plot_logistic_multinomial

May 29, 2023

[1]:
```
%matplotlib inline
```

1 Plot multinomial and One-vs-Rest Logistic Regression

Plot decision surface of multinomial and One-vs-Rest Logistic Regression. The hyperplanes corresponding to the three One-vs-Rest (OVR) classifiers are represented by the dashed lines.

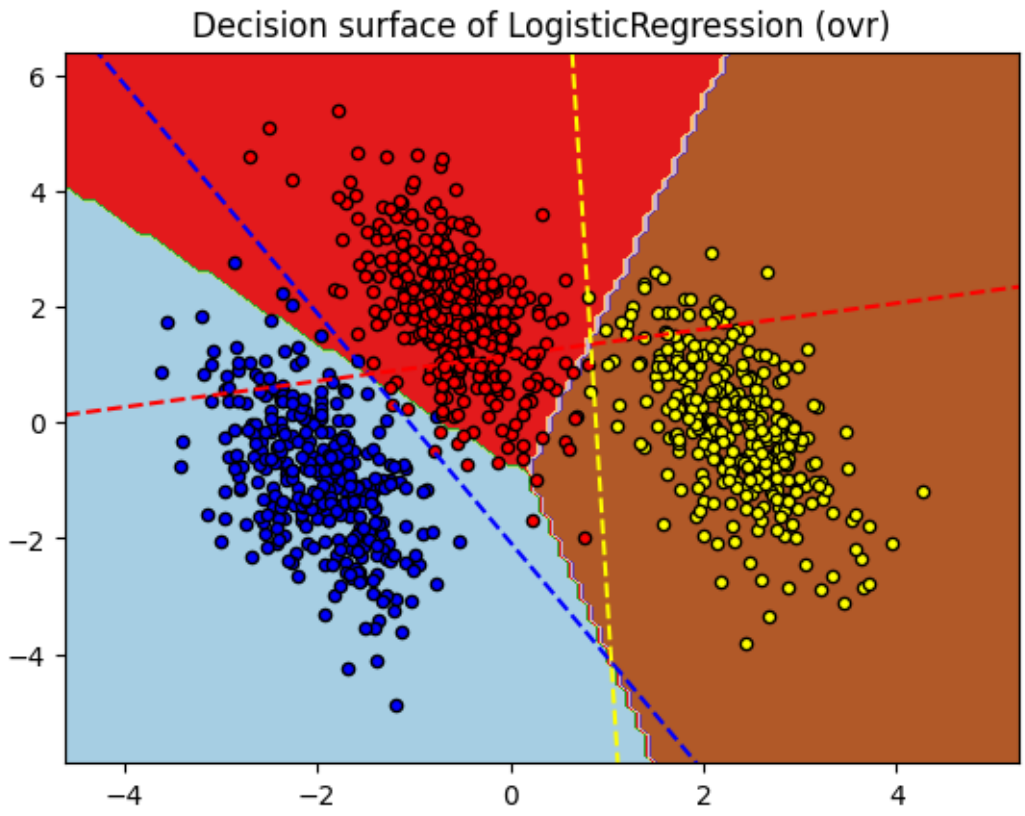
[81]:
```
# Authors: Tom Dupre la Tour <tom.dupre-la-tour@m4x.org>
# License: BSD 3 clause

import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.linear_model import LogisticRegression
from sklearn.inspection import DecisionBoundaryDisplay

# make 3-class dataset for classification
centers = [[-5, 0], [0, 1.5], [5, -1]]
X, y = make_blobs(n_samples=1000, centers=centers, random_state=40)
transformation = [[0.4, 0.2], [-0.4, 1.2]]
X = np.dot(X, transformation)



for multi_class in ("multinomial", "ovr"):
    clf = LogisticRegression(
        solver="sag", max_iter=100, random_state=42, multi_class=multi_class
    ).fit(X, y)

    # print the training scores
    print("training score : %.3f (%s)" % (clf.score(X, y), multi_class))

    _, ax = plt.subplots()
    DecisionBoundaryDisplay.from_estimator(
        clf, X, response_method="predict", cmap=plt.cm.Paired, ax=ax
    )
    plt.title("Decision surface of LogisticRegression (%s)" % multi_class)
```

```
    plt.axis("tight")

    # Plot also the training points
    colors = ["blue", "red", "yellow"]
    for i, color in zip(clf.classes_, colors):
        idx = np.where(y == i)
        plt.scatter(
            X[idx, 0], X[idx, 1], c=color, cmap=plt.cm.Paired,␣
 ↪edgecolor="black", s=20
        )
    # Plot the three one-against-all classifiers
    xmin, xmax = plt.xlim()
    ymin, ymax = plt.ylim()
    params = (clf.coef_, clf.intercept_)
    coef = clf.coef_
    intercept = clf.intercept_
    def plot_hyperplane(c, color):
        def line(x0):
            return (-(x0 * coef[c, 0]) - intercept[c]) / coef[c, 1]

        plt.plot([xmin, xmax], [line(xmin), line(xmax)], ls="--", color=color)

    for i, color in zip(clf.classes_, colors):
        plot_hyperplane(i, color)

plt.show()
```

```
training score : 0.995 (multinomial)
(-4.611241757981748, 5.269091007054253) (-5.88341015185696, 6.406472714716773)
training score : 0.976 (ovr)
(-4.611241757981748, 5.269091007054253) (-5.88341015185696, 6.406472714716773)
```

```
C:\Users\pyk93\AppData\Local\Temp\ipykernel_289596\1692758557.py:37:
UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will
be ignored
  plt.scatter(
C:\Users\pyk93\AppData\Local\Temp\ipykernel_289596\1692758557.py:37:
UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will
be ignored
  plt.scatter(
```


