НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Анализ данных»

Домашнее задание на тему:
«Лабораторная работа №5»

Выполнил: Осипов Лев,
студент группы 301ПИ (1).

Москва, 2015 г.

СОДЕРЖАНИЕ

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Ответ – синяя кривая. Во время кризиса банку важнее кредитовать как можно меньше ненадежных заемщиков. Так как FRP отвечает за вероятность того, что ненадежного определили как надежного (ошибка второго рода), при низком значении FRP нужно высокое значение TPR, которое отвечает за вероятность того, что надежного определили как ненадежного (ошибка первого рода). Именно синий график демонстрирует такую зависимость.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Для решения задания была написана программа, исследующая решающих правил для линейного ядра.

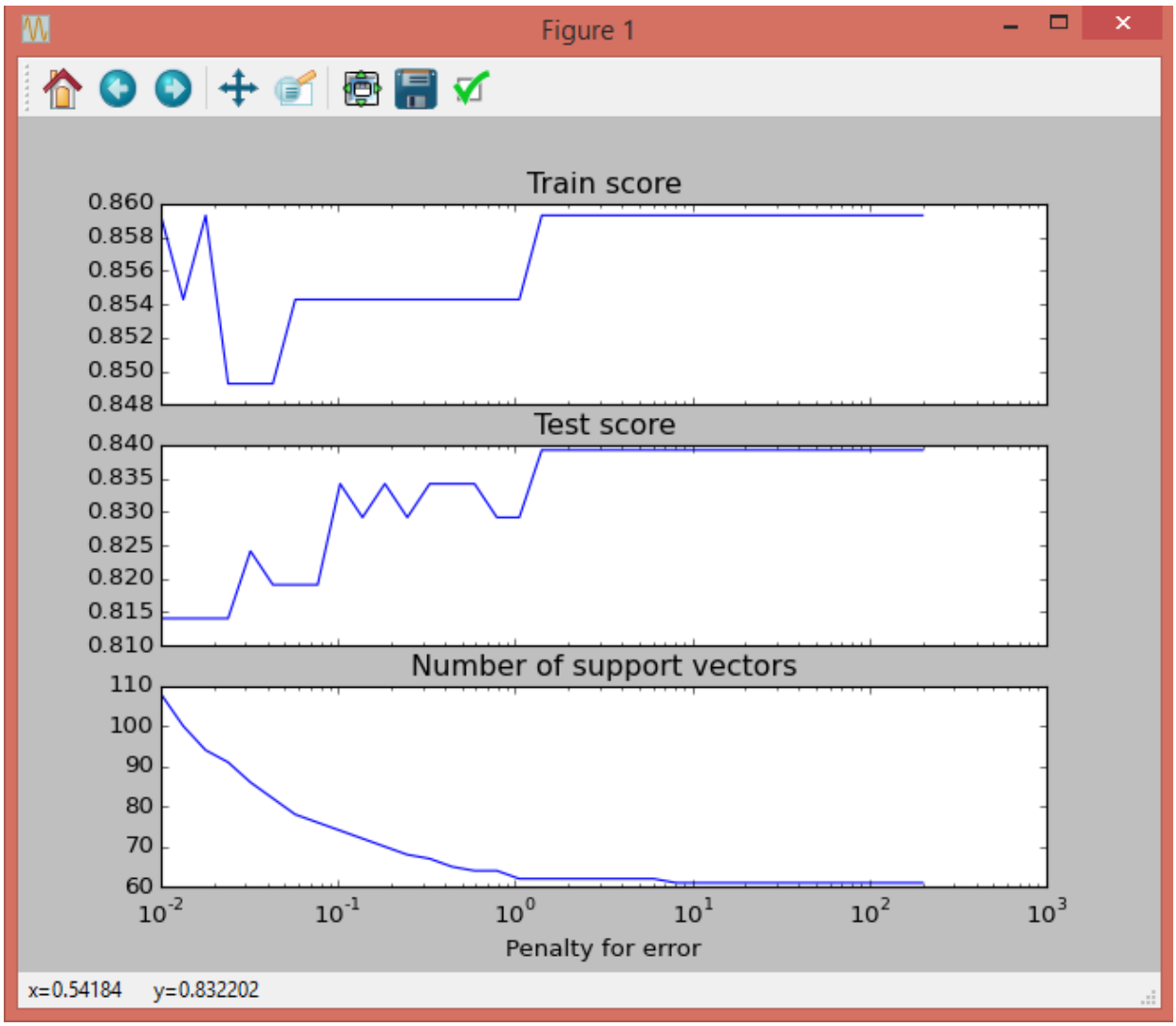

Рис. 1. Зависимость верно классифицированных объектов на обучающей и тестовой выборках, а также количество опорных векторов от C

Использовать точность возможно.

