Advanced Machine Learning Midterm Assignment

(Final report assigned by Shimosaka)

20M14894 工学院経営工学系経営工学コース　笛木正雄

Problem 3

$$\widehat{\boldsymbol{w}} = \underset{w\in\mathbb{R}^d}{\operatorname{argmin}}\left(\sum_{i=1}^{n}\max(0, 1 - y_i\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i) + \lambda\|\boldsymbol{w}\|_2^2\right)$$

1.　双対ラグランジュ関数の導出

$$\min.\ \frac{1}{n}\sum_{i=1}^{n}\xi_i + \lambda\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$$

$$\mathrm{s.t.}\,\xi_i \geq \max(0, 1 - y_i\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i)\,, i = 1, \dots, n$$

$$\min.\ \boldsymbol{1}^{\mathrm{T}}\boldsymbol{\xi} + \lambda\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$$

$$\mathrm{s.t.}\,\xi_i \geq 0, \xi_i \geq 1 - y_i\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i), i = 1, \dots, n$$

上記のラグランジュ関数を考える。

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \boldsymbol{1}^{\mathrm{T}}\boldsymbol{\xi} + \lambda\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + \sum_{i=1}^{n}\alpha_i(1 - y_i\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - \xi_i) - \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\xi}$$

KKT 条件を以下に示す。

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} = 2\lambda\boldsymbol{w} - \sum_{i=1}^{n}\alpha_i y_i\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i \Rightarrow \widehat{\boldsymbol{w}} = \frac{1}{2\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i y_i\boldsymbol{x}_i$$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\xi}} = \boldsymbol{1} - \boldsymbol{\alpha} - \boldsymbol{\beta} \Rightarrow \widehat{\alpha}_i + \widehat{\beta}_i = 1, i = 1, \dots, n$$

$$\boldsymbol{\alpha} \geq \boldsymbol{0}$$

$$\boldsymbol{\beta} \geq \boldsymbol{0}$$

$$\widehat{\alpha}_i(1 - y_i\widehat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{x}_i - \xi_i) = 0, i = 1, \dots, n$$

$$\widehat{\beta}_i\widehat{\xi}_i = 0, i = 1, \dots, n$$

$\widehat{\boldsymbol{w}}, \widehat{\alpha}_i + \widehat{\beta}_i = 1$を$\mathcal{L}$に代入する。

$$\tilde{\mathcal{L}}(\boldsymbol{\alpha}) = \boldsymbol{1}^{\mathrm{T}}\boldsymbol{\xi} + \lambda\left(\frac{1}{2\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i y_i\boldsymbol{x}_i\right)^{\mathrm{T}}\left(\frac{1}{2\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i y_i\boldsymbol{x}_i\right) + \sum_{i=1}^{n}\widehat{\alpha}_i\left(1 - y_i\left(\frac{1}{2\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i y_i\boldsymbol{x}_i\right)^{\mathrm{T}}\boldsymbol{x}_i - \xi_i\right) - (\boldsymbol{1} - \boldsymbol{\alpha})^{\mathrm{T}}\boldsymbol{\xi}$$

$$= \frac{1}{4\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n}\widehat{\alpha}_i\widehat{\alpha}_j y_i y_j\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j - \frac{1}{2\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n}\widehat{\alpha}_i\widehat{\alpha}_j y_i y_j\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j - \sum_{i=1}^{n}\alpha_i\xi_i + \sum_{i=1}^{n}\alpha_i + \boldsymbol{1}^{\mathrm{T}}\boldsymbol{\xi} - (\boldsymbol{1} - \boldsymbol{\alpha})^{\mathrm{T}}\boldsymbol{\xi}$$

$$= \frac{1}{4\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n}\widehat{\alpha}_i\widehat{\alpha}_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j - \frac{1}{2\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n}\widehat{\alpha}_i\widehat{\alpha}_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j - \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\xi} + \mathbf{1}^{\mathrm{T}}\boldsymbol{\alpha} + \mathbf{1}^{\mathrm{T}}\boldsymbol{\xi} - \mathbf{1}^{\mathrm{T}}\boldsymbol{\xi} + \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\xi}$$

$$= -\frac{1}{4\lambda}\boldsymbol{\alpha}^{\mathrm{T}}K\boldsymbol{\alpha} + \mathbf{1}^{\mathrm{T}}\boldsymbol{\alpha} \quad (\because K_{ij} = \{y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j\})$$

