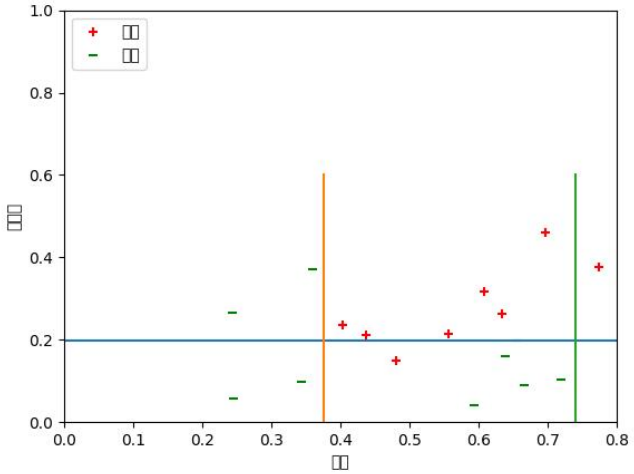
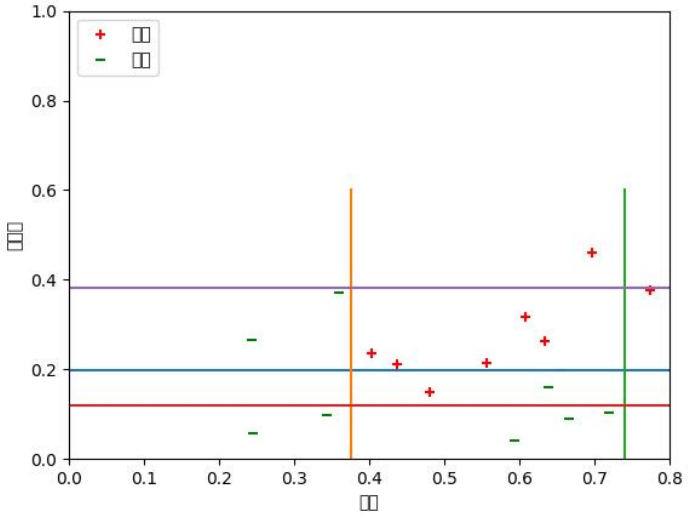
8.3 从网上下载或自己编程实现 AdaBoost, 以不剪枝决策树为基学习器, 在西瓜数据集 3.0α 上训练一个 AdaBoost 集成, 并与图 8.4 进行比较.

集成学习器（字典）： G3 = {0: {'alpha': 0.7702225204735745, 'stump': {'feature': 1, 'threshVal': 0.19875, 'inequal': 'it', 'err': 0.1764705882352941}}, 1: {'alpha': 0.7630281517475247, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'it', 'err': 0.17857142857142855}}, 2: {'alpha': 0.5988515956561702, 'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'it', 'err': 0.23188405797101452}}}
准确率= 0.9411764705882353

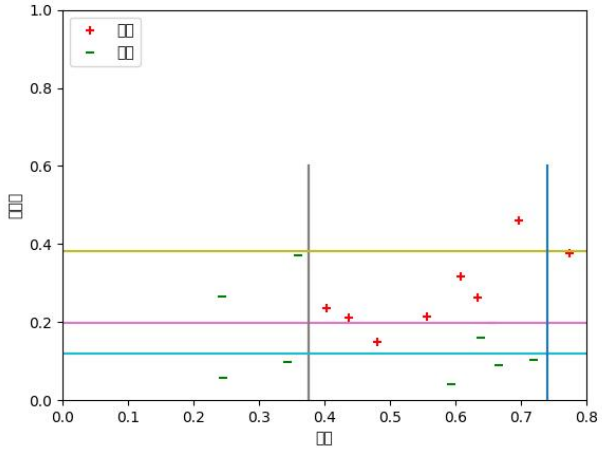


集成学习器（字典）： G5 = {0: {'alpha': 0.7702225204735745, 'stump': {'feature': 1, 'threshVal': 0.19875, 'inequal': 'it', 'err': 0.1764705882352941}}, 1: {'alpha': 0.7630281517475247, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'it', 'err': 0.17857142857142855}}, 2: {'alpha': 0.5988515956561702, 'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'it', 'err': 0.23188405797101452}}, 3: {'alpha': 0.517116813427337, 'stump': {'feature': 1, 'threshVal': 0.12037500000000001, 'inequal': 'it', 'err': 0.2622641509433963}}, 4: {'alpha': 0.38449883125155576, 'stump': {'feature': 1, 'threshVal': 0.381625, 'inequal': 'it', 'err': 0.31669597186700765}}}
准确率= 0.9411764705882353



集成学习器（字典）： G11 = {0: {'alpha': 0.7702225204735745, 'stump': {'feature': 1, 'threshVal': 0.19875, 'inequal': 'it', 'err': 0.1764705882352941}}, 1: {'alpha': 0.7630281517475247, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'it', 'err': 0.17857142857142855}}, 2: {'alpha': 0.5988515956561702, 'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'it', 'err': 0.23188405797101452}}, 3: {'alpha': 0.517116813427337, 'stump': {'feature': 1, 'threshVal': 0.12037500000000001, 'inequal': 'it', 'err': 0.2622641509433963}}, 4: {'alpha': 0.38449883125155576, 'stump': {'feature': 1, 'threshVal': 0.381625, 'inequal': 'it', 'err': 0.31669597186700765}}, 5: {'alpha': 0.47604085356392073, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'it', 'err': 0.2784663666224749}}, 6: {'alpha': 0.44797704534424615, 'stump': {'feature': 1, 'threshVal': 0.19875, 'inequal': 'it', 'err': 0.28988264206782904}}, 7: {'alpha': 0.3050103325106539, 'stump': {'feature': 0, 'threshVal': 0.37575000000000003, 'inequal': 'it', 'err': 0.352054483963029}}, 8: {'alpha': 0.38606192373470366, 'stump': {'feature': 1, 'threshVal': 0.381625, 'inequal': 'it', 'err': 0.31601985458121185}}, 9: {'alpha': 0.28200872281514566, 'stump': {'feature': 1, 'threshVal': 0.12037500000000001, 'inequal': 'it', 'err': 0.36261841086467406}}, 10: {'alpha': 0.22329255621950722, 'stump': {'feature': 0, 'threshVal': 0.7408125000000001, 'inequal': 'it', 'err': 0.39017299225192476}}}

准确率= 0.9411764705882353



代码：

```
import base64
import matplotlib
import os
import sys

from matplotlib._pylab_helpers import Gcf
from matplotlib.backend_bases import FigureManagerBase, ShowBase
from matplotlib.backends.backend_agg import FigureCanvasAgg
from matplotlib.figure import Figure

from datalore.display import debug, display, SHOW_DEBUG_INFO

PY3 = sys.version_info[0] >= 3

index = int(os.getenv("PYCHARM_MATPLOTLIB_INDEX", 0))

rcParams = matplotlib.rcParams


class Show(ShowBase):
    def __call__(self, **kwargs):
        debug("show() called with args %s" % kwargs)
        managers = Gcf.get_all_fig_managers()
        if not managers:
            debug("Error: Managers list in `Gcf.get_all_fig_managers()` is empty")
            return

        for manager in managers:
            manager.show(**kwargs)
```

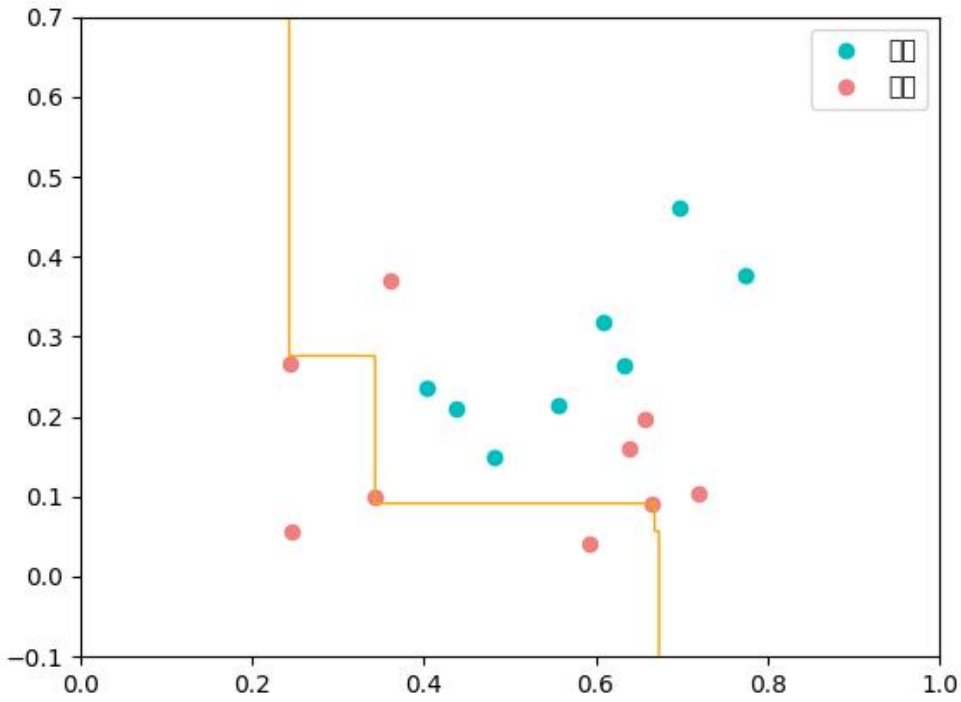
```
        def mainloop(self):
            pass


show = Show()


# from pyplot API
def draw_if_interactive():
        if matplotlib.is_interactive():
            figManager = Gcf.get_active()
            if figManager is not None:
                figManager.canvas.show()
            else:
                debug("Error: Figure manager `Gcf.get_active()` is None")


# from pyplot API
def new_figure_manager(num, *args, **kwargs):
        FigureClass = kwargs.pop('FigureClass', Figure)
        figure = FigureClass(*args, **kwargs)
        return new_figure_manager_given_figure(num, figure)


# from pyplot API
def new_figure_manager_given_figure(num, figure):
        canvas = FigureCanvasInterAgg(figure)
        manager = FigureManagerInterAgg(canvas, num)
        return manager


# from pyplot API
class FigureCanvasInterAgg(FigureCanvasAgg):
        def __init__(self, figure):
            FigureCanvasAgg.__init__(self, figure)

        def show(self):
            FigureCanvasAgg.draw(self)

            if matplotlib.__version__ < '1.2':
                buffer = self.tostring_rgb(0, 0)
            else:
                buffer = self.tostring_rgb()
```

```
            if len(set(buffer)) <= 1:
                # do not plot empty
                debug("Error: Buffer FigureCanvasAgg.tostring_rgb() is empty")
                return

            render = self.get_renderer()
            width = int(render.width)
            debug("Image width: %d" % width)

            is_interactive = os.getenv("PYCHARM_MATPLOTLIB_INTERACTIVE", False)
            if is_interactive:
                debug("Using interactive mode (Run with Python Console)")
                debug("Plot index = %d" % index)
            else:
                debug("Using non-interactive mode (Run without Python Console)")
            plot_index = index if is_interactive else -1
            display(DisplayDataObject(plot_index, width, buffer))

        def draw(self):
            FigureCanvasAgg.draw(self)
            is_interactive = os.getenv("PYCHARM_MATPLOTLIB_INTERACTIVE", False)
            if is_interactive and matplotlib.is_interactive():
                self.show()
            else:
                debug("Error: calling draw() in non-interactive mode won't show a plot. Try
to 'Run with Python Console'")


    class FigureManagerInterAgg(FigureManagerBase):
        def __init__(self, canvas, num):
            FigureManagerBase.__init__(self, canvas, num)
            global index
            index += 1
            self.canvas = canvas
            self._num = num
            self._shown = False

        def show(self, **kwargs):
            self.canvas.show()
            Gcf.destroy(self._num)


    class DisplayDataObject:
        def __init__(self, plot_index, width, image_bytes):
```

```
        self.plot_index = plot_index
        self.image_width = width
        self.image_bytes = image_bytes

    def _repr_display_(self):
        image_bytes_base64 = base64.b64encode(self.image_bytes)
        if PY3:
            image_bytes_base64 = image_bytes_base64.decode()
        body = {
            'plot_index': self.plot_index,
            'image_width': self.image_width,
            'image_base64': image_bytes_base64
        }
        return ('pycharm-plot-image', body)
```

8.5 试编程实现 Bagging, 以决策树桩为基学习器, 在西瓜数据集 3.0α 上训练一个 Bagging 集成, 并与图 8.6 进行比较.

