinear' 不適用在penalty='none'的情形

步驟二: 產生一個 LogisticRegression 實例

decision_function(self, X)	Predict confidence scores for samples.
densify(self)	Convert coefficient matrix to dense array format.
fit(self, X, y[, sample_weight])	Fit the model according to the given training data.
get_params(self[, deep])	Get parameters for this estimator.
predict(self, X)	Predict class labels for samples in X.
predict_log_proba(self, X)	Predict logarithm of probability estimates.
predict_proba(self, X)	Probability estimates.
score(self, X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
set_params(self, **params)	Set the parameters of this estimator.
sparsify(self)	Convert coefficient matrix to sparse format.

步驟二: 產生一個 LogisticRegression 實例

```
from sklearn.linear_model import LogisticRegression
LOGRObject = LogisticRegression(C=1, solver = 'lbfgs')
LOGRFitting = LOGRObject.fit(X,y)
```

```
import numpy as np
X_Test = np.array([[1,2]])
LOGRFitting.decision_function(X_Test)
```

```
array([0.96556035])
```

```
LOGRFitting.coef_
```

```
array([[-0.07856755, -1.79309266]])
```

```
LOGRFitting.intercept_
```

```
array([4.63031323])
```

```
LOGRFitting.coef_[0,0]*1+LOGRFitting.coef_[0,1]*2+LOGRFitting.intercept_
```

```
array([0.96556035])
```

步驟三: 視覺化

```
x_min, x_max = X[:,0].min()-1, X[:,0].max()+1
y_min, y_max = X[:,1].min()-1, X[:,1].max()+1
```

```
import numpy as np
step_size = 0.02
x_grid, y_grid = np.meshgrid(np.arange(x_min,x_max,step_size),np.arange(y_min,y_max,step_size))
```

```
X_Samples = np.c_[x_grid.ravel(),y_grid.ravel()]
X_Samples.shape
```

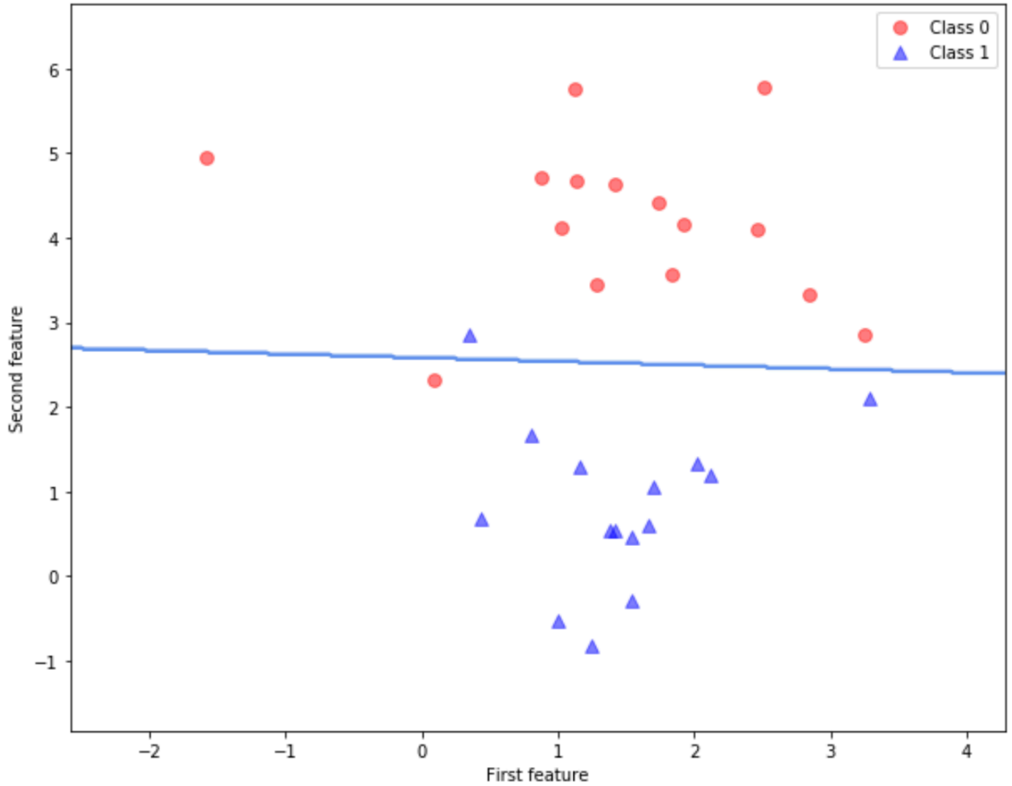
```
(148264, 2)
```

```
Prediction_results_X_Samples = LOGRFitting.predict(X_Samples)
```

