Machine Learning Assignment 3

Bohan YANG
Student ID: 2330016056

November 5, 2025

Question 1

$$\text{Class } +1: \ (1,2),\ (2,3) \qquad \text{Class } -1: \ (2,1),\ (3,2)$$

(1)

The separating hyperplane is

$$w^\top x + b = 0, \quad w \in \mathbb{R}^2,\ b \in \mathbb{R}.$$

For the hard-margin SVM, the optimization problem is

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|_2^2 \\ \text{s.t.} \quad & y_i(w^\top x_i + b) \geq 1, \quad i = 1, \ldots, 4. \end{aligned}$$

(2)

The Lagrangian is

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|_2^2 - \sum_{i=1}^{4} \alpha_i \Big(y_i(w^\top x_i + b) - 1 \Big).$$

The KKT conditions:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{4} \alpha_i y_i x_i,$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{4} \alpha_i y_i = 0.$$

Substitute these into $\mathcal{L}$ to obtain the dual optimization problem:

$$\boxed{\begin{aligned} \max_{\alpha \geq 0} \quad & \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j x_i^\top x_j, \\ \text{s.t.} \quad & \sum_{i=1}^{4} \alpha_i y_i = 0. \end{aligned}}$$

(3)

Assume all $\alpha_i = 0.5$ and that the point $(1, 2)$ is a support vector.

Then,

$$w = \sum_{i=1}^{4} \alpha_i y_i x_i = 0.5\Big[(1,2) + (2,3) - (2,1) - (3,2)\Big] = 0.5\,(-2,2) = (-1,1).$$

For the support vector $(1, 2)$ with $y = +1$, the KKT equality holds:

$$y_i(w^\top x_i + b) = 1 \Rightarrow (-1,1) \cdot (1,2) + b = 1 \Rightarrow (-1+2) + b = 1 \Rightarrow b = 0.$$

Therefore, the optimal separating hyperplane is

$$w^\top x + b = 0 \quad \Rightarrow \quad (-1,1) \cdot (x_1, x_2) + 0 = 0 \quad \Rightarrow \quad \boxed{x_2 - x_1 = 0.}$$

Question 2

(1)

Given training data

$$\{(x_i, y_i)\}_{i=1}^{n}, \quad x_i \in \mathbb{R}^d,\ y_i \in \{+1, -1\}.$$

Let

$$w \in \mathbb{R}^d, \quad b \in \mathbb{R}, \quad \xi = [\xi_1, \ldots, \xi_n]^\top \in \mathbb{R}^n,\ \xi_i \geq 0.$$

The soft-margin SVM primal optimization problem is

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|_2^2 + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y_i(w^\top x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n, \\ & \xi_i \geq 0, \quad i = 1, \ldots, n. \end{aligned}$$

With $C = 1$, this becomes

$$\min_{w,b,\xi} \ \frac{1}{2}\|w\|_2^2 + \sum_{i=1}^{n} \xi_i \quad \text{s.t.}\ \ y_i(w^\top x_i + b) \geq 1 - \xi_i,\ \ \xi_i \geq 0.$$

(2)

Introduce Lagrange multipliers

$$\begin{aligned}
&\alpha_i \geq 0 \text{ for } y_i(w^\top x_i + b) \geq 1 - \xi_i \\
&\mu_i \geq 0 \text{ for } \xi_i \geq 0.
\end{aligned}$$

The Lagrangian is

$$\mathcal{L}(w, b, \xi; \alpha, \mu) = \frac{1}{2}\|w\|_2^2 + \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i \Big(y_i(w^\top x_i + b) - 1 + \xi_i \Big) - \sum_{i=1}^n \mu_i \xi_i.$$

Stationarity conditions:

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^n \alpha_i y_i x_i, \\
&\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0, \\
&\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Rightarrow 1 - \alpha_i - \mu_i = 0 \Rightarrow \alpha_i + \mu_i = 1.
\end{aligned}$$

Since $\mu_i \geq 0$, we have

$$0 \leq \alpha_i \leq 1 \quad (\text{generally } 0 \leq \alpha_i \leq C).$$

Substitute back:

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\|w\|_2^2 - \sum_{i=1}^n \alpha_i y_i w^\top x_i + \sum_{i=1}^n \alpha_i \\
&= -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^n \alpha_i,
\end{aligned}$$

Dual problem:

$$\begin{aligned}
\max_{\alpha} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^\top x_j, \\
\text{s.t.} \quad & \sum_{i=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq 1.
\end{aligned}$$

KKT conditions:

$$\begin{cases}
w = \sum_i \alpha_i y_i x_i, \\
\sum_i \alpha_i y_i = 0, \\
0 \leq \alpha_i \leq 1, \\
y_i(w^\top x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \\
\alpha_i [y_i(w^\top x_i + b) - 1 + \xi_i] = 0, \\
(1 - \alpha_i)\xi_i = 0.
\end{cases}$$