Без доступа к тестовой выборке можно воспользоваться число опорных векторов.

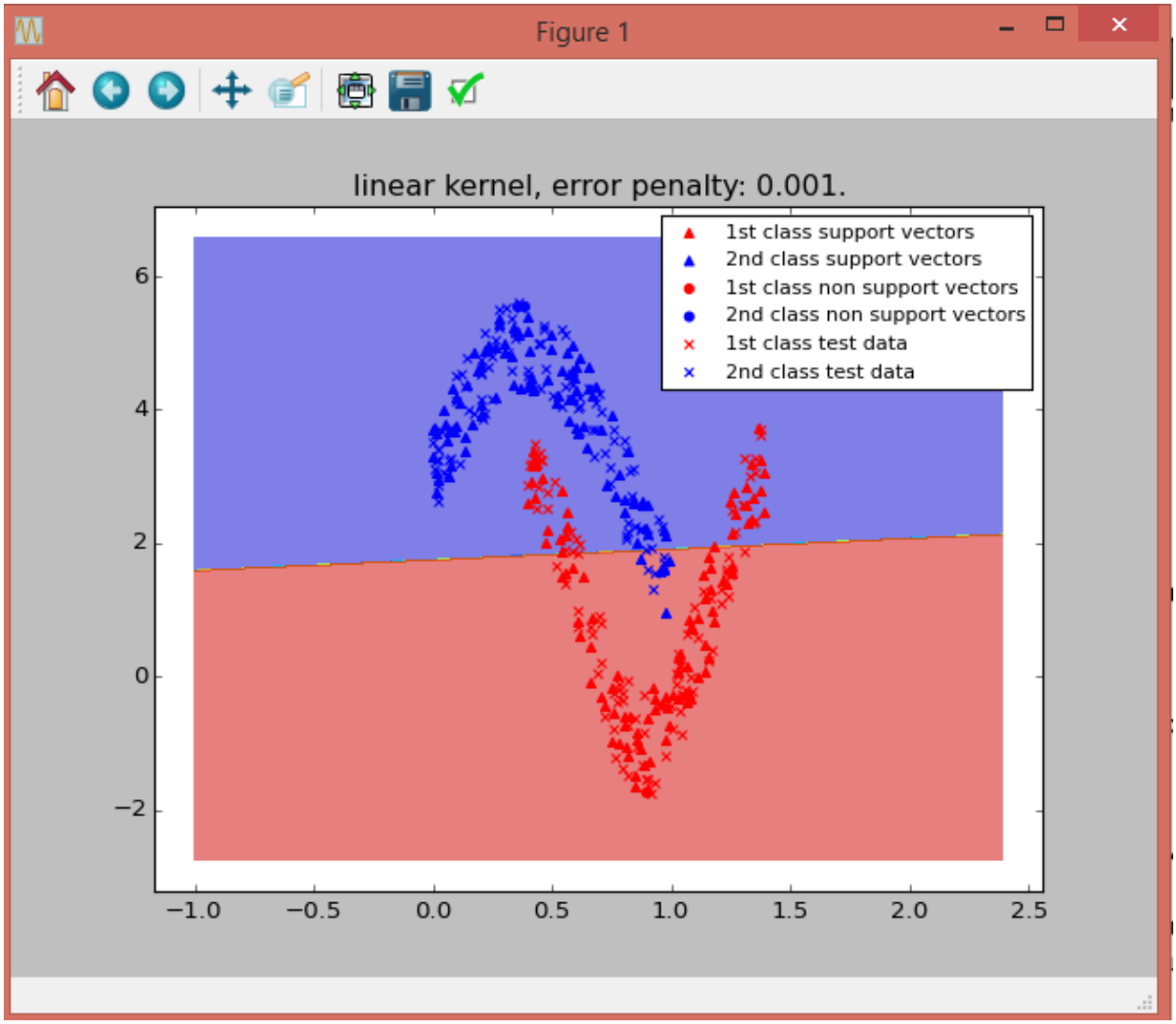

Рис. 2. Оптимальное C

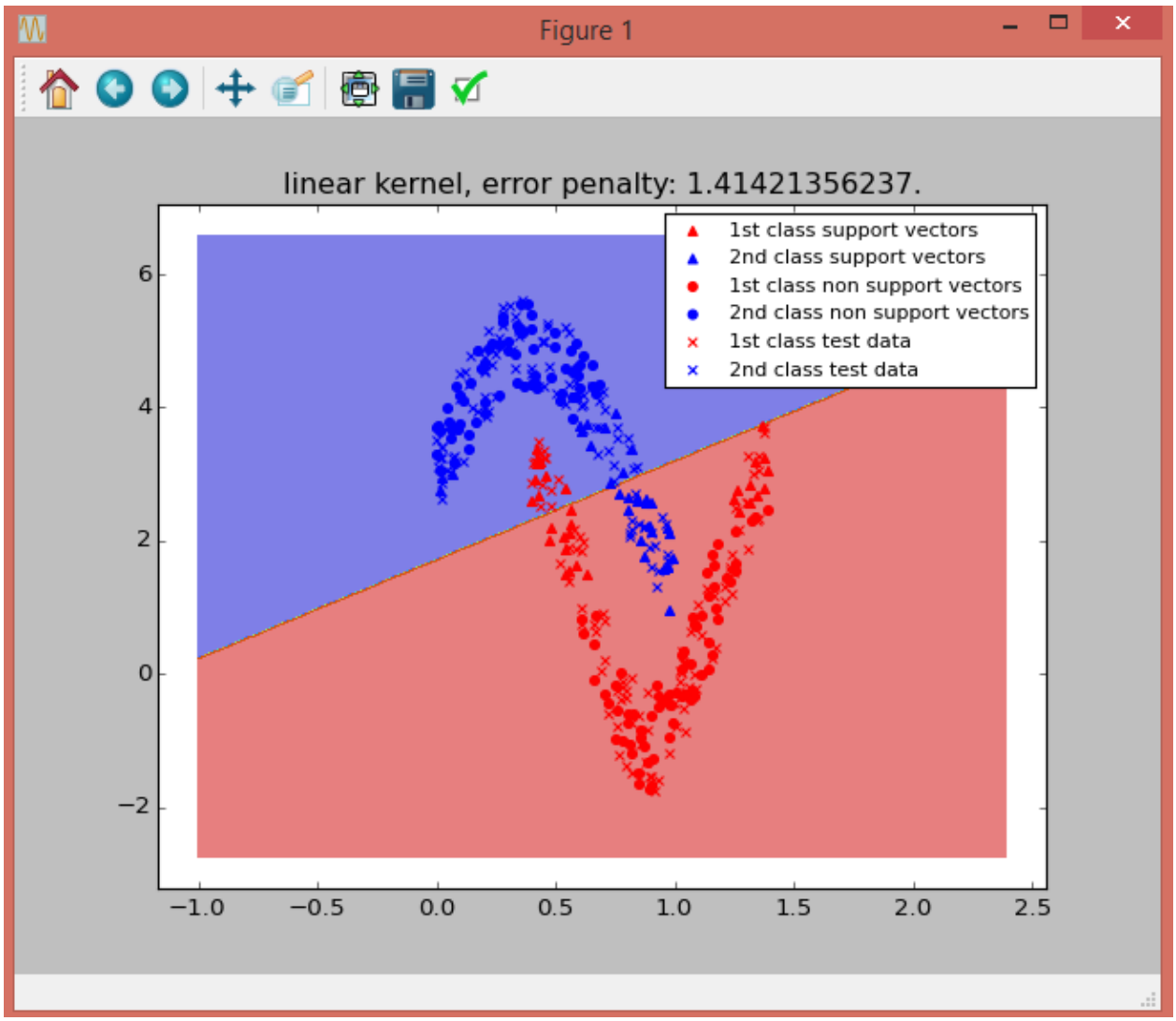