$\boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\beta} \geq \mathbf{0}$より、$\mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{1}$

ここで、$\tilde{\mathcal{L}}(\boldsymbol{\alpha})$は、$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$の下界であるため、

$$\text{max.} \quad -\frac{1}{4\lambda}\boldsymbol{\alpha}^{\mathrm{T}}K\boldsymbol{\alpha} + \mathbf{1}^{\mathrm{T}}\boldsymbol{\alpha}$$

$$\text{s.t.} \quad \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{1}$$

2. KKT 条件から、$\boldsymbol{\alpha}$から$\boldsymbol{w}$が導出されることを示す。

1 より、

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 2\lambda\boldsymbol{w} - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i \Rightarrow \widehat{\boldsymbol{w}} = \frac{1}{2\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i y_i \boldsymbol{x}_i$$

3. Projected gradient を使用して、1 で導出した最適化問題を解くコードを実装する。

```
class SVM:
    def __init__(self, lam):
        self.lam = lam

    def fit(self, x, y, eta):
        n = x.shape[0]
        self.alpha = np.random.rand(n)
        K = np.zeros((n, n))
        for i in range(n):
            for j in range(n):
                K[i][j] = y[i] * y[j] * x[i] @ x[j]

        while True:
            self.alpha = np.clip(self.alpha - eta * ((1 /
(2 * self.lam)) * K @ self.alpha - 1), a_min=0, a_max=1)
            score_D = -(1 / (4 * self.lam)) * self.alpha @
K @ self.alpha + sum(self.alpha)

            self.w = (1 / (2 * self.lam)) *
sum([self.alpha[i] * y[i] * x[i] for i in range(n)])
```

```
            score_P = sum([np.max([0, 1 - y[i] * self.w @ x[i]]) for i in range(n)])+ self.lam * self.w @ self.w

            if abs(score_P - score_D) <= 1e-5:
                break

    def predict(self, x):
        return np.array([2 * (self.w @ xi >= 0) - 1 for xi in x])
```

データセット 2

```
# dataset 2
np.random.seed(1)
n = 30
omega = np.random.randn()
noise = 0.8 * np.random.randn(n)

x_d2 = np.random.randn(n, 2) + 0
y_d2 = 2 * (omega * x_d2[:,0] + x_d2[:,1] + noise > 0) - 1
```

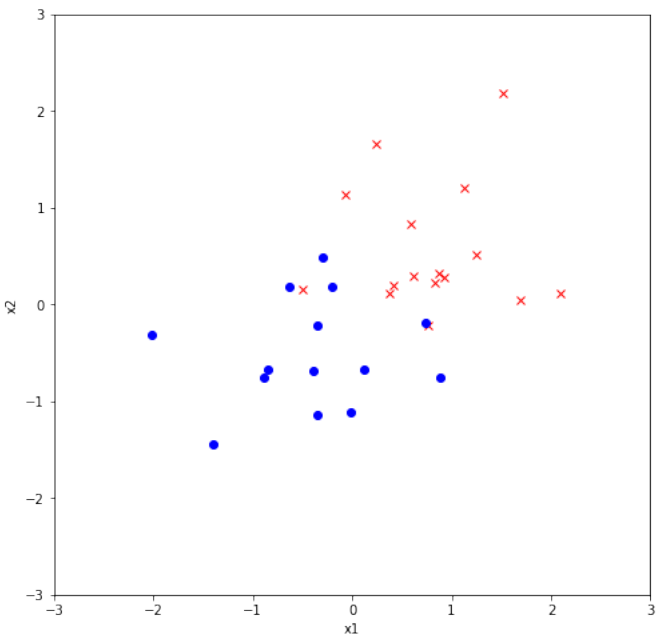


図 1:データセット 2

データセット 2 での分類領域を以下に示す。

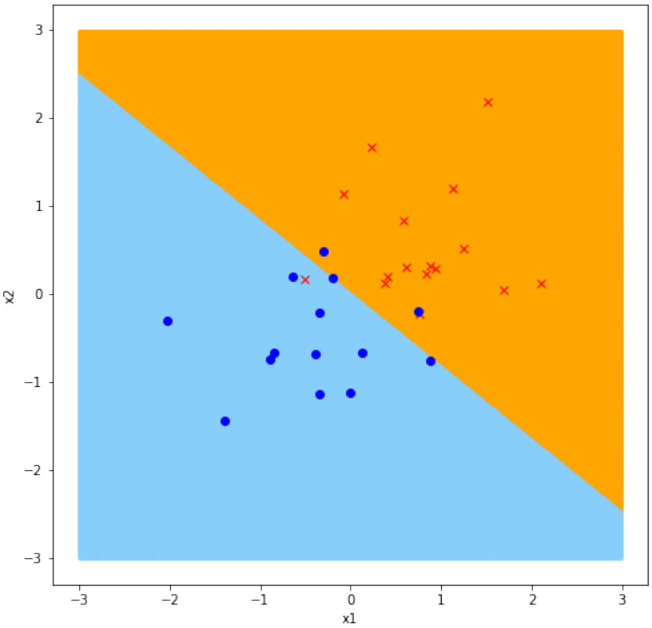


図 2:SVM(lambda=3)による分類結果

Problem 4

$$\hat{\boldsymbol{w}} = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \max(0, 1 - y_i \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i) + \lambda \|\boldsymbol{w}\|_1 \right)$$

1. 線形計画問題を導出する

$$\min. \ \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \sum_{i=1}^{n} |w_i|$$

$$\mathrm{s.t.}\ \xi_i \geq \max(0, 1 - y_i \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i),\ i = 1, \dots, n$$

$$\begin{aligned} \min.\ & \mathbf{1}^{\mathrm{T}} \boldsymbol{\xi} + \lambda \mathbf{1}^{\mathrm{T}} \boldsymbol{e} \\ \mathrm{s.t.}\ & \xi_i \geq 0, i = 1, \dots, n, \\ & \xi_i \geq 1 - y_i \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i, i = 1, \dots, n, \\ & -\boldsymbol{w} \leq \boldsymbol{e} \leq \boldsymbol{w}, \\ & \mathbf{0} \leq \boldsymbol{e} \end{aligned}$$

$$\begin{aligned} \min.\ & \mathbf{1}^{\mathrm{T}} \boldsymbol{\xi} + \lambda \mathbf{1}^{\mathrm{T}} \boldsymbol{e} \\ \mathrm{s.t.}\ & -\xi_i \leq 0, i = 1, \dots, n, \\ & -y_i \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i - \xi_i \leq -1, i = 1, \dots, n, \\ & -w_i - e_i \leq 0, i = 1, \dots, n, \\ & w_i - e_i \leq 0, i = 1, \dots, n, \\ & -e_i \leq 0, i = 1, \dots, n, \end{aligned}$$

ここで、

$$
\boldsymbol{c} = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ \lambda \\ \vdots \\ \lambda \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{n+d+d}, \qquad \boldsymbol{z} = \begin{pmatrix} \boldsymbol{\xi} \\ \boldsymbol{e} \\ \boldsymbol{w} \end{pmatrix} \in \mathbb{R}^{n+d+d}, \qquad \boldsymbol{b} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ \vdots \\ -1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}
$$

$$
A = \begin{pmatrix}
-1 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & -1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
-1 & \cdots & 0 & 0 & \cdots & 0 & -y_1 x_{11} & \cdots & -y_1 x_{1d} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
0 & \cdots & -1 & 0 & \cdots & 0 & -y_n x_{n1} & \cdots & -y_n x_{nd} \\
0 & \cdots & 0 & -1 & \cdots & 0 & -1 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & \cdots & -1 & 0 & \cdots & -1 \\
0 & \cdots & 0 & -1 & \cdots & 0 & 1 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & \cdots & -1 & 0 & \cdots & 1 \\
0 & \cdots & 0 & -1 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & \cdots & -1 & 0 & \cdots & 0
\end{pmatrix} \in \mathbb{R}^{(n+n+d+d+d)\times(n+d+d)}
$$

とおくと、以下のように表せる。

min. $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{z}$

s.t. $A\boldsymbol{z} \leq \boldsymbol{b}$

2. cvxpy と proximal subgradient method で実装する。

cvxpy

```
def LP_L1_hinge(x, y, lam):
    lam = 2
    n = x.shape[0]
    d = x.shape[1]
    c = np.array([1]*n + [lam]*d + [0]*d).reshape(n+d+d, 1)
    z = cv.Variable((n+d+d, 1))

    A1 = np.hstack([-np.eye(n), np.zeros((n, d+d))])
```

```
    A2 = np.hstack([-np.eye(n), np.zeros((n, d)), np.vstack([-y[i]
* x[i] for i in range(n)])])
    A3 = np.hstack([np.zeros((d, n)), -np.eye(d), -np.eye(d)])
    A4 = np.hstack([np.zeros((d, n)), -np.eye(d), np.eye(d)])
    A5 = np.hstack([np.zeros((d, n)), -np.eye(d), np.zeros((d,
d))])
    A = np.vstack([A1, A2, A3, A4, A5])

    b = np.array([0]*n + [-1]*n + [0]*d + [0]*d +
[0]*d).reshape(n+n+d+d+d, 1)

    obj_fn = c.T @ z
    objective = cv.Minimize(obj_fn)

    constraints = [A @ z <= b]

    prob = cv.Problem(objective, constraints)
    result = prob.solve(solver=cv.CVXOPT)

    w = np.ravel(z.value[-d:])
    return w
```

proximal subgradient method

```
def subgrad_loss(x, y, w):
    ywx = y * w @ x
    if ywx > 1:
        return np.zeros((w.shape))
    elif ywx == 1:
        # e = np.random.rand()
        e = 1
        return -e * y * x
    else:
        return -y * x

def subgrad_reg(lam, w):
    if w > 0:
```

```
        return lam
    elif w == 0:
        # e = 2 * np.random.rand() - 1
        e = 1
        return e * lam
    else:
        return -lam

def subgrad(x, y, eta, T=10000):
    eta = 0.1
    n = x_d4.shape[0]
    w = np.random.rand(4)
    lam = 2

    loss_hist = []

    for t in range(T):
        g_loss = np.sum(np.array([subgrad_loss(x_d4[i], y_d4[i], w)
for i in range(n)]), axis=0)
        g_reg = np.array([subgrad_reg(lam, w[i]) for i in
range(len(w))])

        w = w - eta * (1 / (t+1)**(1/2)) * (g_loss + g_reg)
    return w
```

データセット 4

```
# dataset 4
np.random.seed(0)
n = 200
x_d4 = 3 * (np.random.rand(n, 4) - 0.5)
y_d4 = (2 * x_d4[:, 0] - 1 * x_d4[:,1] + 0.5 + 0.5 *
np.random.randn(n)) > 0
y_d4 = 2 * y_d4 -1
```

データセット 4 で cvxpy と proximal subgradient method で求めた w は以下の通りである。

表 1: cvxpy と proximal subgradient method により求められたパラメータ

	w1	w2	w3	w4
cvxpy	2.3415193	-0.94152256	-0.16324716	-0.15151678
subgradient method	2.34512617	-0.94749772	-0.1620871	-0.15158547

subgradient method でのイテレーションごとの損失は以下の通りである。

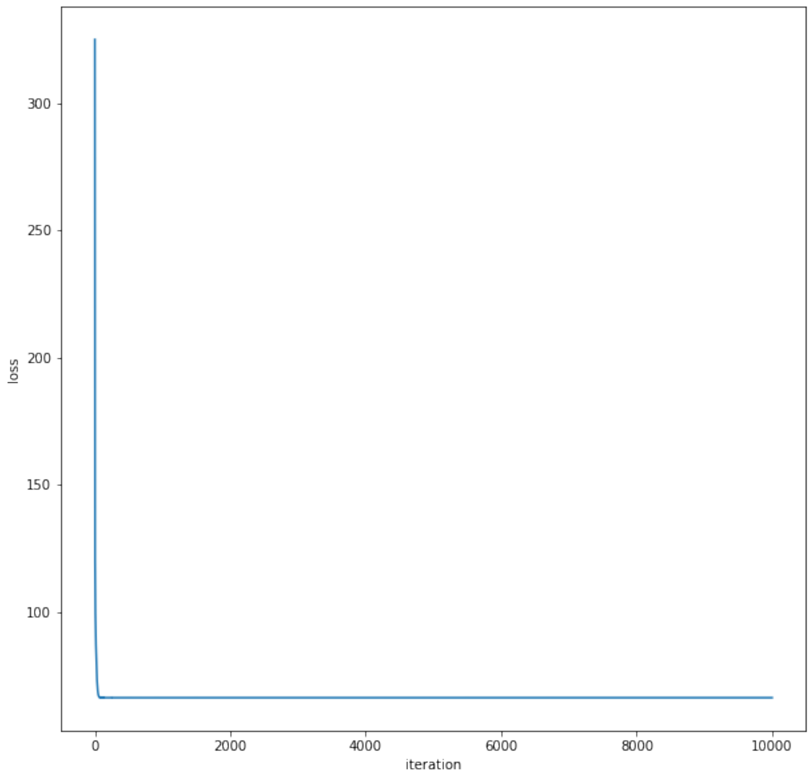


図 3:イテレーションごとの損失結果

Problem 3

1. 損失関数に hinge 関数を使用し、L2 正則化の分類器を考える。Problem3 で実装した SVM クラスを使用する。

データセット 1

```
# dataset 1
np.random.seed(123)
n = 100
x_d1 = 3 * (np.random.rand(n, 2)-0.5)
radius = x_d1[:,0]**2 + x_d1[:,1]**2
y_d1 = (radius > 0.7 + 0.1 * np.random.randn(n)) &( radius < 2.2
+ 0.1 * np.random.randn(n))
y_d1 = 2 * y_d1 -1
```

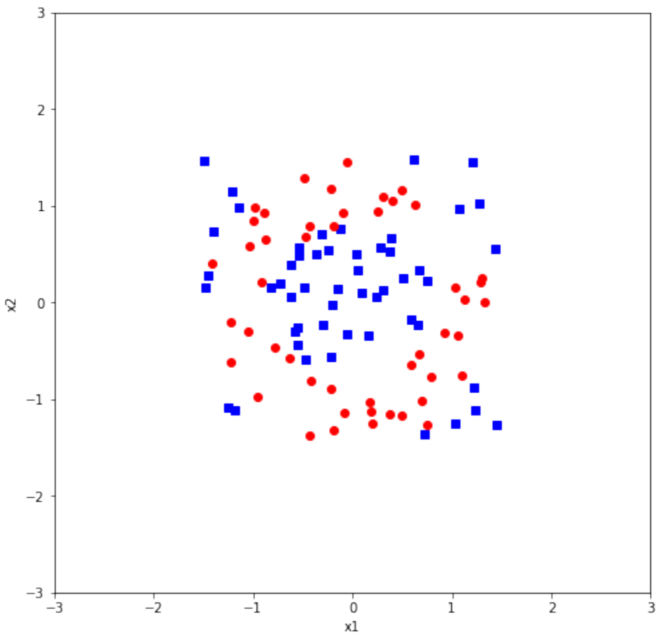


図 4:データセット 1

カーネルを使用しない場合の分類結果を以下に示す

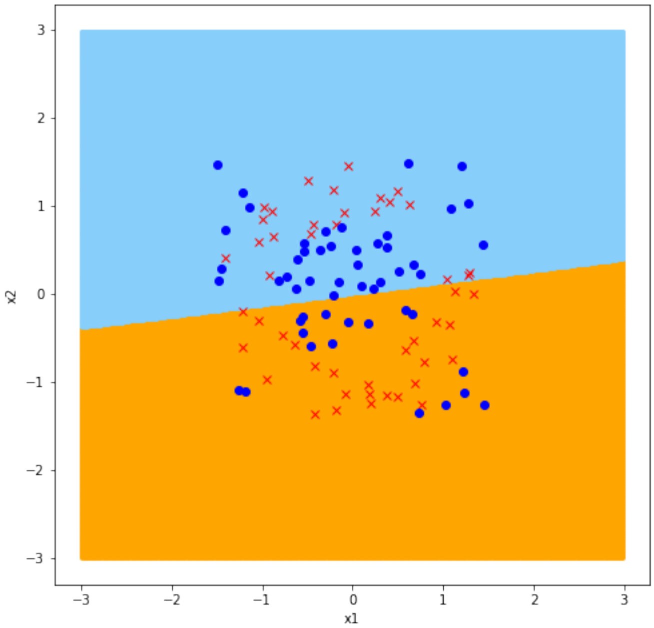


図 5:カーネルを使用しない SVM(λ=3)の分類結果(データセット 1)

