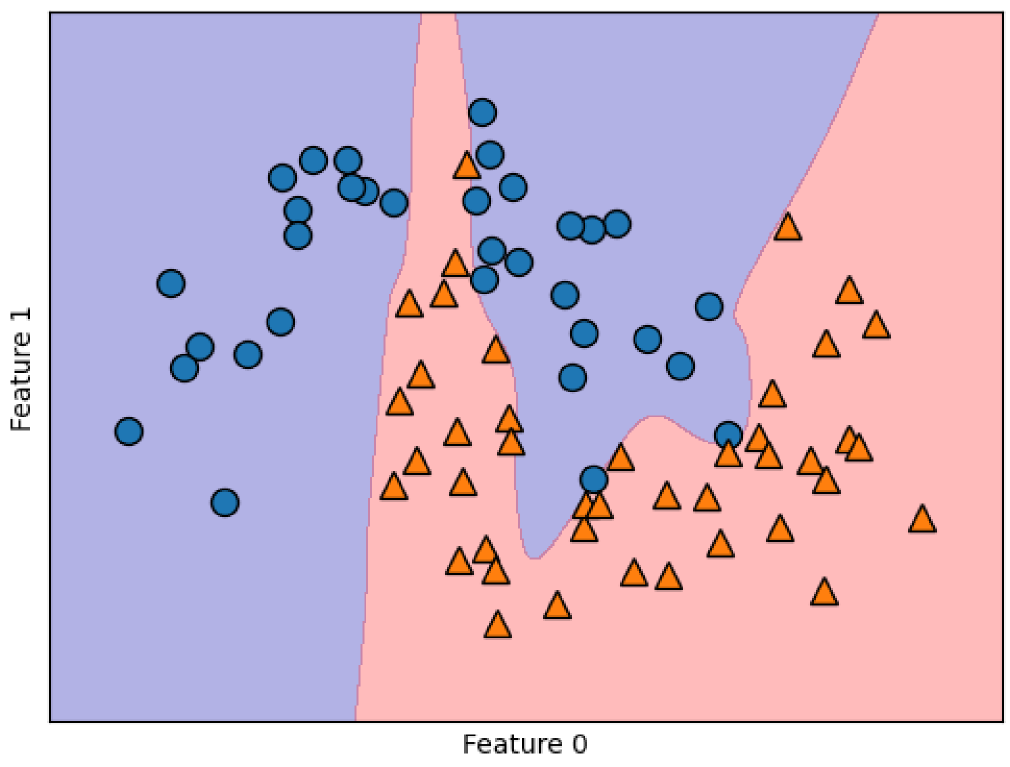
Lab8

November 24, 2024

0.0.1 1. Neural networks

```
[1]: from sklearn.neural_network import MLPClassifier
     from sklearn.datasets import make_moons
     from sklearn.model_selection import train_test_split
     import mglearn
     %matplotlib inline
     import matplotlib.pyplot as plt

     X,y = make_moons(n_samples=100, noise=0.25, random_state=42)
     X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=42)
     mlp = MLPClassifier(solver='lbfgs', activation='tanh', random_state=42,
                         hidden_layer_sizes=[10]).fit(X_train, y_train)

     mglearn.plots.plot_2d_separator(mlp, X_train, fill=True, alpha=0.3)
     mglearn.discrete_scatter(X_train[:,0], X_train[:,1], y_train)
     plt.xlabel("Feature 0")
     plt.ylabel("Feature 1")
```