Decision surface of LogisticRegression (multinomial)
6
4
2
0
−2
−4
−4
−2
0
2
4

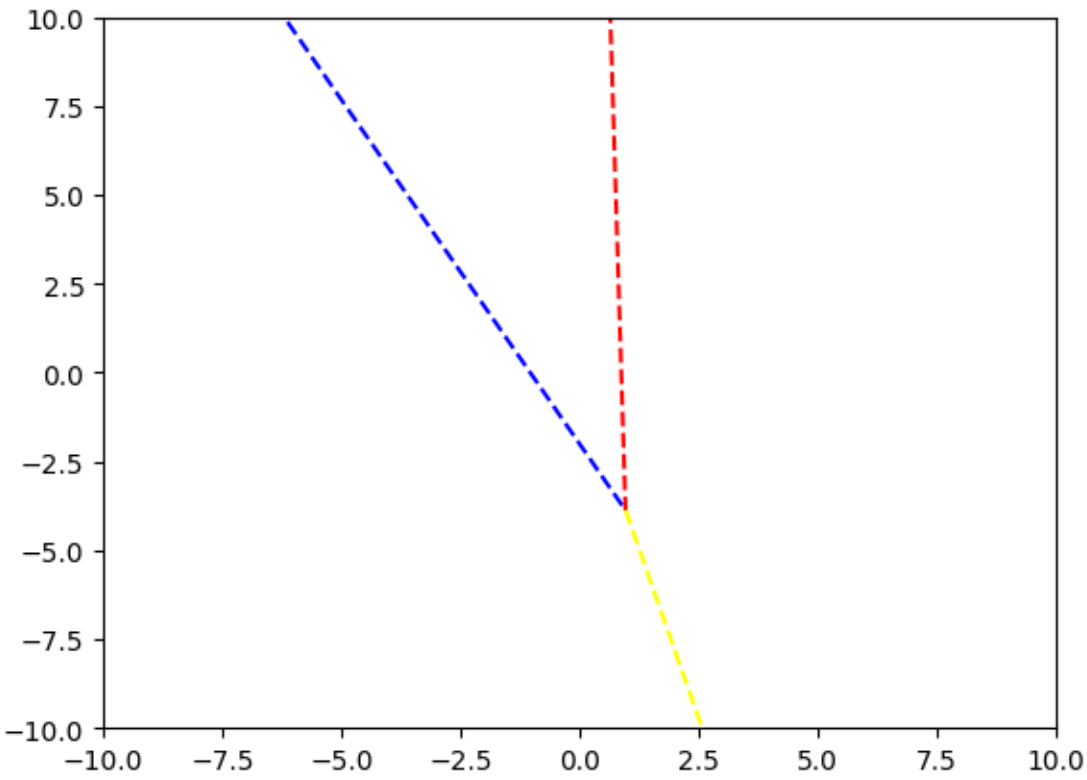


[86]:
```
for multi_class in ["multinomial","ovr"]:
    clf = LogisticRegression(
        solver="sag", max_iter=100, random_state=42, multi_class=multi_class
    ).fit(X, y)

    # print the training scores
    print("training score : %.3f (%s)" % (clf.score(X, y), multi_class))

    _, ax = plt.subplots()
    '''
    DecisionBoundaryDisplay.from_estimator(
        clf, X, response_method="predict", cmap=plt.cm.PiYG, ax=ax
    )
    plt.title("Decision surface of LogisticRegression (%s)" % multi_class)

    '''

    params = (clf.coef_, clf.intercept_)

    # get center
```