(3)

At optimality,

$$w^\star = \sum_{i=1}^{n} \alpha_i^\star y_i x_i.$$

For any support vector with $0 < \alpha_i^\star < 1$,

$$y_i\Big((w^\star)^\top x_i + b^\star\Big) = 1.$$

Thus,

$$b^\star = y_i - (w^\star)^\top x_i.$$

The decision function is

$$f(x) = \text{sign}\left((w^\star)^\top x + b^\star\right), \quad \text{and the decision boundary is } (w^\star)^\top x + b^\star = 0.$$

(4)

In the Lagrangian, ξ_i appears only linearly:

$$\sum_i \xi_i - \sum_i \alpha_i \xi_i - \sum_i \mu_i \xi_i = \sum_i \xi_i (1 - \alpha_i - \mu_i).$$

Setting the derivative w.r.t. ξ_i to zero gives $1 - \alpha_i - \mu_i = 0$, or $\alpha_i + \mu_i = 1$.

Substituting this back eliminates all terms involving ξ_i, so ξ_i no longer appears in the dual function.

Its only effect is to constrain $\alpha_i \le C(= 1)$, which forms the box constraint $0 \le \alpha_i \le C$.

Question 3

$x_1 = (1, 2),\ y_1 = +1; \quad x_2 = (2, 3),\ y_2 = +1; \quad x_3 = (3, 1),\ y_3 = -1; \quad x_4 = (4, 3),\ y_4 = -1.$

Polynomial kernel: $k(x_i, x_j) = (x_i^\top x_j + 1)^2$. Regularization $C = 1$.

(1)

$$\begin{aligned}
&x_1^\top x_1 = 5, \quad x_1^\top x_2 = 8, \quad x_1^\top x_3 = 5, \quad x_1^\top x_4 = 10,\\
&x_2^\top x_2 = 13, \quad x_2^\top x_3 = 9, \quad x_2^\top x_4 = 17,\\
&x_3^\top x_3 = 10, \quad x_3^\top x_4 = 15,\\
&x_4^\top x_4 = 25.
\end{aligned}$$

$$K = \begin{pmatrix} (5+1)^2 & (8+1)^2 & (5+1)^2 & (10+1)^2 \\ (8+1)^2 & (13+1)^2 & (9+1)^2 & (17+1)^2 \\ (5+1)^2 & (9+1)^2 & (10+1)^2 & (15+1)^2 \\ (10+1)^2 & (17+1)^2 & (15+1)^2 & (25+1)^2 \end{pmatrix} = \begin{pmatrix} 36 & 81 & 36 & 121 \\ 81 & 196 & 100 & 324 \\ 36 & 100 & 121 & 256 \\ 121 & 324 & 256 & 676 \end{pmatrix}.$$

(2)

Using the soft-margin SVM dual with kernel,

$$\begin{aligned}\max_{\alpha\in\mathbb{R}^4} \quad & \sum_{i=1}^{4} \alpha_i - \frac{1}{2}\sum_{i=1}^{4}\sum_{j=1}^{4} \alpha_i\alpha_j y_i y_j K_{ij}, \\ \text{s.t.} \quad & \sum_{i=1}^{4} \alpha_i y_i = 0, \qquad 0 \le \alpha_i \le C(=1).\end{aligned}$$

At optimality, the decision function is

$$f(x) = \sum_{i=1}^{4} \alpha_i^\star y_i\, k(x_i, x) + b^\star, \quad \hat{y}(x) = \operatorname{sign}\big(f(x)\big).$$

(3)

Given

$$\alpha_1 = 0.0182, \quad \alpha_2 = 0.0068, \quad \alpha_3 = 0.0250, \quad \alpha_4 = 0,$$

and x_3 is a support vector with $y_3 = -1$.

For any margin support vector x_s with $0 < \alpha_s < C$,

$$y_s\Big(\sum_{j=1}^{4} \alpha_j y_j K_{js} + b\Big) = 1.$$

Take $s = 3$:

$$-1\Big(\sum_{j=1}^{4} \alpha_j y_j K_{j3} + b\Big) = 1 \;\Rightarrow\; b = -1 - \sum_{j=1}^{4} \alpha_j y_j K_{j3}.$$

Compute the sum using the 3rd column of K: $(36, 100, 121, 256)^\top$.

$$\sum_{j=1}^{4} \alpha_j y_j K_{j3} = 0.0182\cdot(+1)\cdot 36 + 0.0068\cdot(+1)\cdot 100 + 0.0250\cdot(-1)\cdot 121 + 0\cdot(-1)\cdot 256 = -1.6898.$$