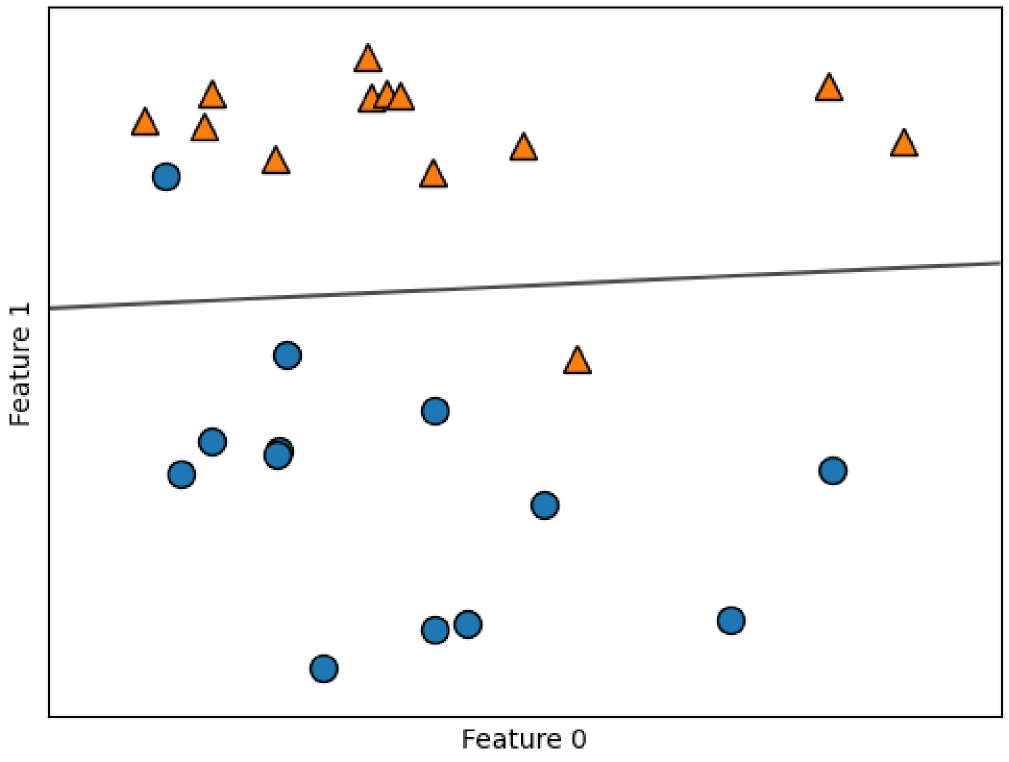
```
[1]: Text(0, 0.5, 'Feature 1')
```



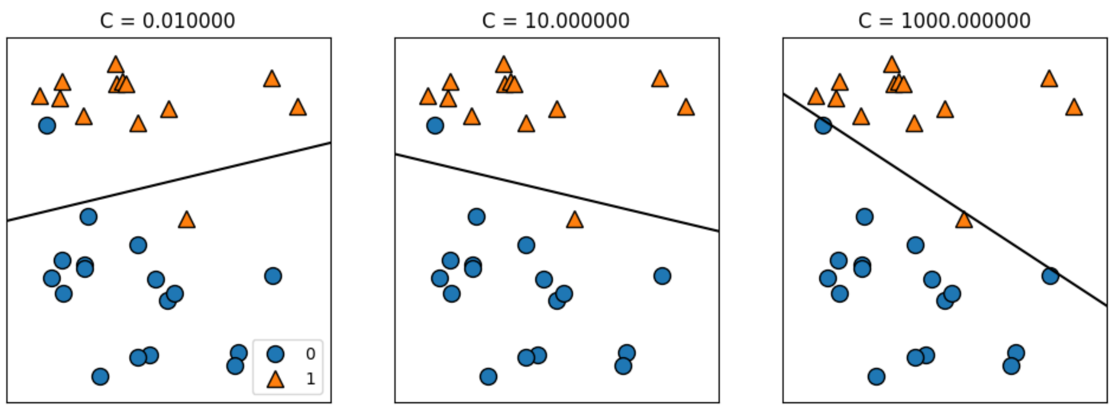
0.0.2 2. Linear SVM

[2]:
```
from sklearn.svm import LinearSVC
X, y = mglearn.datasets.make_forge()
svm = LinearSVC().fit(X, y)
mglearn.plots.plot_2d_separator(svm, X, fill=False, eps=0.5, alpha=0.7)
mglearn.discrete_scatter(X[:,0], X[:,1], y)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```

[2]:
```
Text(0, 0.5, 'Feature 1')
```



[3]:
```
mglearn.plots.plot_linear_svc_regularization()
```



0.0.3 3. Kernel SVM

[4]:
```
from sklearn.svm import SVC
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
```

```python
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,␣
↪random_state=42)
svc = SVC()
svc.fit(X_train, y_train)
print("Accuracy on training set:", svc.score(X_train, y_train))
print("Accuracy on test set:", svc.score(X_test, y_test))
```

```
Accuracy on training set: 0.9107981220657277
Accuracy on test set: 0.951048951048951
```

[5]:
```python
print(cancer["DESCR"])
print(cancer["feature_names"])
print(cancer["data"])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

:Number of Instances: 569

:Number of Attributes: 30 numeric, predictive attributes and the class

:Attribute Information:
    - radius (mean of distances from center to points on the perimeter)
    - texture (standard deviation of gray-scale values)
    - perimeter
    - area
    - smoothness (local variation in radius lengths)
    - compactness (perimeter^2 / area - 1.0)
    - concavity (severity of concave portions of the contour)
    - concave points (number of concave portions of the contour)
    - symmetry
    - fractal dimension ("coastline approximation" - 1)

    The mean, standard error, and "worst" or largest (mean of the three
    worst/largest values) of these features were computed for each image,
    resulting in 30 features.  For instance, field 0 is Mean Radius, field
    10 is Radius SE, field 20 is Worst Radius.

    - class:
            - WDBC-Malignant
            - WDBC-Benign

:Summary Statistics:
```

```
===================================== ====== ======
                                       Min    Max
===================================== ====== ======
radius (mean):                        6.981  28.11
texture (mean):                       9.71   39.28
perimeter (mean):                     43.79  188.5
area (mean):                          143.5  2501.0
smoothness (mean):                    0.053  0.163
compactness (mean):                   0.019  0.345
concavity (mean):                     0.0    0.427
concave points (mean):                0.0    0.201
symmetry (mean):                      0.106  0.304
fractal dimension (mean):             0.05   0.097
radius (standard error):              0.112  2.873
texture (standard error):             0.36   4.885
perimeter (standard error):           0.757  21.98
area (standard error):                6.802  542.2
smoothness (standard error):          0.002  0.031
compactness (standard error):         0.002  0.135
concavity (standard error):           0.0    0.396
concave points (standard error):      0.0    0.053
symmetry (standard error):            0.008  0.079
fractal dimension (standard error):   0.001  0.03
radius (worst):                       7.93   36.04
texture (worst):                      12.02  49.54
perimeter (worst):                    50.41  251.2
area (worst):                         185.2  4254.0
smoothness (worst):                   0.071  0.223
compactness (worst):                  0.027  1.058
concavity (worst):                    0.0    1.252
concave points (worst):               0.0    0.291
symmetry (worst):                     0.156  0.664
fractal dimension (worst):            0.055  0.208
===================================== ====== ======

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2
```

```
Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. dropdown:: References

  - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
    for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
    Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
    San Jose, CA, 1993.
  - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
    prognosis via linear programming. Operations Research, 43(4), pages 570-577,
    July-August 1995.
  - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
    to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
    163-171.

['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

```
[[1.799e+01 1.038e+01 1.228e+02 … 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 … 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 … 2.430e-01 3.613e-01 8.758e-02]
 …
 [1.660e+01 2.808e+01 1.083e+02 … 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 … 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 … 0.000e+00 2.871e-01 7.039e-02]]
```

[6]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
svc.fit(X_train_scaled, y_train)
print("Accuracy on training set:", svc.score(X_train_scaled, y_train))
print("Accuracy on test set:", svc.score(X_test_scaled, y_test))
```

```
Accuracy on training set: 0.9835680751173709
Accuracy on test set: 0.9790209790209791
```

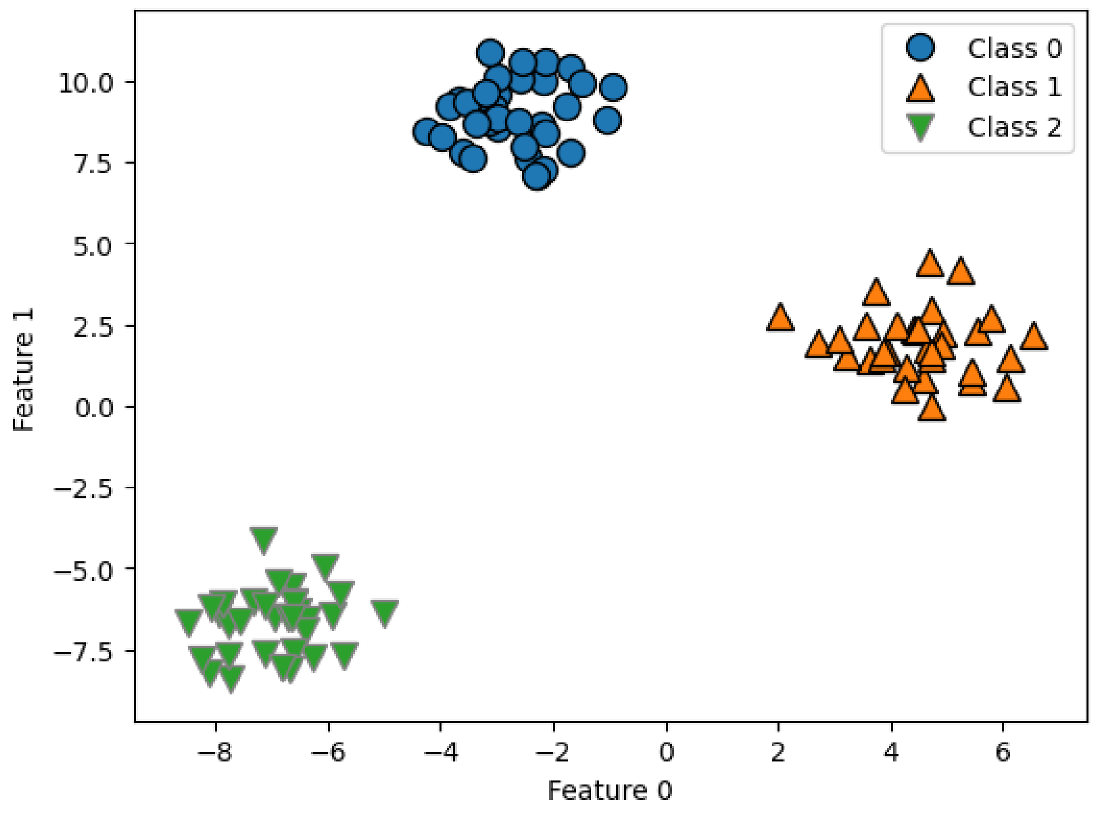
[7]:
```
svc=SVC(C=1000)
svc.fit(X_train_scaled, y_train)
print("Accuracy on training set:", svc.score(X_train_scaled, y_train))
print("Accuracy on test set:", svc.score(X_test_scaled, y_test))
```