准确率 0.7647058823529411




```
from matplotlib import pyplot as plt
from sklearn.utils import resample
```

```
def stumpClassify(X, dim, thresh_val, thresh_inequal):
    ret_array = np.ones((X.shape[0], 1))

    if thresh_inequal == 'lt':
        ret_array[X[:, dim] <= thresh_val] = -1
    else:
        ret_array[X[:, dim] > thresh_val] = -1

    return ret_array

#建立树桩
def buildStump(X, y):
    m, n = X.shape
    best_stump = {}
    min_error = 1
    for dim in range(n):
        x_min = np.min(X[:, dim])
        x_max = np.max(X[:, dim])
        # 这里第一次尝试使用排序后的点作为分割点，效果很差，因为那样会错过一些更好的分割点；
        # 所以后来切割点改成将最大值和最小值之间分割成 20 等份。
        # sorted_x = np.sort(X[:, dim])
        # split_points = [(sorted_x[i] + sorted_x[i + 1]) / 2 for i in range(m - 1)]
        split_points = [(float(x_max) - float(x_min)) / 20 * i + x_min for i in range(20)]
        for inequal in ['lt', 'gt']:
            for thresh_val in split_points:
                ret_array = stumpClassify(X, dim, thresh_val, inequal)
                error = np.mean(ret_array != y)
                if error < min_error:
                    best_stump['dim'] = dim
                    best_stump['thresh'] = thresh_val
                    best_stump['inequal'] = inequal
                    best_stump['error'] = error
                    min_error = error
    return best_stump

def stumpBagging(X, y, nums=20):
    stumps = []
    seed = 16
    for _ in range(nums):
        X_, y_ = resample(X, y, random_state=seed)  # sklearn 中自带的实现自助采样的方法
        seed += 1
        stumps.append(buildStump(X_, y_))
```

```
    return stumps

def stumpPredict(X, stumps):
    ret_arrays = np.ones((X.shape[0], len(stumps)))

    for i, stump in enumerate(stumps):
        ret_arrays[:, [i]] = stumpClassify(X, stump['dim'], stump['thresh'], stump['inequal'])

    return np.sign(np.sum(ret_arrays, axis=1))

#可视化
def pltStumpBaggingDecisionBound(X_, y_, stumps):
    pos = y_ == 1
    neg = y_ == -1
    x_tmp = np.linspace(0, 1, 600)
    y_tmp = np.linspace(-0.1, 0.7, 600)

    X_tmp, Y_tmp = np.meshgrid(x_tmp, y_tmp)
    Z_ = stumpPredict(np.c_[X_tmp.ravel(), Y_tmp.ravel()], stumps).reshape(X_tmp.shape)

    plt.contour(X_tmp, Y_tmp, Z_, [0], colors='orange', linewidths=1)

    plt.scatter(X_[pos, 0], X_[pos, 1], label='好瓜', color='c')
    plt.scatter(X_[neg, 0], X_[neg, 1], label='坏瓜', color='lightcoral')
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
   # plt.legend(loc='upper left')
    plt.legend()
    plt.show()


if __name__ == "__main__":
    import numpy as np

    #     import pandas as pd
    #     data_path = 'F:\\python\\dataset\\chapter8\\watermelon3_0a_Ch.txt'

    #     data = pd.read_table(data_path, delimiter=' ')

    #     X = data.iloc[:, :2].values
    #     y = data.iloc[:, 2].values
    #     XX = np.array(X)
    #     yy = np.array(y)
```

```
XX = [[0.697, 0.460],
        [0.774, 0.376],
        [0.634, 0.264],
        [0.608, 0.318],
        [0.556, 0.215],
        [0.403, 0.237],
        [0.481, 0.149],
        [0.437, 0.211],
        [0.666, 0.091],
        [0.243, 0.267],
        [0.245, 0.057],
        [0.343, 0.099],
        [0.639, 0.161],
        [0.657, 0.198],
        [0.360, 0.370],
        [0.593, 0.042],
        [0.719, 0.103]
        ]
yy = [1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
X = np.array(XX)
y = np.array(yy)
stumps = stumpBagging(X, y, 21)

print(np.mean(stumpPredict(X, stumps) == y))
pltStumpBaggingDecisionBound(X, y, stumps)
```