Hence

$$\boxed{b = -1 - (-1.6898) = 0.6898.}$$

(4)

Evaluate

$$f(x_5) = \sum_{i=1}^{4} \alpha_i y_i\, k(x_i, x_5) + b.$$

First compute $k(x_i, x_5) = (x_i^\top x_5 + 1)^2$:

$$\begin{aligned}x_1^\top x_5 &= 1\cdot 2 + 2\cdot 1 = 4 \Rightarrow k(x_1, x_5) = (4+1)^2 = 25, \\ x_2^\top x_5 &= 2\cdot 2 + 3\cdot 1 = 7 \Rightarrow k(x_2, x_5) = (7+1)^2 = 64, \\ x_3^\top x_5 &= 3\cdot 2 + 1\cdot 1 = 7 \Rightarrow k(x_3, x_5) = (7+1)^2 = 64, \\ x_4^\top x_5 &= 4\cdot 2 + 3\cdot 1 = 11 \Rightarrow k(x_4, x_5) = (11+1)^2 = 144.\end{aligned}$$

Thus

$$\begin{aligned}
f(x_5) &= 0.0182 \cdot (+1) \cdot 25 + 0.0068 \cdot (+1) \cdot 64 + 0.0250 \cdot (-1) \cdot 64 + 0 \cdot (-1) \cdot 144 + 0.6898 \\
&= 0.455 + 0.4352 - 1.6 + 0 + 0.6898 \\
&= -0.0200 \text{ (approximately).}
\end{aligned}$$

Therefore,

$$\boxed{\hat{y}(x_5) = \operatorname{sign}(f(x_5)) = -1.}$$

DS4023_assignment3

November 5, 2025

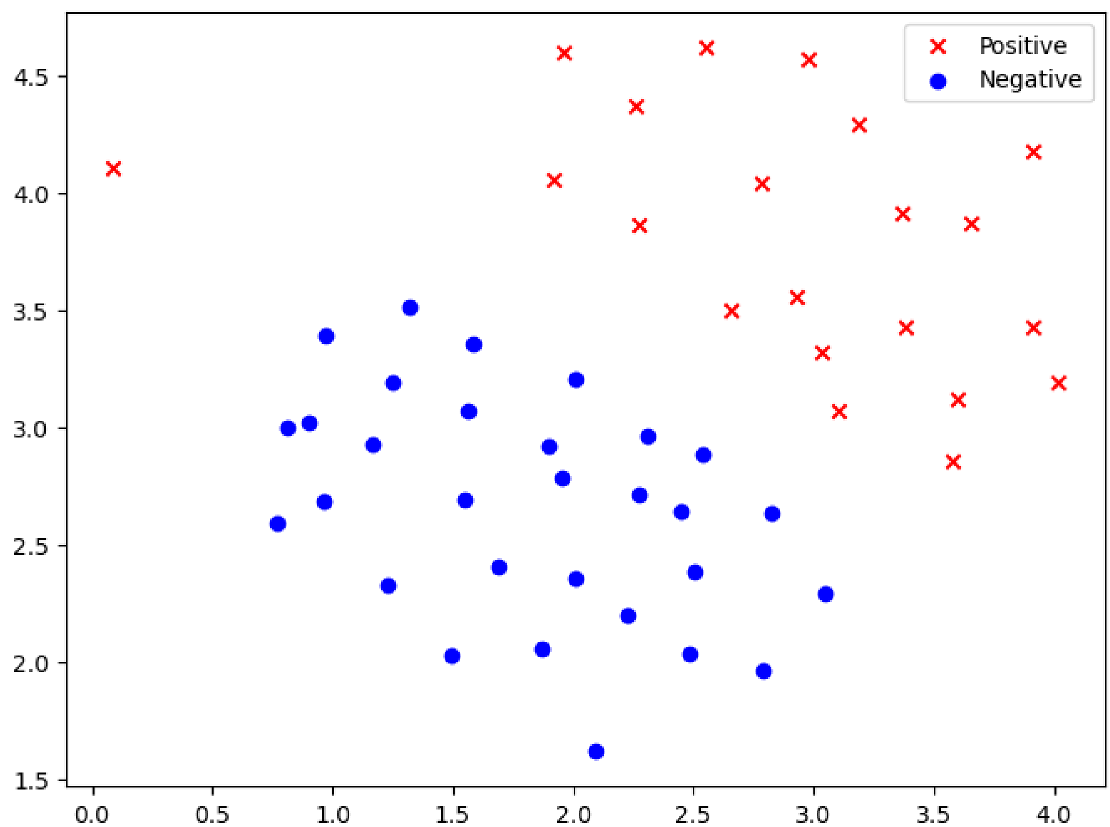
1 DS4023 Assignment 3: SVM and Ensemble Learning(45 pts)

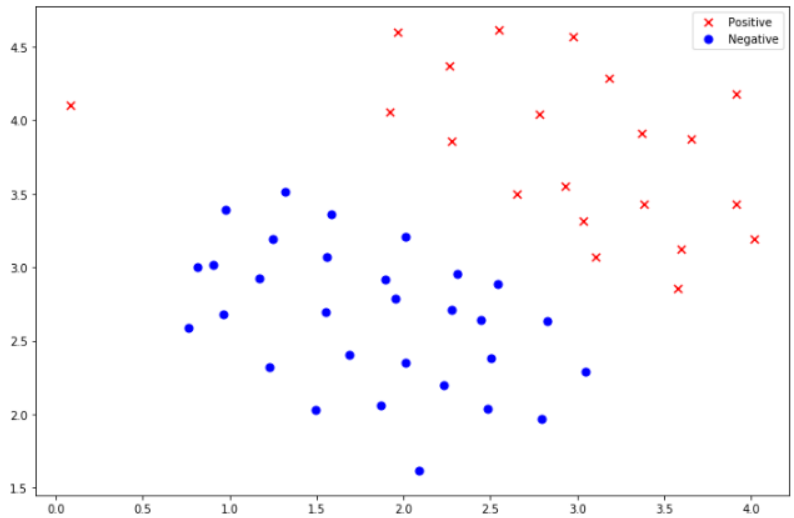
1.0.1 Q1: Soft-margin SVM (9 pts)

You are given a dataset (svm_soft.mat) containing 2D points ['X'] from two classes with corresponding label (0 or 1) in column ['y']. Please visualize the datapoints and classify them using linear SVM with soft margin. You may assume that the penalty term C is set to 1 in $min_{w,b}\frac{1}{2}||w||^2 + C\sum_i \epsilon_i$.

[1]:
```
"""
Part 1: Visualize the datapoints. Your plot should look similar to the sample␣
↪one.
"""

import pandas as pd
import matplotlib.pyplot as plt
from scipy.io import loadmat


def plot_svm_soft(boundary=False, svc=None):
    raw_data = loadmat('data/svm_soft.mat')

    data = pd.DataFrame(raw_data['X'], columns=['X1', 'X2'])
    data['y'] = raw_data['y']

    pos = data['y'] == 1
    neg = data['y'] == 0

    plt.figure(figsize=(8, 6))
    plt.scatter(data.loc[pos, 'X1'], data.loc[pos, 'X2'],
                marker='x', color='red',   label='Positive')
    plt.scatter(data.loc[neg, 'X1'], data.loc[neg, 'X2'],
                marker='o', color='blue',  label='Negative')

    if boundary:
        X = data[['X1', 'X2']].values
        y = data['y'].values

        x_min, x_max = X[:, 0].min()-0.5, X[:, 0].max()+0.5
```

```
        y_min, y_max = X[:, 1].min()-0.5, X[:, 1].max()+0.5
        xx, yy = np.meshgrid(
            np.linspace(x_min, x_max, 300),
            np.linspace(y_min, y_max, 300)
        )

        Z = svc.decision_function(
            np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

        plt.contour(xx, yy, Z, levels=[0],
                    linestyles=['-'], colors='lightblue')

    plt.legend()
    plt.show()


plot_svm_soft()
```



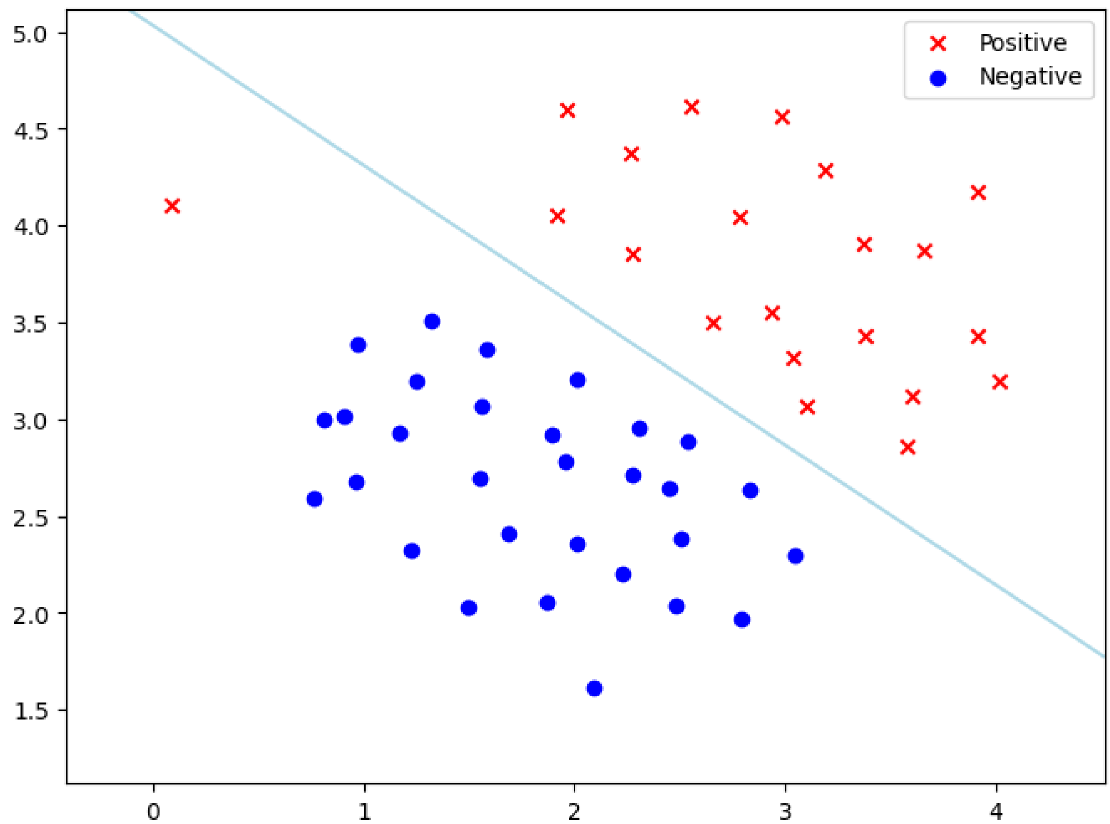


[2]:
```python
"""
Part 2: Classify the datapoints and visualize the decision boundary.
        You should print out the coefficients and the interception for the␣
 ↪linear classifier.
        Your plot should look like the sample one.
"""
from sklearn import svm
import numpy as np


def classify_svm_soft():
    raw_data = loadmat('data/svm_soft.mat')
    data = pd.DataFrame(raw_data['X'], columns=['X1', 'X2'])
    data['y'] = raw_data['y']

    svc = svm.LinearSVC(C=1, loss='hinge', max_iter=1_000_000)
    svc.fit(data[['X1', 'X2']], data['y'])

    print('Coefficients (w):', svc.coef_[0])
    print('Intercept (b):', svc.intercept_[0])

    plot_svm_soft(boundary=True, svc=svc)

    return svc


classify_svm_soft()
```

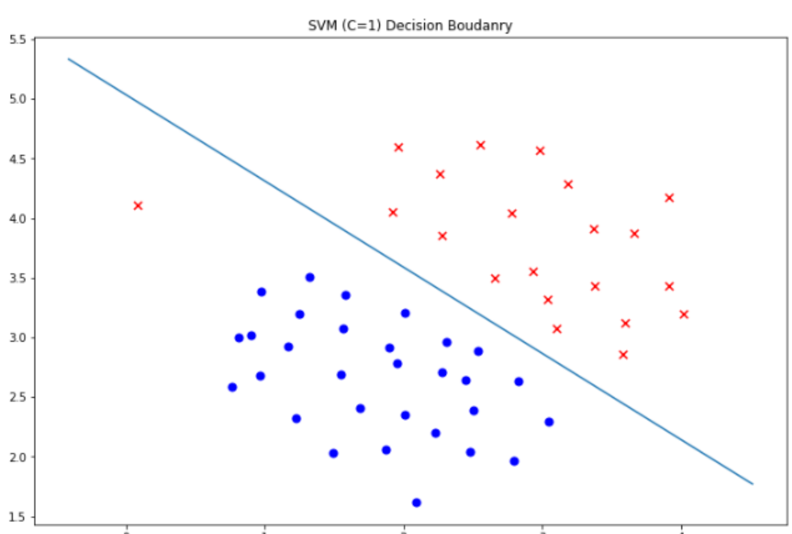
```
Coefficients (w): [0.59153686 0.81825054]
Intercept (b): -4.11954105475478
```

```
/opt/homebrew/anaconda3/lib/python3.11/site-
```

```
packages/sklearn/utils/validation.py:2749: UserWarning: X does not have valid
feature names, but LinearSVC was fitted with feature names
  warnings.warn(
```



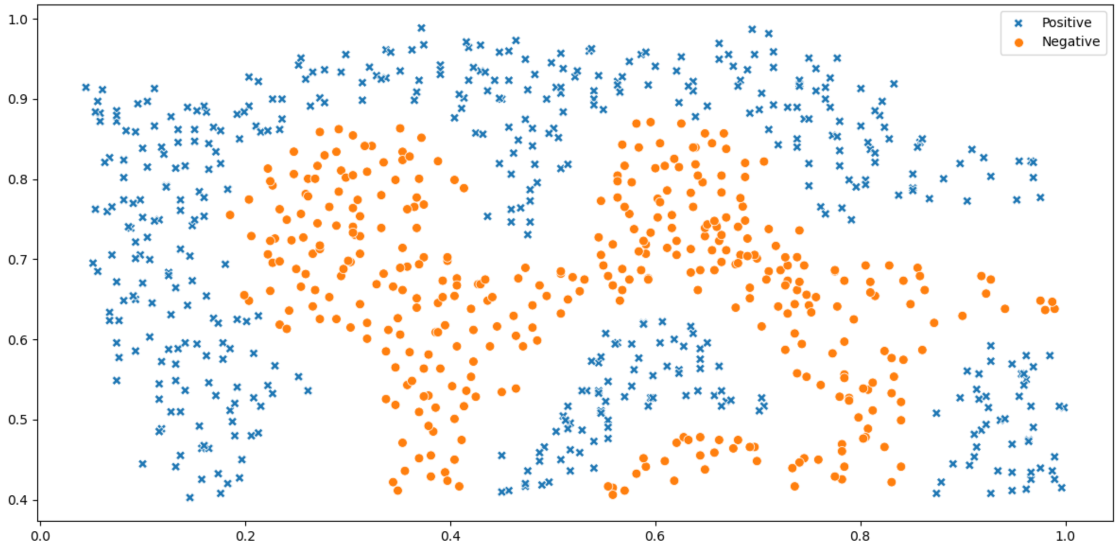
[2]: LinearSVC(C=1, loss='hinge', max_iter=1000000)

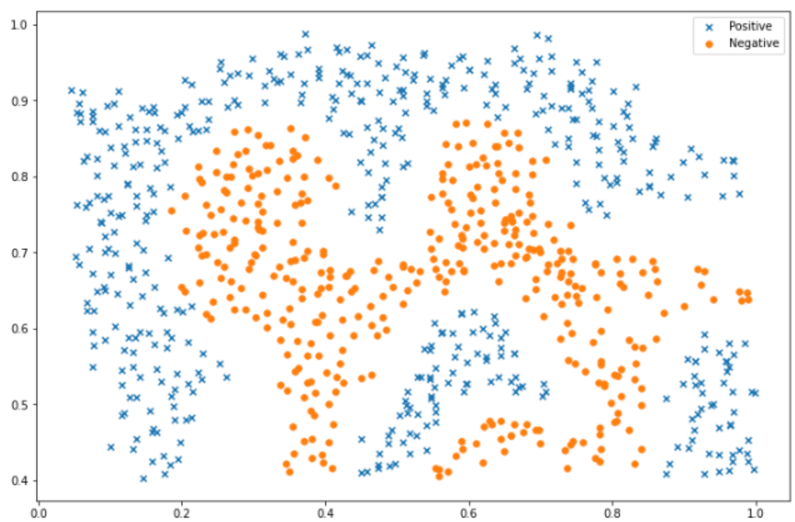


1.0.2 Q2: Kernel SVM (9 pts)

You are given a dataset (svm_kernel.mat) containing 2D points ['X'] from two classes with corresponding label (0 or 1) in column ['y']. Please visualize the datapoints and classify them using linear SVM with Gaussian kernel.

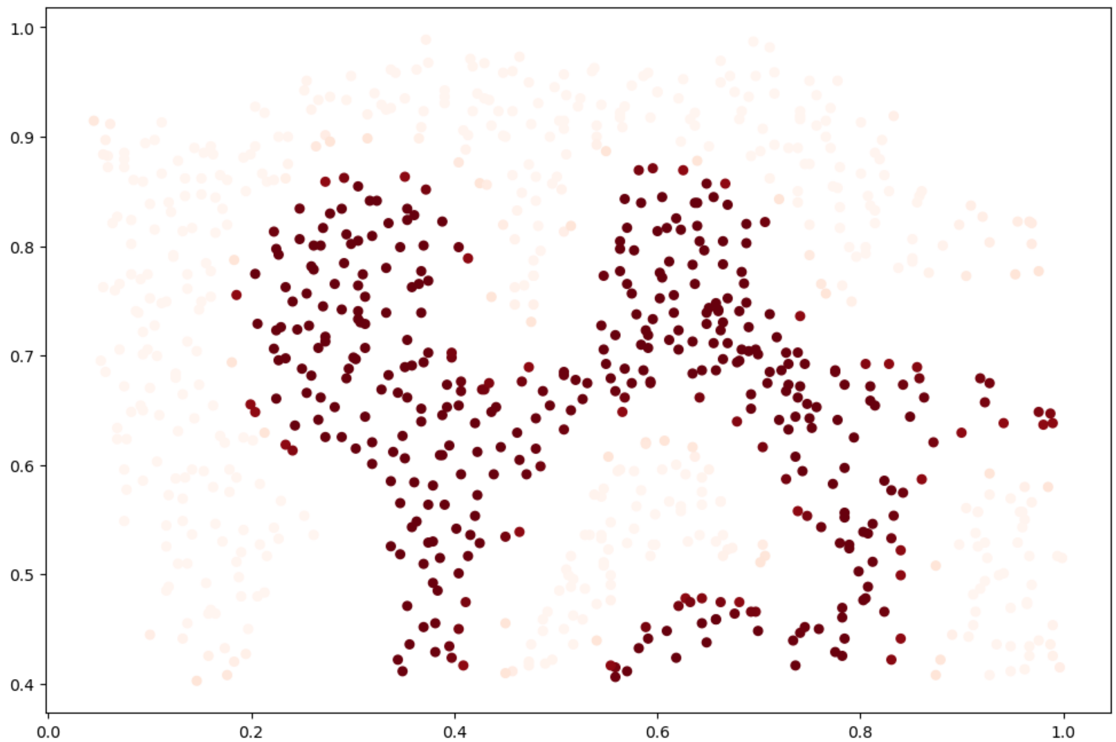
[8]:
```
"""
Part 1: Visualize the datapoints. Your plot should look similar to the sample␣
↪one.
"""

import pandas as pd
import matplotlib.pyplot as plt
from scipy.io import loadmat
import seaborn as sb

def plot_svm_kernel():
    global data_svm_kernel

    data_svm_kernel = loadmat('data/svm_kernel.mat')
    X2 = data_svm_kernel['X']
    y2 = data_svm_kernel['y'].ravel()

    data = pd.DataFrame(data_svm_kernel['X'], columns=['X1', 'X2'])
    data['y'] = data_svm_kernel['y']

    plt.figure(figsize=(12, 6))
    sb.scatterplot(
        data=data,
        x='X1', y='X2',
        hue='y',
        style='y',
        palette={1: '#1f77b4', 0: '#ff7f0e'},
        markers={1: 'X', 0: 'o'},
        s=50,
    )
    plt.xlabel('')
    plt.ylabel('')
    plt.legend(labels=['Positive', 'Negative'])
    plt.tight_layout()
    plt.show()


plot_svm_kernel()
```