Рис. 1. Неадекватно маленькое C

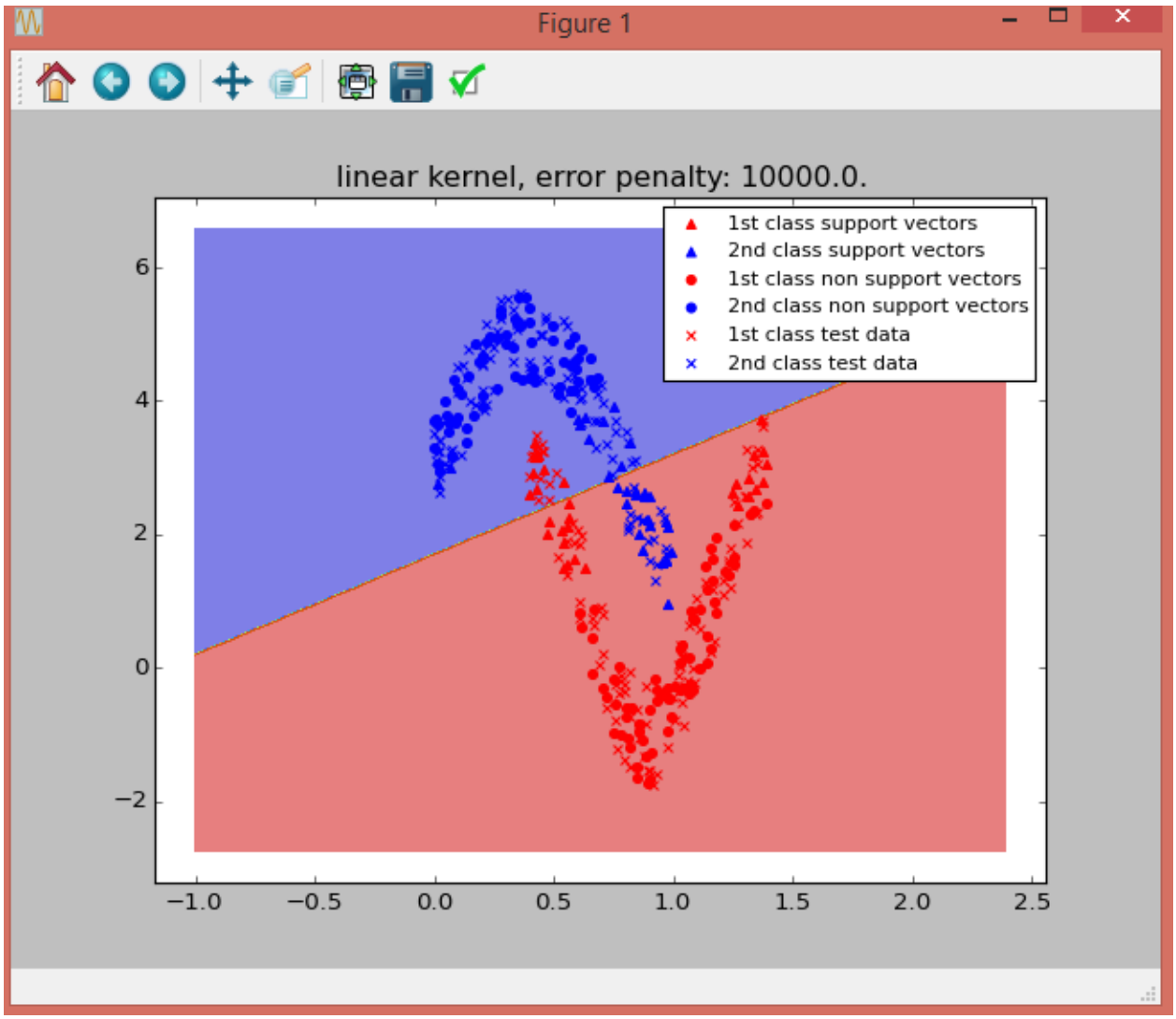

Рис. 1. Неадекватно большое C

СПИСОК ЛИТЕРАТУРЫ

1) **Анализ данных (Программная инженерия)** – http://wiki.cs.hse.ru/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_%28%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%8F%29#.D0.9E.D1.84.D0.BE.D1.80.D0.BC.D0.BB.D0.B5.D0.BD.D0.B8.D0.B5_.D0.BF.D0.B8.D1.81.D0.B5.D0.BC