ガウシアンカーネルは以下のように実装した。

```
class GausianKernel:
    def __init__(self, x_all, alpha, d_tilda):
        self.alpha = alpha
        # 使用するデータをランダム抽出
        self.x_use_index =
np.random.choice(np.array(range(len(x_all))), d_tilda,
replace=False)
        self.x_d_tilda = x_all[self.x_use_index]


    def __call__(self, input_x_vec):
        return np.array([np.exp(-self.alpha * (np.linalg.norm(x_vec
- self.x_d_tilda, ord=2, axis=1)**2)) for x_vec in input_x_vec])
```

次に、データセット 1 を使用して、ガウシアンカーネル($\tilde{d}$=90, α=3)を用いた SVM(λ=3)の分類結果を以下に示す。

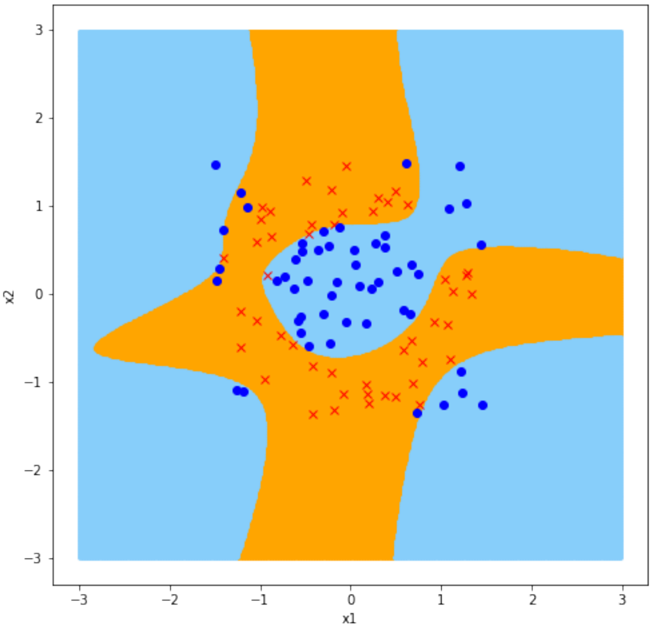


図 6:ガウシアンカーネル($\tilde{d}$=90, α=3)を使用した SVM(λ=3)の分類結果(データセット 1)

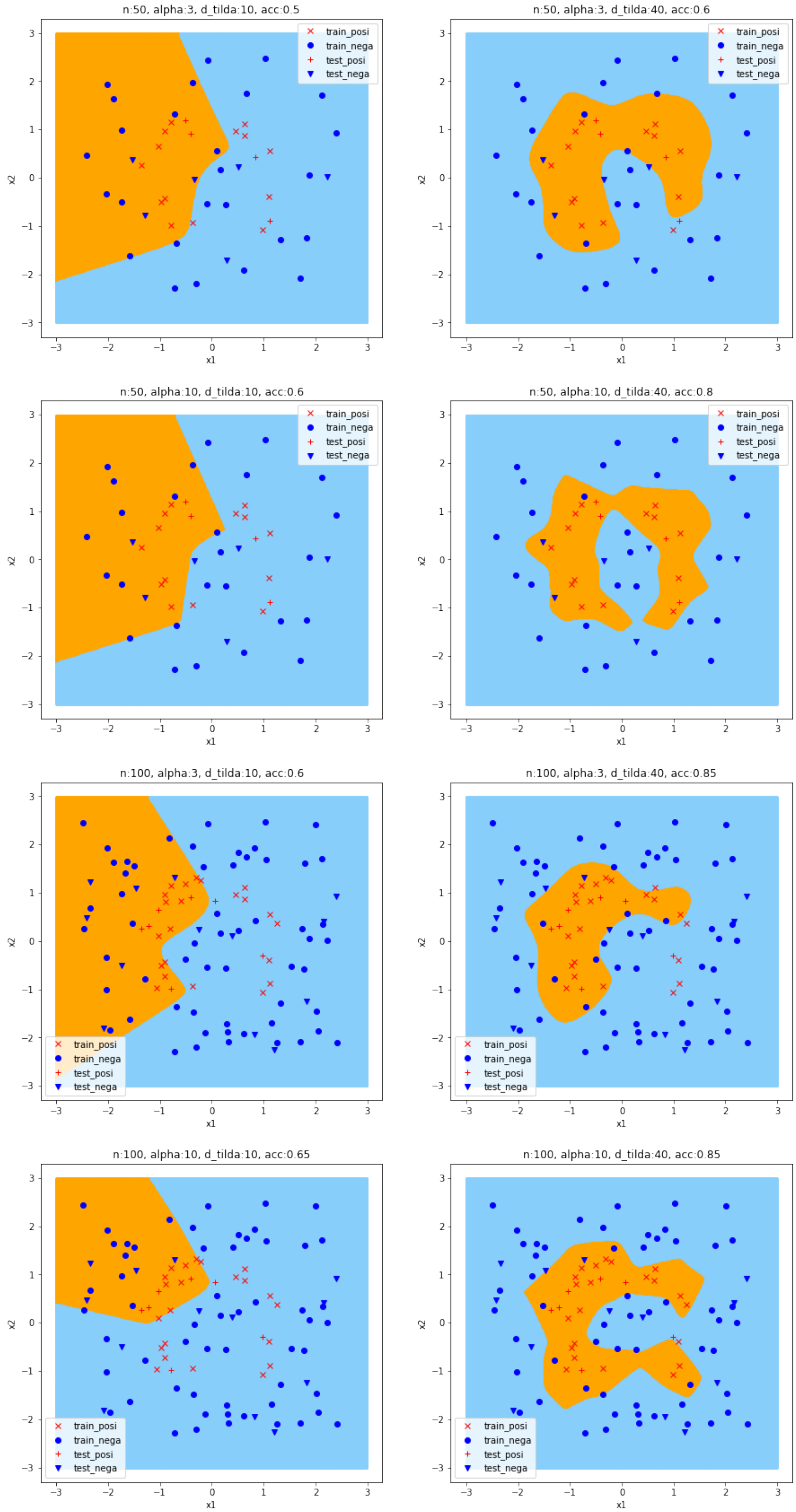
2.　ハイパーパラメータごとの性能を比較する。

データセット 1 を 8:2 で学習データとテストデータに分けて各ハイパーパラメータを評価する。

データ数(n=50, 100)、ガウシアンカーネルのパラメータ(α=0.1, 10)、写像先の次元数($\tilde{d}$=2, 40)について検証する。正則化係数はλ=3 で固定とする。

写像先の次元数が高くなると、モデルの自由度が高くなることがわかる。

また、ガウシアンカーネルのパラメータが大きくなると、モデルの自由度が高くなることがわかる。今回のデータセットでは、モデルの自由度があるほど、精度が向上しやすいため、写像先の次元数が高く、ガウシアンカーネルのパラメータの大きいモデルがテスト精度が高くなっている。


n:50, alpha:3, d_tilda:10, acc:0.5
n:50, alpha:3, d_tilda:40, acc:0.6
n:50, alpha:10, d_tilda:10, acc:0.6
n:50, alpha:10, d_tilda:40, acc:0.8
n:100, alpha:3, d_tilda:10, acc:0.6
n:100, alpha:3, d_tilda:40, acc:0.85
n:100, alpha:10, d_tilda:10, acc:0.65
n:100, alpha:10, d_tilda:40, acc:0.85
train_posi
train_nega
test_posi
test_nega
x1
x2

下記リンクにて、上記で作成したコードを公開しています。
[Github]
https://github.com/plue1011/Advanced_Machine_Learning/blob/master/Assignment_midterm.ipynb