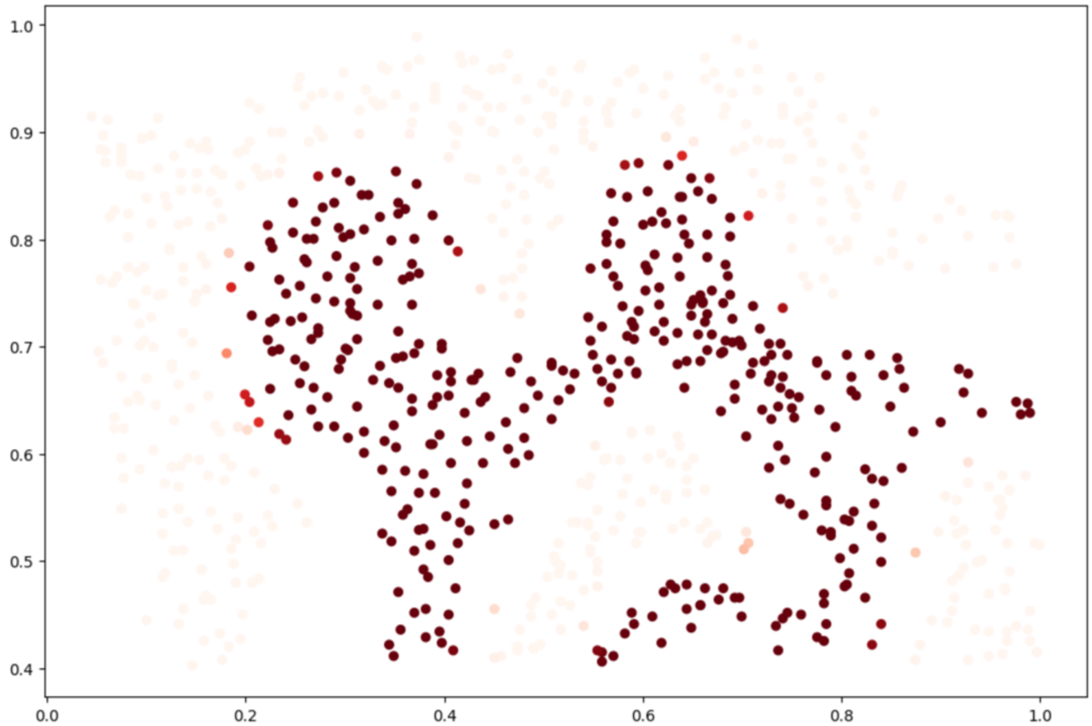




[6]:
```
"""
Part 2. Classify the data points using Gaussian kernel.
        Visualize the probabilities of the classification results.
        If your classification is correct, your plot should look similar to the␣
 ↪sample one.
"""



def classify_svm_kernel():
    global data_svm_kernel
    data = loadmat('data/svm_kernel.mat')
    X = data['X']
```

```
    y = data['y'].flatten()
    data_svm_kernel = {
        'X1': X[:, 0],
        'X2': X[:, 1]
    }
    svc = svm.SVC(kernel='rbf', C=1000, gamma=100, probability=True)
    svc.fit(X, y)
    probabilities = svc.predict_proba(X)[:, 0]
    return probabilities


probabilities = classify_svm_kernel()

fig, ax = plt.subplots(figsize=(12, 8))
ax.scatter(data_svm_kernel['X1'], data_svm_kernel['X2'],
           s=30, c=probabilities, cmap='Reds')
plt.show()
```





1.0.3 Q3: SVM in Email Spam Detection (9 pts)

Many email services today provide spam filters that are able to classify emails into spam and non-spam email with high accuracy. Please write an SVM classifier to detect spam emails. There are two data files that you need to load, one for training, i.e., "email_train.mat", and the other for testing, i.e., "email_test.mat". Your model should train on the training dataset and be applied to the test dataset to obtain the spam detection accuracy. **Your classifier should have an accuracy higher than 90% in both training and testing. Otherwise, every 10% of accuracy decrease leads to a 2 pts deduction in marks. For example, an accuracy of [80%, 90%) will get deducted for 2 pts overall, and [70%, 80%) will get deducted for 4 pts.**

[ ]:
```
from scipy.io import loadmat


def spam_detector_svm():
    train_data = loadmat('data/email_train.mat')
    X = train_data['X']
    y = train_data['y'].flatten()

    test_data = loadmat('data/email_test.mat')
    Xtest = test_data['Xtest']
    ytest = test_data['ytest'].flatten()

    svc = svm.SVC(kernel='rbf', C=1000, gamma='scale')
    svc.fit(X, y)
```

```
    # Do not remove.
    print('Training accuracy = {0}%'.format(
        np.round(svc.score(X, y) * 100, 2)))
    print('Test accuracy = {0}%'.format(
        np.round(svc.score(Xtest, ytest) * 100, 2)))


spam_detector_svm()
```

```
Training accuracy = 100.0%
Test accuracy = 98.0%
```

1.0.4 Q4: Bagging (9 pts)

You are given a tumor dataset `tumor.csv`. The dataset contains instances with 9 features. Each instance is labeled in either benign or malignant classes (0 for benign, 1 for malignant in last column). The dataset only contains numeric values and has been normalized. Use `RandomForestClassifier` to calculate the mean accuracy by using 8-folds cross-validation. Number of trees set to 50.

[ ]:
```
import numpy as np
import pandas as pd
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier


def classify_bagging():
    rand_seed = 50  # Do not change

    df = pd.read_csv('data/tumor.csv')

    X = df.iloc[:, :-1].values
    y = df.iloc[:, -1].values

    rf = RandomForestClassifier(n_estimators=50, random_state=rand_seed)
    results = model_selection.cross_val_score(rf, X, y, cv=8)

    # Do not remove.
    print("Performance of random forest:%f" % (results.mean()))


classify_bagging()
```

```
Performance of random forest:0.965713
```

1.0.5 Q5: AdaBoost (9 pts)

Use AdaBoostClassifier to classify the same dataset `tumor.csv`. Compute the mean accuracy by using 8-folds cross-validation and 50 estimators.

[ ]:

```python
from sklearn.ensemble import AdaBoostClassifier


def classify_adaboost():
    rand_seed = 50  # Do not change

    df = pd.read_csv('data/tumor.csv')
    X = df.iloc[:, :-1].values
    y = df.iloc[:, -1].values

    ada = AdaBoostClassifier(n_estimators=50, random_state=rand_seed)
    cv = StratifiedKFold(n_splits=8, shuffle=True, random_state=rand_seed)
    results = cross_val_score(
        ada, X, y, cv=cv, scoring='accuracy', n_jobs=-1)

    # Do not remove.
    print("Performance of adaboost:%f" % (results.mean()))


classify_adaboost()
```

```
Performance of adaboost:0.954219
```

[ ]:

```python
```