```
Accuracy on training set: 1.0
Accuracy on test set: 0.951048951048951
```

0.0.4 4. Multiclass classification

[8]:
```
from sklearn.datasets import make_blobs
X, y = make_blobs(random_state=42)
mglearn.discrete_scatter(X[:,0], X[:,1], y)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
plt.legend(["Class 0", "Class 1", "Class 2"])
```

[8]:
```
<matplotlib.legend.Legend at 0x72a31cba5ed0>
```

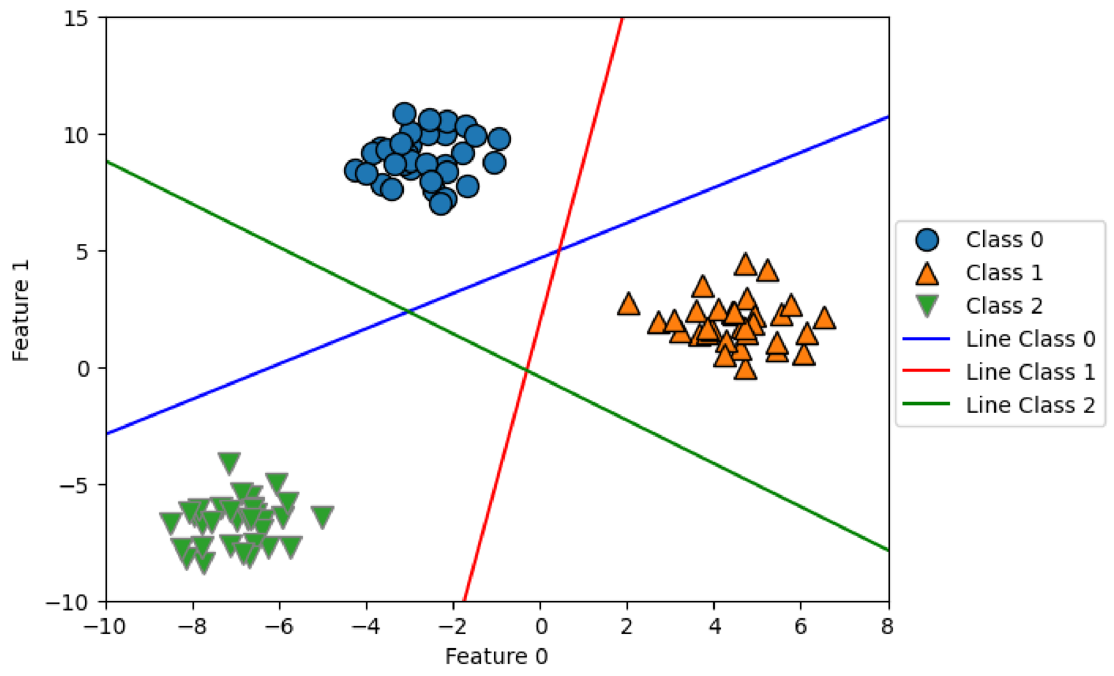


[9]:
```
svm = LinearSVC().fit(X, y)
print("Coefficient shape:", svm.coef_.shape)
print("Intercept shape:", svm.intercept_.shape)
```