ТЕКСТ ПРОГРАММЫ

```
__author__ = 'Lev Osipov'

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from math import log10


def learn_stat(train_data, train_labels, test_data, test_labels,
kernel, C):
    clf = SVC(C=C, kernel=kernel)
    clf.fit(train_data, train_labels)
    train_score = clf.score(train_data, train_labels)
    test_score = clf.score(test_data, test_labels)
    support = clf.support_.size
    return train_score, test_score, support


def plot_stat(train_data, train_labels, test_data, test_labels,
kernel, minC, maxC, steps):

    train_score = np.empty(steps)
    test_score = np.empty(steps)
    support = np.empty(steps)

    c_interval = np.logspace(minC, maxC, steps)
    for i, C in enumerate(c_interval):
        train_score[i], test_score[i], support[i] = \
            learn_stat(train_data, train_labels, test_data,
test_labels, kernel, C)
    f, ax = plt.subplots(3, sharex=True)
    i = 0

    ax[i].plot(c_interval, train_score)
    ax[i].set_title("Train score")
    ax[i].set_xscale('log', basex=10)
    i += 1

    ax[i].plot(c_interval, test_score)
    ax[i].set_title("Test score")
    ax[i].set_xscale('log', basex=10)
    i += 1

    ax[i].plot(c_interval, support)
    ax[i].set_title("Number of support vectors")
    ax[i].set_xlabel("Penalty for error")

    ax[i].set_xscale('log', basex=10)

    plt.show()
```

```
def plot_support(train_data, train_labels, test_data,
test_labels, kernel, C):

    clf = SVC(C=C, kernel=kernel)
    clf.fit(train_data, train_labels)

    support_i = clf.support_

    train_sup_vec = train_data[support_i]
    train_sup_lab = train_labels[support_i]

    train_non_sup_vec = np.delete(train_data, support_i, axis=0)
    train_non_sup_lab = np.delete(train_labels, support_i,
axis=0)

    train_sup_vec_1 = train_sup_vec[np.where(train_sup_lab ==
1)]
    train_sup_vec_2 = train_sup_vec[np.where(train_sup_lab !=
1)]

    train_non_sup_vec_1 =
train_non_sup_vec[np.where(train_non_sup_lab == 1)]
    train_non_sup_vec_2 =
train_non_sup_vec[np.where(train_non_sup_lab != 1)]

    test_data_1 = test_data[np.where(test_labels == 1)]
    test_data_2 = test_data[np.where(test_labels != 1)]

    all = np.concatenate((test_data, train_data), axis=0)
    h = .01

    x_min, x_max = all[:, 0].min() - 1, all[:, 0].max() + 1
    y_min, y_max = all[:, 1].min() - 1, all[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))

    z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    z = z.reshape(xx.shape)
    plt.contourf(xx, yy, z, alpha=0.5)

    plt.scatter(train_sup_vec_1[:, 0], train_sup_vec_1[:, 1],
color='r', marker='^',
                label='1st class support vectors')
    plt.scatter(train_sup_vec_2[:, 0], train_sup_vec_2[:, 1],
color='b', marker='^',
                label='2nd class support vectors')

    plt.scatter(train_non_sup_vec_1[:, 0],
train_non_sup_vec_1[:, 1], color='r', marker='o',
                label='1st class non support vectors')
    plt.scatter(train_non_sup_vec_2[:, 0],
```

```
train_non_sup_vec_2[:, 1], color='b', marker='o',
                label='2nd class non support vectors')

    plt.scatter(test_data_1[:, 0], test_data_1[:, 1], color='r',
marker='x', label='1st class test data')
    plt.scatter(test_data_2[:, 0], test_data_2[:, 1], color='b',
marker='x', label='2nd class test data')

    plt.legend(scatterpoints=1, fontsize=10)
    plt.title("{0} kernel, error penalty: {1}.".format(kernel,
C))

    plt.show()
    plt.clf()


def find_optimal_c(train_data, train_labels, test_data,
test_labels, kernel, minC, maxC, steps):
    c_interval = np.logspace(minC, maxC, steps)
    max_score = 0
    c = minC
    for i, C in enumerate(c_interval):
        test_score = learn_stat(train_data, train_labels,
test_data, test_labels, kernel, C)[1]
        if test_score > max_score:
            max_score = test_score
            c = C
    return c

# Task 1
tr_data = pd.read_csv('synth_train.csv').as_matrix()
tr_labels = tr_data[:, 0]
tr_data = np.delete(tr_data, 0, axis=1)

te_data = pd.read_csv('synth_test.csv').as_matrix()
te_labels = te_data[:, 0]
te_data = np.delete(te_data, 0, axis=1)

# Task 2
minimC = log10(1e-2)
maximC = log10(200)
steps_count = 35
plot_stat(tr_data, tr_labels, te_data, te_labels, 'linear',
minimC, maximC, steps_count)

# Task 3
c = find_optimal_c(tr_data, tr_labels, te_data, te_labels,
'linear', minimC, maximC, steps_count)
plot_support(tr_data, tr_labels, te_data, te_labels, 'linear',
1e-3)
plot_support(tr_data, tr_labels, te_data, te_labels, 'linear',
c)
```

```
plot_support(tr_data, tr_labels, te_data, te_labels, 'linear',
1e4)
```