步驟三: 視覺化

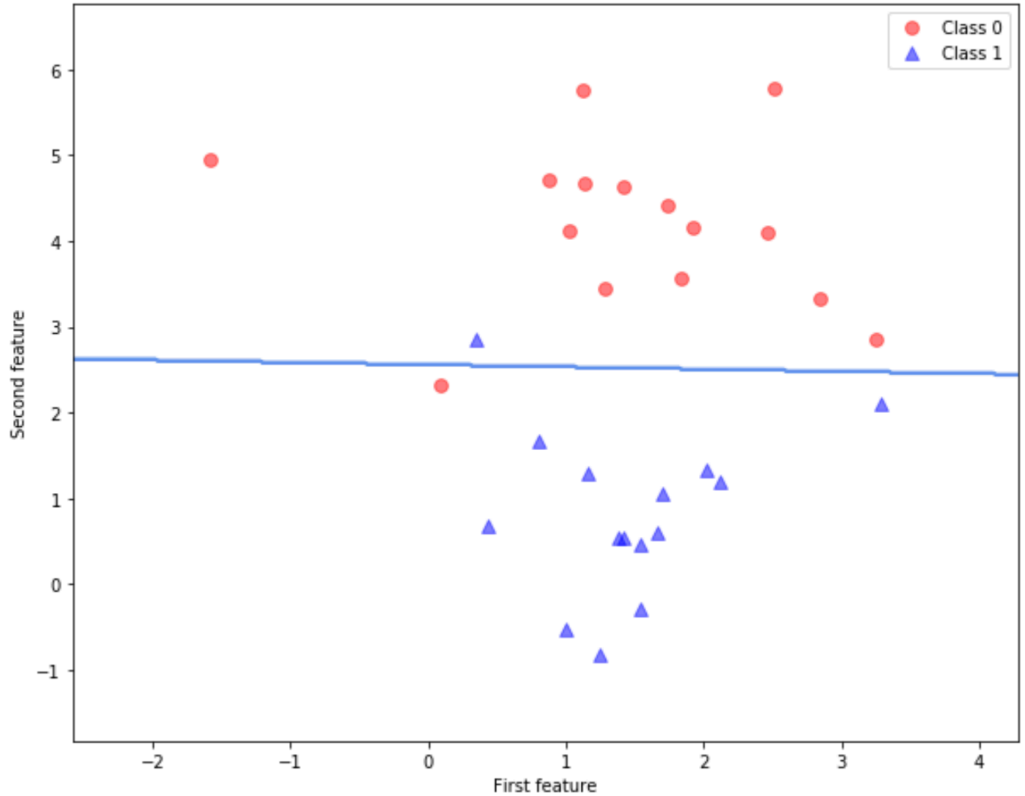
```
from sklearn.datasets import make_blobs
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt


plt.figure(figsize=(10,8))
Prediction_results_X_Samples = Prediction_results_X_Samples.reshape(x_grid.shape)
plt.contour(x_grid, y_grid, Prediction_results_X_Samples, colors = 'cornflowerblue')
plt.scatter(X[y==0, 0], X[y==0, 1], marker='o', c='red',s=60, alpha=0.5, label='Class 0')
plt.scatter(X[y==1, 0], X[y==1, 1], marker='^', c='blue',s=60, alpha=0.5, label='Class 1')
plt.xlabel('First feature')
plt.ylabel('Second feature')
plt.legend(loc='upper right')
plt.show()
```

步驟三: 視覺化


Class 0
Class 1
6
5
4
3
2
1
0
−1
Second feature
−2
−1
0
1
2
3
4
First feature

C=1

步驟三: 視覺化



Class 0
Class 1
Second feature
First feature
6
5
4
3
2
1
0
−1
−2
−1
0
1
2
3
4

C=10

線性分類模型

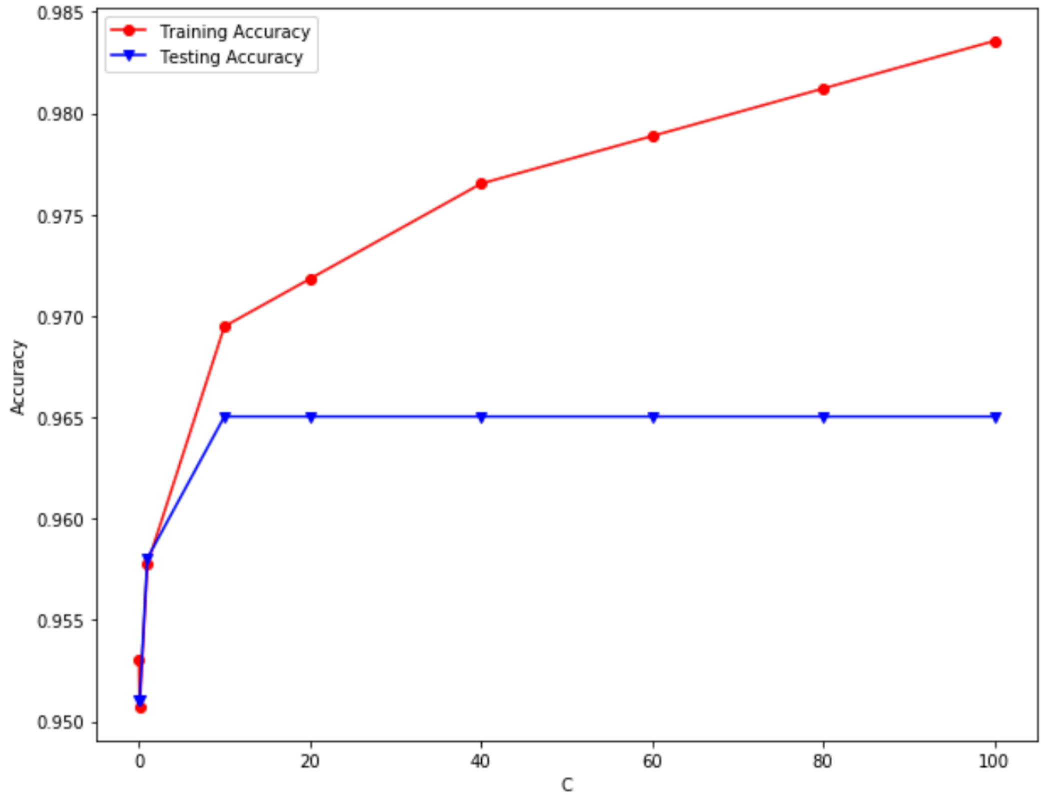
Logistic Regression

Breast Cancer dataset

Breast Cancer dataset

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
cancer.data, cancer.target, stratify=cancer.target, random_state=42)

Training_Accuracy = []
Testing_Accuracy = []
Cvalue_array = np.array([0.01, 0.1, 1, 10, 20, 40, 60, 80, 100])
for Cvalue in Cvalue_array:
    LOGRObject = LogisticRegression(C=Cvalue, solver = 'lbfgs', max_iter=10000)
    LOGRFitting = LOGRObject.fit(X_train,y_train)
    Training_Accuracy.append(LOGRFitting.score(X_train,y_train))
    Testing_Accuracy.append(LOGRFitting.score(X_test,y_test))

plt.figure(figsize=(10,8))
plt.plot(Cvalue_array,Training_Accuracy, color = 'red', marker = 'o', label ='Training Accuracy')
plt.plot(Cvalue_array,Testing_Accuracy, color = 'blue', marker = 'v', label ='Testing Accuracy')

plt.xlabel('C')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Breast Cancer dataset



課堂練習
Pima Indian Diabetes dataset

```
import pandas as pd
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
pima = pd.read_csv("pima-indians-diabetes.csv", header=None, names=col_names)
```

```
pima.head()
```

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0

```
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X_Temp = pima[feature_cols]
X = X_Temp[1:].values
y_Temp = pima.label
y = y_Temp[1:].values
```

課堂練習
Pima Indian Diabetes dataset

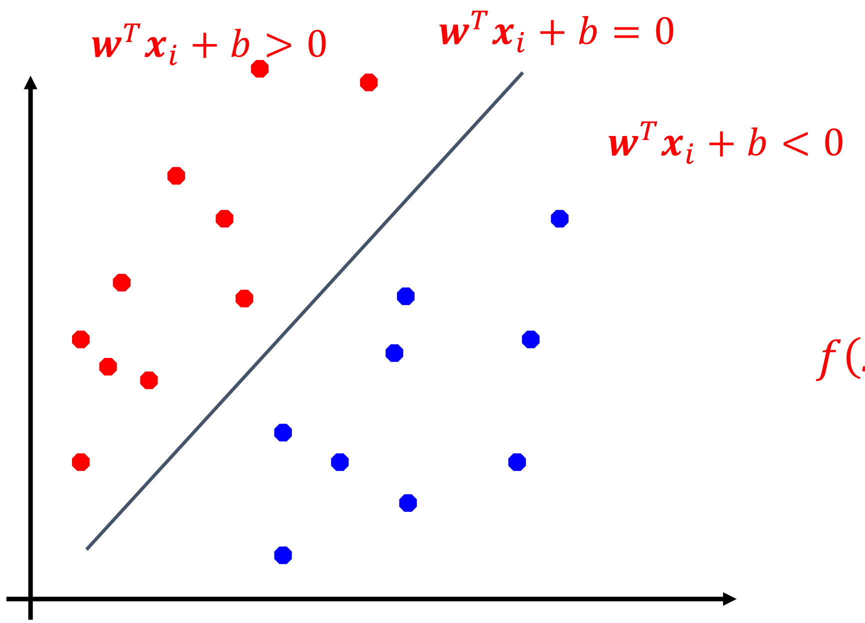
- Use logistic regression to build a prediction model

線性分類模型

Linear SVC

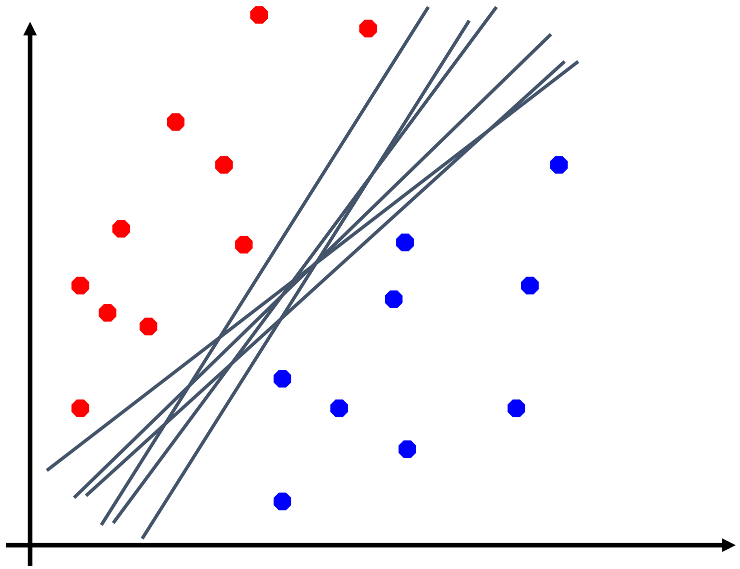
基礎

線性分隔



$$f(\boldsymbol{x}_\mathrm{i}) = \mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}_i + b)$$