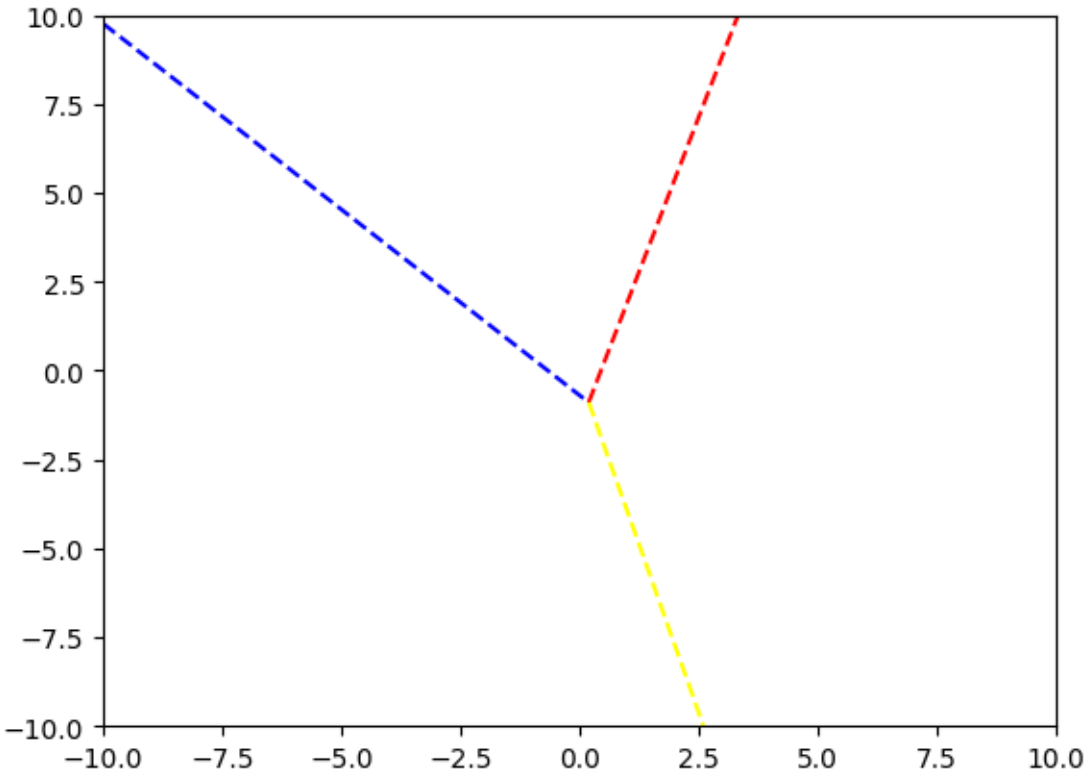
```
    c2 = np.concatenate((coef_set,cc3),axis=1)

    center_value = np.dot(np.linalg.inv(c2),-intercept_set)

    coef_set, intercept_set = params

    coef_boundary = np.array([coef_set[1]-coef_set[0],
                              coef_set[2]-coef_set[1], coef_set[0]-coef_set[2]])

    intercept_boundery = np.array([intercept_set[1]-intercept_set[0],
                                   intercept_set[2]-intercept_set[1],␣
↪intercept_set[0]-intercept_set[2]])


    bound_center = np.dot(np.linalg.inv(coef_boundary[0:2,0:
↪2]),intercept_boundery[0:2])
    #print(bound_center)


    for i in range(3):
        ␣
↪print(f"{coef_boundary[i][0]}x+{coef_boundary[i][1]}y+{intercept_boundery[i]}=0")
    colors = ["blue", "red", "yellow"]
    def plot_hyperplane(c, color):
        def line(x0):
            return (-(x0 * coef_boundary[c][0]) - intercept_boundery[c]) /␣
↪coef_boundary[c][1]
        xc = -bound_center[0]
        xmin,xmax = -10,10

        if coef_boundary[c][1] <= 0:
            xmin,xmax = xc,xmax
        else:
            xmin,xmax = xmin,xc


        plt.plot([xmin, xmax], [line(xmin), line(xmax)], ls="--", color=color)

    for i, color in zip(clf.classes_, colors):
        plot_hyperplane(i, color)
        pass
    plt.axis([-10,10,-10,10])
    plt.show()
```

```
training score : 0.995 (multinomial)
4.428219400944098x+2.2868373828145003y+4.595700908102701=0
```

4.695770842558826x+0.1077741252092068 2y+-4.095153678178038=0
-9.123990243502924x+-2.3946115080237074y+-0.5005472299246638=0



training score : 0.976 (ovr)
4.0481891566447405x+3.870991255055096y+2.696246523243621=0
5.070736974871446x+-1.458886483500355y+-2.2581699388845773=0
-9.118926131516186x+-2.412104771554741y+-0.4380765843590435=0


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

[ ]: