先端データ解析論
第九回　レポート

37-196360　森田涼介

2019 年 6 月 25 日

宿題 1

ガウスカーネルモデル

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n+n'} \theta_j K(\boldsymbol{x},\ \boldsymbol{x}_j) = \boldsymbol{K}\boldsymbol{\theta} \tag{1}$$

$$K(\boldsymbol{x},\ \boldsymbol{c}) = \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{c}||^2}{2h^2}\right) \tag{2}$$

$$\boldsymbol{K} = \begin{bmatrix} K(\boldsymbol{x}_1,\ \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_{n+n'},\ \boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_{n+n'},\ \boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_{n+n'},\ \boldsymbol{x}_{n+n'}) \end{bmatrix} \tag{3}$$

に対してラプラス正則化最小二乗分類を実装する。

$$\sum_{i,\ i'=1}^{m} W_{i,\ i'}(a_i - a_{i'})^2 = 2\sum_{i,\ i'=1}^{m} L_{i,\ i'} a_i a_{i'} \tag{4}$$

$$\boldsymbol{D} = \mathrm{diag}\left(\sum_{i=1}^{m} W_{1,\ i},\ \cdots,\ \sum_{i=1}^{m} W_{m,\ i}\right) \tag{5}$$

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \tag{6}$$

近傍グラフの重みにはガウスカーネルを用い，

$$W_{i,\ i'} = \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_{i'}||^2}{2h^2}\right) \tag{7}$$

とする。

目的関数は，

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(f_{\boldsymbol{\theta}(\boldsymbol{x}_i)-y_i}\right)^2 + \lambda||\boldsymbol{\theta}||^2 + \nu \sum_{i,\ i'=1}^{n+n'} W_{i,\ i'}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i'}))^2 \tag{8}$$

$$= ||\tilde{\boldsymbol{K}}\boldsymbol{\theta} - \boldsymbol{y}||^2 + \lambda||\boldsymbol{\theta}||^2 + 2\nu\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{K}\boldsymbol{\theta} \tag{9}$$

となる。これを $\boldsymbol{\theta}$ で偏微分すると，

$$\frac{\partial J}{\partial \boldsymbol{\theta}} = 2\tilde{\boldsymbol{K}}^{\mathrm{T}}\tilde{\boldsymbol{K}}\boldsymbol{\theta} - 2\tilde{\boldsymbol{K}}\boldsymbol{y} + 2\lambda\boldsymbol{\theta} + 4\nu\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{K}\boldsymbol{\theta} \tag{10}$$

$$= 2\left(\tilde{\boldsymbol{K}}^{\mathrm{T}}\tilde{\boldsymbol{K}} + \lambda\boldsymbol{I} + 2\nu\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{K}\right)\boldsymbol{\theta} - 2\tilde{\boldsymbol{K}}\boldsymbol{y} \tag{11}$$

$$= \boldsymbol{0} \tag{12}$$

これより，

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \left(\tilde{\boldsymbol{K}}^{\mathrm{T}}\tilde{\boldsymbol{K}} + \lambda\boldsymbol{I} + 2\nu\boldsymbol{K}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{K}\right)^{-1}\tilde{\boldsymbol{K}}\boldsymbol{y} \tag{13}$$

である。

$h = 1.0,\ \lambda = 1.0,\ \nu = 1.0$ としてこれを実装したものが 8 ページの Listing 1 である。

結果は図 1 に示した通りである。教師ありのデータは各クラス 1 つずつしかないが，ラベルなしのデータについても上手く分けられている。

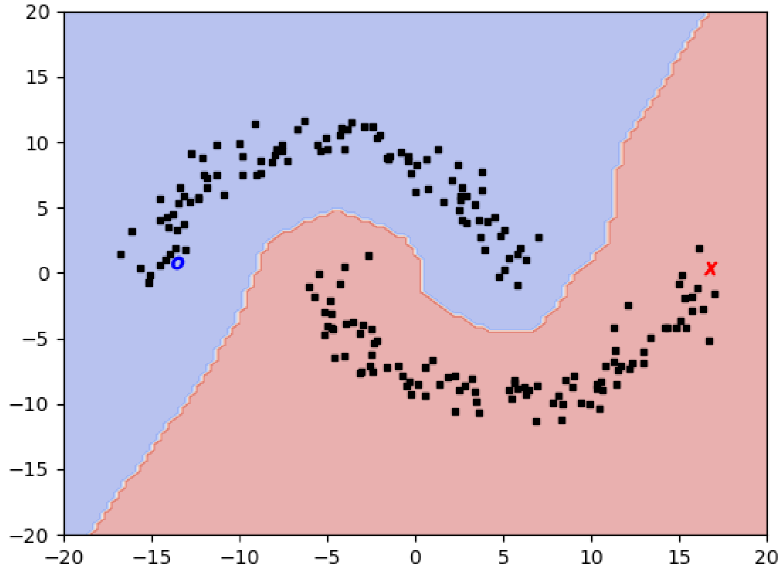

図 1: 結果

宿題 2

訓練標本の分布を p_{train}，テスト標本の分布を p_{test}，テスト標本における各クラスの訓練標本の分布の混合比を π，混合分布を q_π とする。二値分類を考えると，

$$q_\pi = \pi p_{\mathrm{train}}(\boldsymbol{x}|y=+1) + (1-\pi) p_{\mathrm{train}}(\boldsymbol{x}|y=-1) \tag{14}$$

となる。いま，エネルギー距離の二乗は，

$$D_{\mathrm{E}}^2(p_{\mathrm{test}},\ q_\pi) = 2\mathbb{E}_{\boldsymbol{x}'\sim p_{\mathrm{test}},\ \boldsymbol{x}\sim q_\pi}\left[||\boldsymbol{x}'-\boldsymbol{x}||\right] - \mathbb{E}_{\boldsymbol{x}',\ \tilde{\boldsymbol{x}}'\sim p_{\mathrm{test}}}\left[||\boldsymbol{x}'-\tilde{\boldsymbol{x}}'||\right] - \mathbb{E}_{\boldsymbol{x},\ \tilde{\boldsymbol{x}}\sim q_\pi}[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||] \tag{15}$$

である。ここで，

$$A_{y,\tilde{y}} = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{train}}(\boldsymbol{x}|y),\ \tilde{\boldsymbol{x}}\sim p_{\mathrm{train}}(\boldsymbol{x}|\tilde{y})}[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||] \tag{16}$$

$$b_y = \mathbb{E}_{\boldsymbol{x}'\sim p_{\mathrm{test}},\ \boldsymbol{x}\sim p_{\mathrm{train}}(\boldsymbol{x}|y)}\left[||\boldsymbol{x}'-\boldsymbol{x}||\right] \tag{17}$$

とおく。式 (15) の第一項は，

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}'\sim p_{\mathrm{test}},\ \boldsymbol{x}\sim q_\pi}\left[||\boldsymbol{x}'-\boldsymbol{x}||\right] &= \mathbb{E}_{\boldsymbol{x}'\sim p_{\mathrm{test}}}\left[\mathbb{E}_{\boldsymbol{x}\sim q_\pi}\left[||\boldsymbol{x}'-\boldsymbol{x}||\right]\right] \\
&= \mathbb{E}_{\boldsymbol{x}'\sim p_{\mathrm{test}}}\left[\pi\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{train}}(\boldsymbol{x}|+1)}\left[||\boldsymbol{x}'-\boldsymbol{x}||\right] + (1-\pi)\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{train}}(\boldsymbol{x}|-1)}\left[||\boldsymbol{x}'-\boldsymbol{x}||\right]\right] \\
&= \pi\mathbb{E}_{\boldsymbol{x}'\sim p_{\mathrm{test}},\ \boldsymbol{x}\sim p_{\mathrm{train}}(\boldsymbol{x}|+1)}\left[||\boldsymbol{x}'-\boldsymbol{x}||\right] + (1-\pi)\mathbb{E}_{\boldsymbol{x}'\sim p_{\mathrm{test}},\ \boldsymbol{x}\sim p_{\mathrm{train}}(\boldsymbol{x}|-1)}\left[||\boldsymbol{x}'-\boldsymbol{x}||\right] \\
&= \pi b_{+1} + (1-\pi) b_{-1}
\end{aligned}$$

よって，

$$\mathbb{E}_{\boldsymbol{x}'\sim p_{\mathrm{test}},\ \boldsymbol{x}\sim q_\pi}\left[||\boldsymbol{x}'-\boldsymbol{x}||\right] = (b_{+1}-b_{-1})\pi + b_{-1} \tag{18}$$

第二項は，

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x},\ \tilde{\boldsymbol{x}}\sim q_\pi}[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||] &= \mathbb{E}_{\boldsymbol{x}\sim q_\pi}[\mathbb{E}_{\tilde{\boldsymbol{x}}\sim q_\pi}[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||]] \\
&= \mathbb{E}_{\boldsymbol{x}\sim q_\pi}\left[\pi\mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{\mathrm{train}}(\boldsymbol{x}|+1)}[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||] + (1-\pi)\mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{\mathrm{train}}(\boldsymbol{x}|-1)[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||]}\right] \\
&= \pi\left\{\pi\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{train}}(\boldsymbol{x}|+1),\ \tilde{\boldsymbol{x}}\sim p_{\mathrm{train}}(\tilde{\boldsymbol{x}}|+1)}[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||] + (1-\pi)\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{train}}(\boldsymbol{x}|-1),\ \tilde{\boldsymbol{x}}\sim p_{\mathrm{train}}(\tilde{\boldsymbol{x}}|+1)}[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||]\right\} \\
&\quad + (1-\pi)\left\{\pi\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{train}}(\boldsymbol{x}|+1),\ \tilde{\boldsymbol{x}}\sim p_{\mathrm{train}}(\tilde{\boldsymbol{x}}|-1)}[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||] + (1-\pi)\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{train}}(\boldsymbol{x}|-1),\ \tilde{\boldsymbol{x}}\sim p_{\mathrm{train}}(\tilde{\boldsymbol{x}}|-1)}[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||]\right\} \\
&= \pi^2 A_{+1,+1} + \pi(1-\pi)A_{-1,+1} + (1-\pi)\pi A_{+1,-1} + (1-\pi)^2 A_{-1,-1} \\
&= (A_{+1,+1} - A_{-1,+1} - A_{+1,-1} + A_{-1,-1})\pi^2 + (A_{-1,+1} + A_{+1,-1} - 2A_{-1,-1})\pi + A_{-1,-1}
\end{aligned}$$

ここで，$A_{+1,-1} = A_{-1,+1}$ から，

$$\mathbb{E}_{\boldsymbol{x},\ \tilde{\boldsymbol{x}}\sim q_\pi}[||\boldsymbol{x}-\tilde{\boldsymbol{x}}||] = (-2A_{+1,-1} + A_{+1,+1} + A_{-1,-1})\pi^2 + 2(A_{+1,-1} - A_{-1,-1})\pi + A_{-1,-1} \tag{19}$$

式 (15) に式 (18)，(19) を代入して，

$$\begin{aligned}
J(\pi) &= 2\{(b_{+1}-b_{-1})\pi + b_{-1}\} - \mathbb{E}_{\boldsymbol{x}',\ \tilde{\boldsymbol{x}}'\sim p_{\mathrm{test}}}\left[||\boldsymbol{x}'-\tilde{\boldsymbol{x}}'||\right] \\
&\quad - \left\{(-2A_{+1,-1} + A_{+1,+1} + A_{-1,-1})\pi^2 + 2(A_{+1,-1} - A_{-1,-1})\pi + A_{-1,-1}\right\} \\
&= (2A_{+1,-1} - A_{+1,+1} - A_{-1,-1})\pi^2 - 2(A_{+1,-1} - A_{-1,-1} - b_{+1} + b_{-1})\pi \\
&\quad + \left\{2b_{-1} - A_{-1,-1} - \mathbb{E}_{\boldsymbol{x}',\ \tilde{\boldsymbol{x}}'\sim p_{\mathrm{test}}}\left[||\boldsymbol{x}'-\tilde{\boldsymbol{x}}'||\right]\right\}
\end{aligned}$$

となる。

宿題 3

線形モデル

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x} + \theta_0 \tag{20}$$

に対し，クラス比重み付き最小二乗法を実装する。

クラス比の推定値は，

$$\tilde{\pi} = \frac{\hat{A}_{+1,-1} - \hat{A}_{-1,-1} - \hat{b}_{+1} + \hat{b}_{-1}}{2\hat{A}_{+1,-1} - \hat{A}_{+1,+1} - \hat{A}_{-1,-1}} \tag{21}$$

$$\hat{\pi} = \min(1,\ \max(0,\ \tilde{\pi})) \tag{22}$$

ここで，

$$\hat{A}_{y,\ \tilde{y}} = \frac{1}{n_y n_{\tilde{y}}} \sum_{i:y_i=y} \sum_{\tilde{i}:y_{\tilde{i}}=\tilde{y}} ||\boldsymbol{x}_i - \boldsymbol{x}_{\tilde{i}}|| \tag{23}$$

$$\hat{b}_y = \frac{1}{n' n_y} \sum_{i'=1}^{n'} \sum_{i:y_i=y} ||\boldsymbol{x}'_{i'} - \boldsymbol{x}_i|| \tag{24}$$

である。

いま，

$$\pi_i = \frac{p_{\mathrm{test}}(y_i)}{p_{\mathrm{train}}(y_i)} = \begin{cases} \hat{\pi} & (y = +1) \\ 1 - \hat{\pi} & (y = -1) \end{cases} \tag{25}$$

$$\Pi = \mathrm{diag}(\pi_1,\ \cdots,\ \pi_n) \tag{26}$$

とし，また，

$$\Phi = \begin{bmatrix} 1 & x_{1,0} & x_{1,1} \\ & \vdots & \\ 1 & x_{n,0} & x_{n,1} \end{bmatrix} \tag{27}$$

$$\Theta = (\theta_0,\ \boldsymbol{\theta}^{\mathrm{T}})^{\mathrm{T}} \tag{28}$$

とすると，目的関数は，

$$J(\Theta) = \sum_{i=1}^{n} \frac{p_{\mathrm{test}}(y_i)}{p_{\mathrm{train}}(y_i)} (f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i)^2 \tag{29}$$

$$= (\Theta\Phi - \boldsymbol{y})^{\mathrm{T}}\Pi(\Theta\Phi - \boldsymbol{y}) \tag{30}$$

となり，この Θ による偏微分が $\mathbf{0}$ となることから，

$$\frac{\partial J}{\partial \Theta} = 2\Phi^{\mathrm{T}}\Pi\Phi\Theta - 2\boldsymbol{y}^{\mathrm{T}}\Phi^{\mathrm{T}}\Pi\Theta = \mathbf{0} \tag{31}$$

$$\hat{\Theta} = \left(\Phi^{\mathrm{T}}\Pi\Phi\right)^{-1}\Phi^{\mathrm{T}}\Pi\boldsymbol{y} \tag{32}$$

を得る。

これを実装すると 12 ページの Listing 2 のようになる。重み付きの結果は，訓練データに対するものを図 2 に，テストデータに対するものを図 3 に示した。また，重み付けなしについても実験を行い，その結果は，訓練データに対するものを図 4 に，テストデータに対するものを図 5 に示した。図を見ると，重み付けありの方が汎化性能が高くなっていることがわかる。

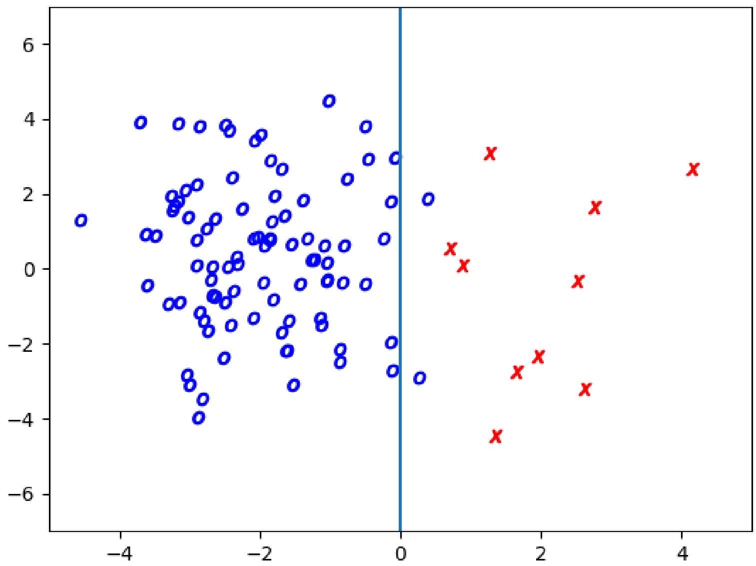

図 2: 重み付き，訓練データに対する結果

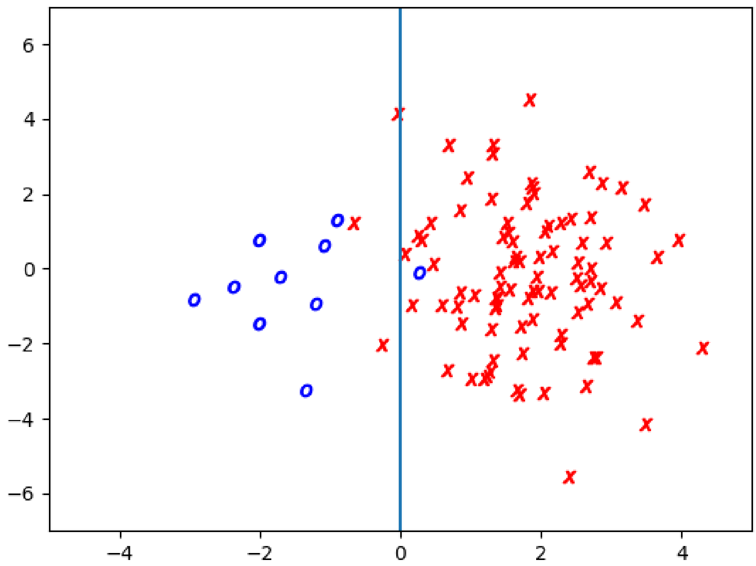

図 3: 重み付き，テストデータに対する結果

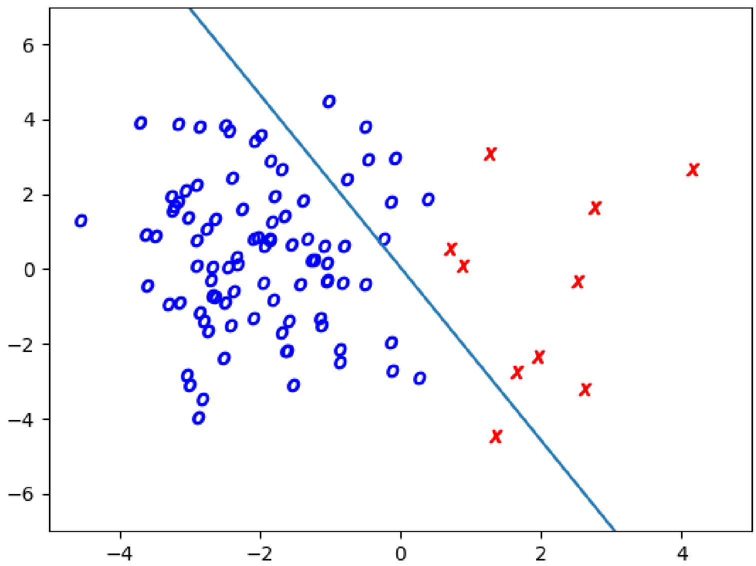

図 4: 重み付けなし，訓練データに対する結果

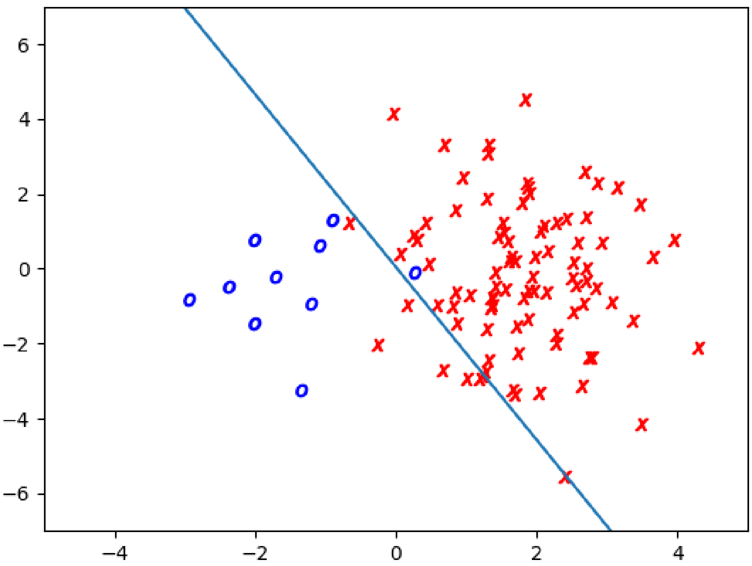

図 5: 重み付けなし，テストデータに対する結果

プログラム

実行環境と用いた言語・ライブラリを以下の表 1 に示す。

表 1: プログラムの実行環境

OS	:	Microsoft Windows 10 Pro (64bit)
CPU	:	Intel(R) Core(TM) i5-4300U
RAM	:	4.00 GB
使用言語	:	Python3.6
可視化	:	matplotlib ライブラリ

Listings 1: assignment1.py

```python
# -*- coding: utf-8 -*-


import numpy as np
import matplotlib.pyplot as plt


def generate_data(n=200):
    x = np.linspace(0, np.pi, n // 2)
    u = np.stack([np.cos(x) + .5, -np.sin(x)], axis=1) * 10.
    u += np.random.normal(size=u.shape)
    v = np.stack([np.cos(x) - .5, np.sin(x)], axis=1) * 10.
    v += np.random.normal(size=v.shape)
    x = np.concatenate([u, v], axis=0)
    y = np.zeros(n)
    y[0] = 1
    y[-1] = -1
    return x, y


def calc_norm(x, c, save_memory=False):
    if save_memory:
        n_x = x.shape[1]
        n_c = c.shape[0]
        d = x.shape[-1]
        norm = np.zeros((n_c, n_x))
        x = np.reshape(x, (n_x, d))
        c = np.reshape(c, (n_c, d))
        for i in range(len(x)):
            x_i = x[i]
            norm[i, :] = np.sum((x_i - c)**2, axis=-1)
    else:
```

```
        norm = np.sum((x - c) ** 2, axis=-1)
    return norm


def gauss_kernel(x, c, h, save_memory=False):
    norm = calc_norm(x, c, save_memory)
    ker = np.exp(- norm / (2*h**2))
    return ker


def calc_design_matrix(x, c, h, kernel):
    return kernel(x[None], c[:, None], h)


def lrls(x, y, h=1., l=1., nu=1., kernel=gauss_kernel):
    """

    :param x: data points
    :param y: labels of data points
    :param h: width parameter of the Gaussian kernel
    :param l: weight decay
    :param nu: Laplace regularization
    :return:
    """
    x_tilde = x[y!=0]
    K = calc_design_matrix(x, x, h, kernel)
    K_tilde = calc_design_matrix(x_tilde, x, h, kernel)

    W = K
    D = np.diag(W.sum(axis=1))
    L = D - W

    tmp = K_tilde.dot(K_tilde.T) + l*np.eye(len(K_tilde)) +
    2*nu*K.dot(L).dot(K)
    theta = np.linalg.inv(tmp).dot(K_tilde).dot(y[y!=0])
    return theta


def supervised_train(x, y, h=1., l=1., kernel=gauss_kernel):
    K = calc_design_matrix(x, x, h, kernel)
    tmp = K.dot(K.T) + l*np.eye(len(K))
    theta = np.linalg.inv(tmp).dot(K).dot(y)
    return theta


def visualize(x, y, theta, h=1., grid_size=100, path=None):
    plt.xlim(-20., 20.)
```

```
    plt.ylim(-20., 20.)
    grid = np.linspace(-20., 20., grid_size)

    X, Y = np.meshgrid(grid, grid)
    mesh_grid = np.stack([np.ravel(X), np.ravel(Y)], axis=1)

    k = np.exp(
        -np.sum(
            (x.astype(np.float32)[:, None] -
    mesh_grid.astype(np.float32)[None])**2,
            axis=2).astype(np.float64)
            / (2 * h ** 2)
        )
    plt.contourf(
        X, Y,
        np.reshape(np.sign(k.T.dot(theta)), (grid_size, grid_size)),
        alpha=.4,
        cmap=plt.cm.coolwarm,
        )
    plt.scatter(x[y==0][:, 0], x[y == 0][:, 1], marker='$.$', c='black')
    plt.scatter(x[y==1][:, 0], x[y == 1][:, 1], marker='$X$', c='red')
    plt.scatter(x[y==-1][:, 0], x[y == -1][:, 1], marker='$O$', c='blue')
    if path:
        plt.savefig(path)
    plt.show()


def main():
    # settings
    h = 1.0
    lamb = 1.0
    nu = 1.0
    n = 200
    fig_path = '../figures/assignment1_result.png'
    np.random.seed(0)


    # generate data
    x, y = generate_data(n)
    #print(x.shape, y.shape)


    # train
    theta = lrls(x, y, h=h, l=lamb, nu=nu)

    # supervised
    #x, y = x[y!=0], y[y!=0]
```

```
    #theta = supervised_train(x, y, h=h, l=lamb,)


    # result
    print(f'#data: {n}')
    print(f'h = {h}  lambda = {lamb}  nu = {nu}')
    #print('theta = \n', theta)

    visualize(x, y, theta, path=fig_path)


if __name__ == '__main__':
    main()
```

Listings 2: `assignment3.py`

```
# -*- coding: utf-8 -*-


import numpy as np
import matplotlib.pyplot as plt


def generate_data(n_total, n_positive):
    x = np.random.normal(size=(n_total, 2))
    x[:n_positive, 0] -= 2
    x[n_positive:, 0] += 2
    x[:, 1] *= 2.
    y = np.empty(n_total, dtype=np.int64)
    y[:n_positive] = 0
    y[n_positive:] = 1
    return x, y


def calc_A(y, y_tilde, x_train, y_train):
    indices_i = np.where(y_train == y)[0]
    indices_i_tilde = np.where(y_train == y_tilde)[0]

    if len(indices_i) * len(indices_i_tilde) == 0:
        return 0

    x_tilde = x_train[indices_i_tilde]
    A = 0
    for idx_i in indices_i:
        A += np.sum(np.sqrt(np.sum((x_train[idx_i] - x_tilde)**2, axis=0)))
    A /= (len(indices_i) * len(indices_i_tilde))
    return A


def calc_b(y, x_train, y_train, x_test):
    x_i = x_test[y_train == y]

    if len(x_i) * len(x_test) == 0:
        return 0

    b = 0
    for x_i_dash in x_test:
        b += np.sum(np.sqrt(np.sum((x_i_dash - x_i)**2, axis=0)))
    b /= (len(x_i) * len(x_test))
    return b


```

```
def estimate_pi(x_train, y_train, x_test):
    x, y = x_train, y_train
    A_pp = calc_A(1, 1, x, y)
    A_pm = calc_A(1, -1, x, y)
    A_mm = calc_A(-1, -1, x, y)
    b_p = calc_b(1, x_train, y_train, x_test)
    b_m = calc_b(-1, x_train, y_train, x_test)

    pi_hat = (A_pm - A_mm - b_p + b_m) / (2*A_pm - A_pp - A_mm)
    pi_hat = min(1, max(0, pi_hat))
    return pi_hat


def cwls(train_x, train_y, test_x, is_weighted=True):
    n = train_y.shape[0]

    if is_weighted:
        pi_hat = estimate_pi(train_x, train_y, test_x)
        Pi = np.zeros(n)
        Pi[train_y == 1] = pi_hat
        Pi[train_y == -1] = 1 - pi_hat
        Pi = np.diag(Pi)
    else:
        Pi = np.eye(n)

    Phi = np.concatenate([np.ones(n)[:, np.newaxis], train_x], axis=1)

    theta = np.linalg.inv(Phi.T.dot(Pi).dot(Phi)).dot(Phi.T).dot(Pi).dot(train_y)
    return theta


def visualize(train_x, train_y, test_x, test_y, theta, is_weighted=True):
    str_weighted = ’weighted’ if is_weighted else ’unweighted’
    for x, y, name in [(train_x, train_y, ’train’), (test_x, test_y, ’test’)]:
        plt.xlim(-5., 5.)
        plt.ylim(-7., 7.)
        lin = np.array([-5., 5.])
        plt.plot(lin, -(theta[2] + lin * theta[0]) / theta[1])
        plt.scatter(x[y==0][:, 0], x[y==0][:, 1], marker=’$O$’, c=’blue’)
        plt.scatter(x[y==1][:, 0], x[y==1][:, 1], marker=’$X$’, c=’red’)
        plt.savefig(’../figures/assignment3_result_{}_{}.png’.format(str_weighted, name))
        plt.show()

```

```

def main():
    # settings
    is_weighted = False
    np.random.seed(0)


    # generate data
    train_x, train_y = generate_data(n_total=100, n_positive=90)
    eval_x, eval_y = generate_data(n_total=100, n_positive=10)


    # train
    theta = cwls(train_x, train_y, eval_x, is_weighted=is_weighted)


    # result
    print('result')
    visualize(train_x, train_y, eval_x, eval_y, theta,
    is_weighted=is_weighted)


if __name__ == '__main__':
    main()
```