線性分隔

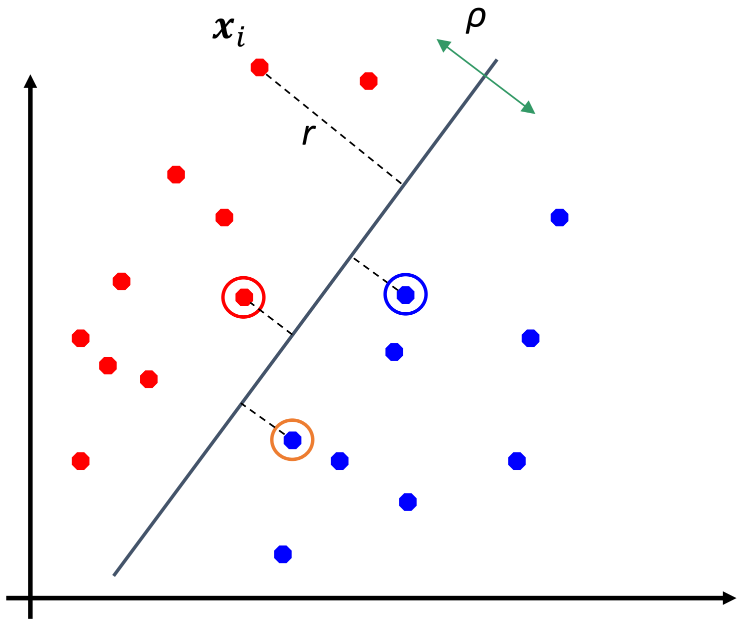


類別間距

- $\boldsymbol{x}_i$ 到決策邊間的距離 r

$$r = \frac{\boldsymbol{w}^T \boldsymbol{x}_i + b}{\|\boldsymbol{w}\|}$$

- 圈出來的資料點我們稱為支持向量 ***(support vectors)***
- 間距 ρ 則為不同類別支持向量間的對大距離
- 我們希望間距 ρ 能越大越好
- 執得注意的是在linear SVM只有支持向量對於決策是有影響的



線性分類模型

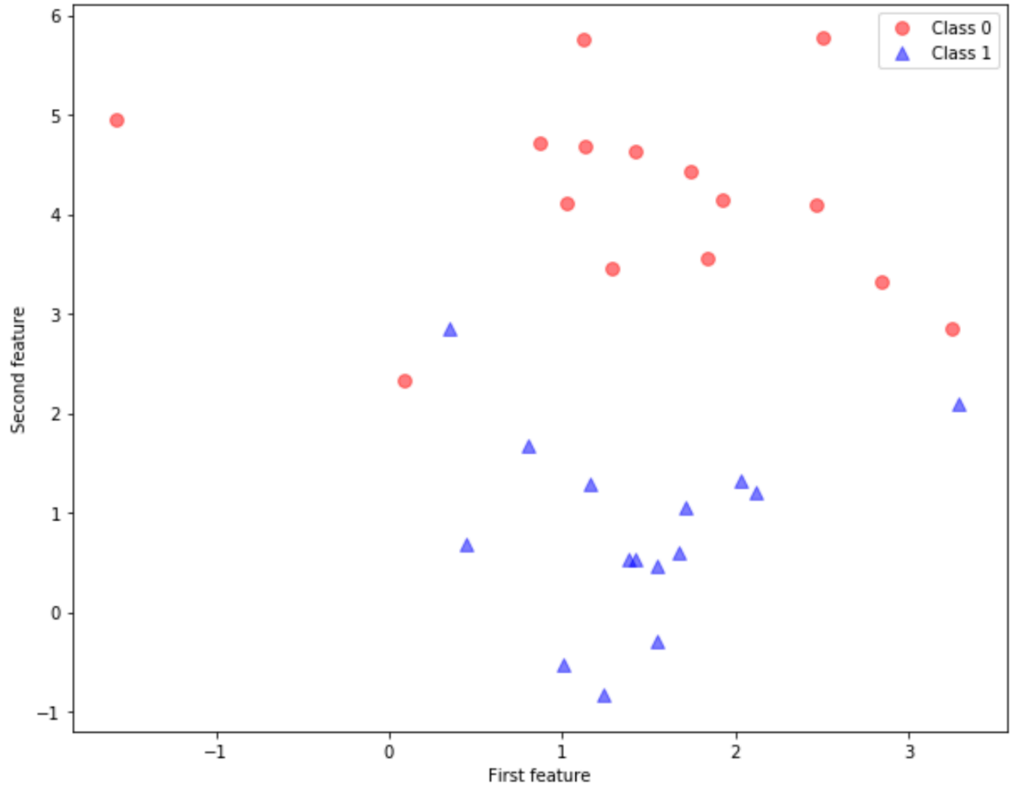
Linear SVC

Implementation

步驟一: 資料的收集或產生

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

X, y = make_blobs(n_samples=30, centers=2, n_features=2,random_state=0)

plt.figure(figsize=(10,8))

plt.scatter(X[y==0, 0], X[y==0, 1], marker='o', c='red',s=60, alpha=0.5, label='Class 0')
plt.scatter(X[y==1, 0], X[y==1, 1], marker='^', c='blue',s=60, alpha=0.5, label='Class 1')

plt.xlabel('First feature')
plt.ylabel('Second feature')
plt.legend(loc='upper right')
plt.show()
```

步驟一: 資料的收集或產生



步驟二: 產生一個 LinearSVC 實例

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*)

- Linear Support Vector Classification
- **penalty**
  - *str, 'l1' or 'l2' (default='l2')*
- **loss**
  - *str, 'hinge' or 'squared_hinge' (default='squared_hinge')*

步驟二: 產生一個 LinearSVC 實例

`decision_function`(self, X)	Predict confidence scores for samples.
`densify`(self)	Convert coefficient matrix to dense array format.
`fit`(self, X, y[, sample_weight])	Fit the model according to the given training data.
`get_params`(self[, deep])	Get parameters for this estimator.
`predict`(self, X)	Predict class labels for samples in X.
`score`(self, X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
`set_params`(self, **params)	Set the parameters of this estimator.
`sparsify`(self)	Convert coefficient matrix to sparse format.

步驟二: 產生一個 LinearSVC 實例

```
from sklearn.svm import LinearSVC
LSVCObject = LinearSVC(C=1,max_iter=10000)
LSVCFitting = LSVCObject.fit(X,y)
```

```
import numpy as np
X_Test = np.array([[1,2]])
LSVCFitting.predict(X_Test)
```

```
array([1])
```

步驟二: 產生一個 LinearSVC 實例

```
DF = LSVCFitting.decision_function(X)
DF
```

```
array([-0.36990725, -1.51582992,  1.08741977,  0.90487158, -0.72511322,
       -1.56059105, -2.22219053,  1.79411285,  0.72185629, -1.09992908,
        0.44120049, -0.56228371,  0.73874507, -1.59567545,  0.28392385,
       -1.8851103 , -1.18665052, -0.2212399 , -1.35714668, -1.1640906 ,
       -0.77184229,  0.82812765,  2.14487367,  1.21077475,  1.29562535,
        1.23451499, -0.03055401,  1.9271162 , -2.28364469,  1.23862123])
```

```
LSVCFitting.intercept_
```

```
array([1.52452103])
```

```
LSVCFitting.coef_
```

```
array([[ 0.05154493, -0.67139134]])
```

```
LSVCFitting.coef_@X.transpose()+LSVCObject.intercept_
```

```
array([[-0.36990725, -1.51582992,  1.08741977,  0.90487158, -0.72511322,
        -1.56059105, -2.22219053,  1.79411285,  0.72185629, -1.09992908,
         0.44120049, -0.56228371,  0.73874507, -1.59567545,  0.28392385,
        -1.8851103 , -1.18665052, -0.2212399 , -1.35714668, -1.1640906 ,
        -0.77184229,  0.82812765,  2.14487367,  1.21077475,  1.29562535,
         1.23451499, -0.03055401,  1.9271162 , -2.28364469,  1.23862123]])
```

步驟二: 產生一個 LinearSVC 實例

```
LSVCFitting.coef_@X.transpose()+LSVCObject.intercept_>0
```

```
array([[False, False,  True,  True, False, False, False,  True,  True,
        False,  True, False,  True, False,  True, False, False, False,
        False, False, False,  True,  True,  True,  True,  True, False,
         True, False,  True]])
```

```
LSVCFitting.predict(X)
```

```
array([0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 1])
```

步驟三: 視覺化

```
x_min, x_max = X[:,0].min()-1, X[:,0].max()+1
y_min, y_max = X[:,1].min()-1, X[:,1].max()+1
```

```
import numpy as np
step_size = 0.02
x_grid, y_grid = np.meshgrid(np.arange(x_min,x_max,step_size),np.arange(y_min,y_max,step_size))
```

```
X_Samples = np.c_[x_grid.ravel(),y_grid.ravel()]
X_Samples.shape
```

```
(148264, 2)
```

```
Prediction_results_X_Samples = LSVCFitting.predict(X_Samples)
print(f'Prediction_results_X_Samples.shape is {Prediction_results_X_Samples.shape}')
```

```
Prediction_results_X_Samples.shape is (148264,)
```

```
Prediction_results_X_Samples = Prediction_results_X_Samples.reshape(x_grid.shape)
print(f'Prediction_results_X_Samples.shape is {Prediction_results_X_Samples.shape}')
```

```
Prediction_results_X_Samples.shape is (431, 344)
```

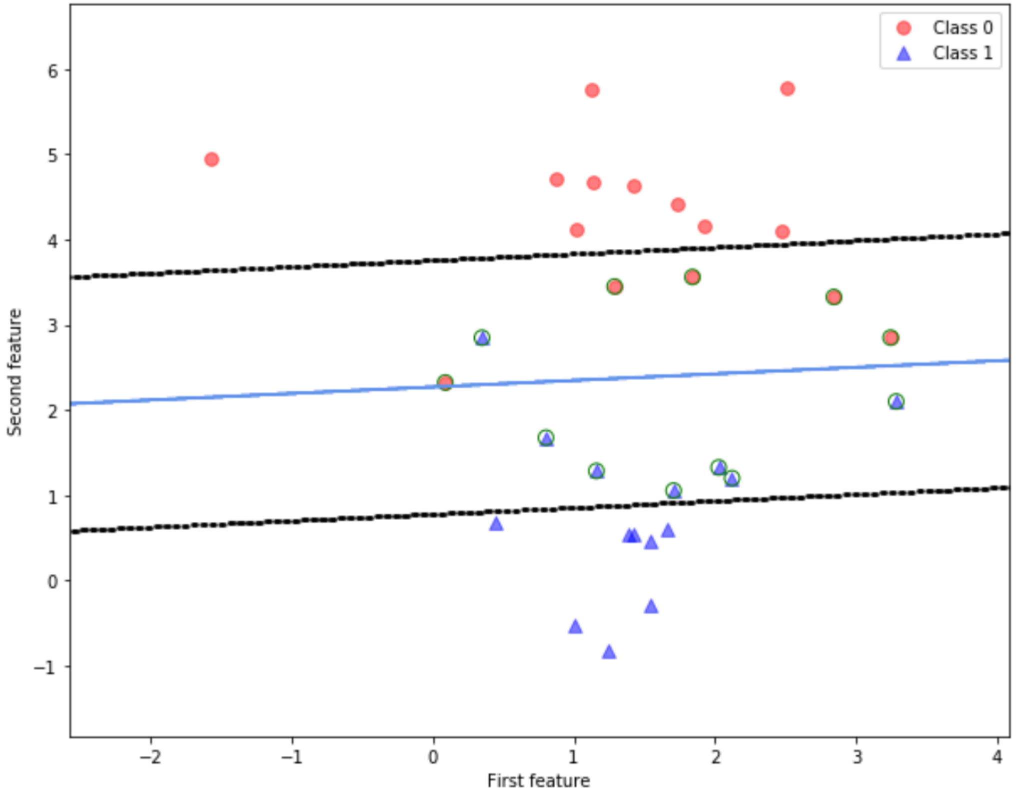
```
DF = LSVCFitting.decision_function(X)
Support_Vector_Candidate = ((2*y-1)*DF<=1)
Support_Vector = X[Support_Vector_Candidate]
Support_Vector_Class = y[Support_Vector_Candidate]
```

步驟三: 視覺化

```
import numpy as np
DF_X_samples = LSVCFitting.decision_function(X_Samples)
Boundary_index01 = ((np.abs(DF_X_samples-1)<0.001))
Boundary_index02 = ((np.abs(DF_X_samples+1)<0.001))
```

```
from sklearn.datasets import make_blobs
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt

plt.figure(figsize=(10,8))
plt.contour(x_grid, y_grid, Prediction_results_X_Samples, colors = 'cornflowerblue')
plt.scatter(X[y==0, 0], X[y==0, 1], marker='o', c='red',s=60, alpha=0.5, label='Class 0')
plt.scatter(X[y==1, 0], X[y==1, 1], marker='^', c='blue',s=60, alpha=0.5, label='Class 1')
Index = Support_Vector_Class==0
plt.scatter(Support_Vector[Index,0], Support_Vector[Index,1], facecolors='none',s=80, edgecolors='green')
Index = Support_Vector_Class==1
plt.scatter(Support_Vector[Index,0], Support_Vector[Index,1], facecolors='none',s=80, edgecolors='green')

plt.scatter(X_Samples[Boundary_index01,0],X_Samples[Boundary_index01,1], c='black', s=8, alpha=0.5)
plt.scatter(X_Samples[Boundary_index02,0],X_Samples[Boundary_index02,1], c='black', s=8, alpha=0.5)

plt.xlim(x_min,x_max-0.2)
plt.xlabel('First feature')
plt.ylabel('Second feature')
plt.legend(loc='upper right')
plt.show()
```

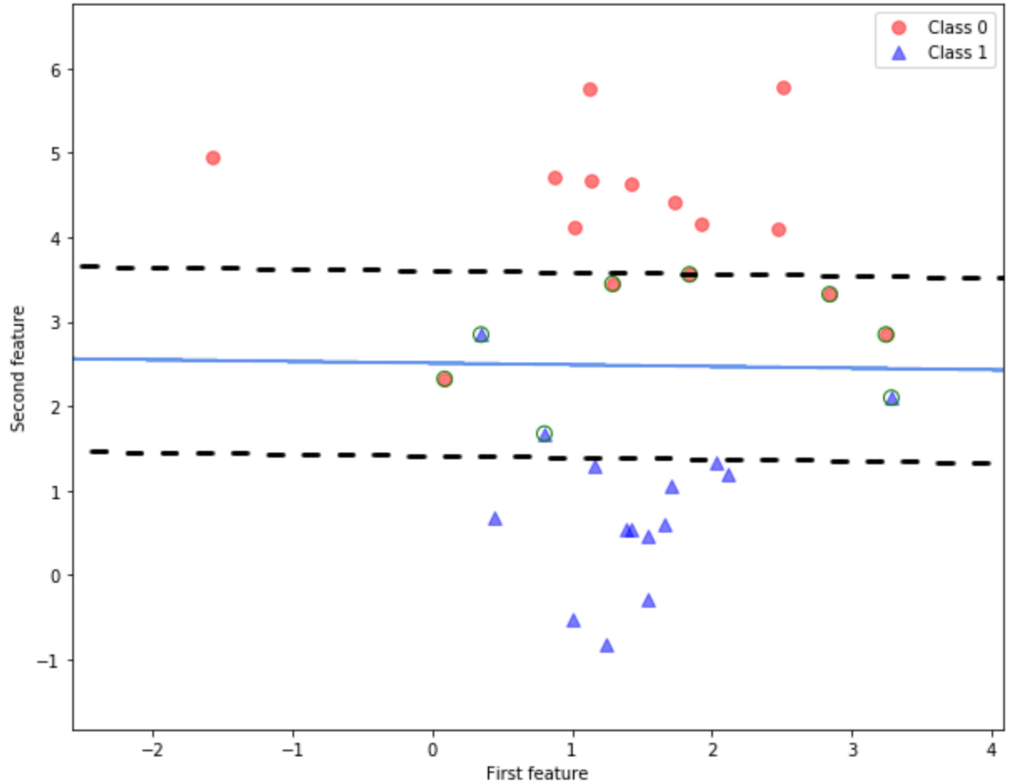
步驟三: 視覺化



Class 0
Class 1
Second feature
First feature
6
5
4
3
2
1
0
−1
−2
−1
0
1
2
3
4

C=1

步驟三: 視覺化


Class 0
Class 1
6
5
4
3
2
1
0
−1
Second feature
−2
−1
0
1
2
3
4
First feature

C=10

課堂練習
Breast Cancer dataset

- 利用 LinearSVC 建構乳癌預測模型

課堂練習
Digits Classification

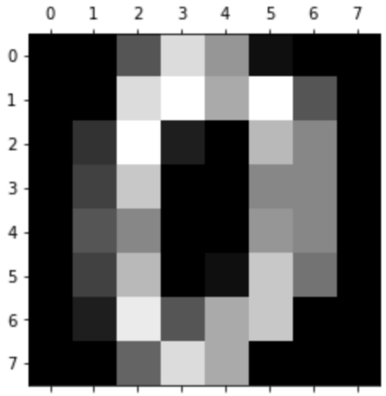
```
from sklearn.datasets import load_digits
digits = load_digits()
```

```
print(digits.data.shape)
```

```
(1797, 64)
```

```
import matplotlib.pyplot as plt
plt.gray()
plt.matshow(digits.images[0])
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



課堂練習
Digits Classification

- 利用 logistic regression 建構手寫數字預測模型
- 利用LinearSVC 建構手寫數字預測模型

Q&A