```
Coefficient shape: (3, 2)
Intercept shape: (3,)
```

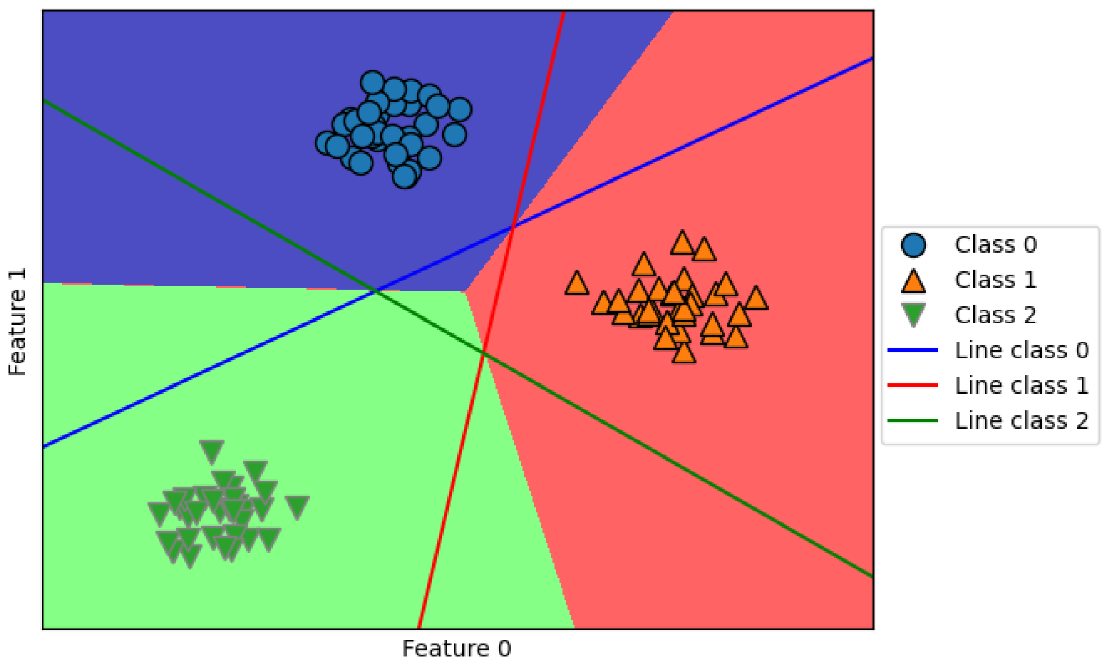
[10]:
```
import numpy as np
mglearn.discrete_scatter(X[:,0], X[:,1], y)
line = np.linspace(-15, 15)
for coef, intercept, color in zip(svm.coef_, svm.intercept_, ['b', 'r', 'g']):
    plt.plot(line, -(line*coef[0]+intercept) / coef[1], c=color)

plt.ylim(-10, 15)
plt.xlim(-10, 8)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
plt.legend(['Class 0', 'Class 1', 'Class 2', 'Line Class 0', 'Line Class 1',␣
 ↪'Line Class 2'], loc=(1.01, 0.3))
```

[10]:
```
<matplotlib.legend.Legend at 0x72a31cbd4510>
```



```
[11]: mglearn.plots.plot_2d_classification(svm, X, fill=True, alpha=.7)
      mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
      line = np.linspace(-15, 15)
      for coef, intercept, color in zip(svm.coef_, svm.intercept_, ['b', 'r', 'g']):
          plt.plot(line, -(line * coef[0] + intercept) / coef[1], c=color)
      plt.legend(['Class 0', 'Class 1', 'Class 2', 'Line class 0',
      'Line class 1', 'Line class 2'], loc=(1.01, 0.3))
      plt.xlabel("Feature 0")
      plt.ylabel("Feature 1")
```

```
[11]: Text(0, 0.5, 'Feature 1')
```



0.0.5 5. Exercises

1. Because for Class 0, the values for feature 0 are negative while the values for feature 1 are positive. For class 1, the values of feature 1 are positive while the values for feature 0 are negative. For class 2, the values of both features are negative so the signs of the coefficients reflect those attributes.

[12]:
```
print(svm.coef_)
```

```
[[-0.17492412  0.23140766]
 [ 0.47622012 -0.06936786]
 [-0.18914207 -0.20400079]]
```

2. Zip combines the arrays of coefficients, intercepts and colors into their separate classes. So for example, coefficients of class 0 are combined with intercepts of class 0 along with the color of class 0 (b). Class 1 coefficients would be combined with class 1 intercepts along with the color r. Although the term "class" is more referring to the index location of how the coefficient and intercept are separated.
3. While the first entry in the plot() function is are the X_values, the second parameter slot is for the calculations of the Y_values. the equation is for calculating the slope of the boundary lines for classfying the various classes.