Linear Algebra Final Report Code Implementation

Hung-Ta, Wang | Student ID: R13546017 | HW_ID: 56

In [52]:
```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.datasets import make_moons
```

Example 1: Titanic from Kaggle

Example 1: Data Preprocessing

In [53]:
```
boat = pd.read_csv("titanic.csv")
print("Step 1: Clear categorical features with too many different values")
print("")
Ticket_list = []
for i in boat["Ticket"]:
    if i not in Ticket_list:
        Ticket_list.append(i)
print("Types of Ticket: ", len(Ticket_list))

Name_list = []
for i in boat["Name"]:
    if i not in Name_list:
        Name_list.append(i)
print("Types of Name: ", len(Name_list))

PassengerId_list = []
for i in boat["PassengerId"]:
    if i not in PassengerId_list:
        PassengerId_list.append(i)
print("Types of PassengerId: ", len(PassengerId_list))

columns_to_drop = ["Ticket", "Name", "PassengerId"]

print("")
print("If we encode Ticket, Name, PassengerId, the matrix would be too sparse to analyze.")
print("To prevent from being costly or overfitting, we delete them.")
print("-----------------------------------------------------------------------------------------------------")
print("Step 2: Clear features with too many NaN values")
print("")

Cabin_counter = 0
for i in boat["Cabin"].isna():
    if i:
        Cabin_counter += 1
print("The number of NaN values in Age: ", Cabin_counter)

Age_counter = 0
for i in boat["Age"].isna():
    if i:
        Age_counter += 1
print("The number of NaN values in Age: ", Age_counter)

columns_to_drop = ["Ticket", "Name", "PassengerId", "Cabin"]

print("")
print("Since the data in Cabin is too sparse to provide useful information, we delete it.")
print("To complete Age and Fare, We use a simple method: fill the median in.")
boat["Age"] = boat["Age"].fillna(boat["Age"].median())
boat["Fare"] = boat["Fare"].fillna(boat["Fare"].median())

boat.drop(columns=columns_to_drop, inplace=True)

print("-----------------------------------------------------------------------------------------------------")

print("Step 3: Transform categorical data into one-hot encoding form")
print("")
print("We transform Sex, Pclass and Embarked into one-hot encoding form, and turn it into float32")
boat = pd.get_dummies(boat, columns=["Sex", "Pclass", "Embarked"], prefix=["Sex", "Pclass", "Embarked"])
boat = boat.astype("Float32")
print("-----------------------------------------------------------------------------------------------------")
print("Final result is shown below: ")
print("")
boat

# boat.to_csv("LA.csv", index=None)
```

```
Step 1: Clear categorical features with too many different values

Types of Ticket:  363
Types of Name:  418
Types of PassengerId:  418

If we encode Ticket, Name, PassengerId, the matrix would be too sparse to analyze.
To prevent from being costly or overfitting, we delete them.
-----------------------------------------------------------------------------------------------
Step 2: Clear features with too many NaN values

The number of NaN values in Age:  327
The number of NaN values in Age:  86

Since the data in Cabin is too sparse to provide useful information, we delete it.
To complete Age and Fare, We use a simple method: fill the median in.
-----------------------------------------------------------------------------------------------
Step 3: Transform categorical data into one-hot encoding form

We transform Sex, Pclass and Embarked into one-hot encoding form, and turn it into float32
-----------------------------------------------------------------------------------------------
Final result is shown below:
```

Out[53]:

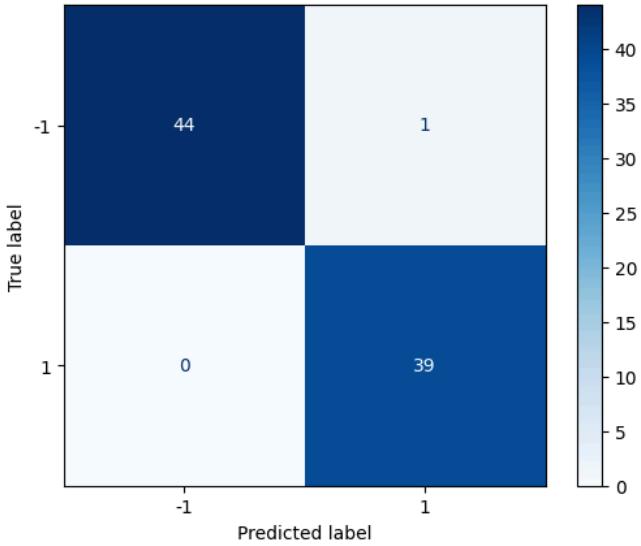
	Survived	Age	SibSp	Parch	Fare	Sex_female	Sex_male	Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarked_Q	Embarked_S
0	0.0	34.5	0.0	0.0	7.8292	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0
1	1.0	47.0	1.0	0.0	7.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
2	0.0	62.0	0.0	0.0	9.6875	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0
3	0.0	27.0	0.0	0.0	8.6625	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
4	1.0	22.0	1.0	1.0	12.2875	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
413	0.0	27.0	0.0	0.0	8.05	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
414	1.0	39.0	0.0	0.0	108.900002	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0
415	0.0	38.5	0.0	0.0	7.25	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
416	0.0	27.0	0.0	0.0	8.05	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
417	0.0	27.0	1.0	1.0	22.358299	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0

418 rows × 13 columns

Example 1: Use sklearn SVC packge

In [54]:
```
def make_cm(cm):
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[-1, 1])
    disp.plot(cmap=plt.cm.Blues)
    plt.show()
```

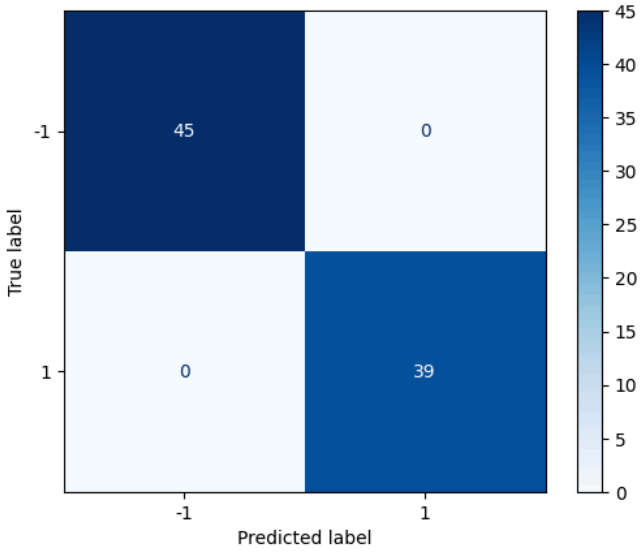
In [55]:
```
x_numerical = boat.iloc[:, 1:5].values
x_categorical = boat.iloc[:, 5:].values
y = boat.iloc[:, 0].values
y = np.where(y == 1, 1, -1)


scaler = StandardScaler()
x_numerical_transformed = scaler.fit_transform(x_numerical)
x = np.hstack((x_numerical_transformed, x_categorical))

x = x.astype(float)
y = y.astype(float)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

svm_model = SVC(kernel='rbf', C=1.0, random_state=0)
svm_model.fit(x_train, y_train)

y_pred = svm_model.predict(x_test)

cm = confusion_matrix(y_test, y_pred)

make_cm(cm)
```

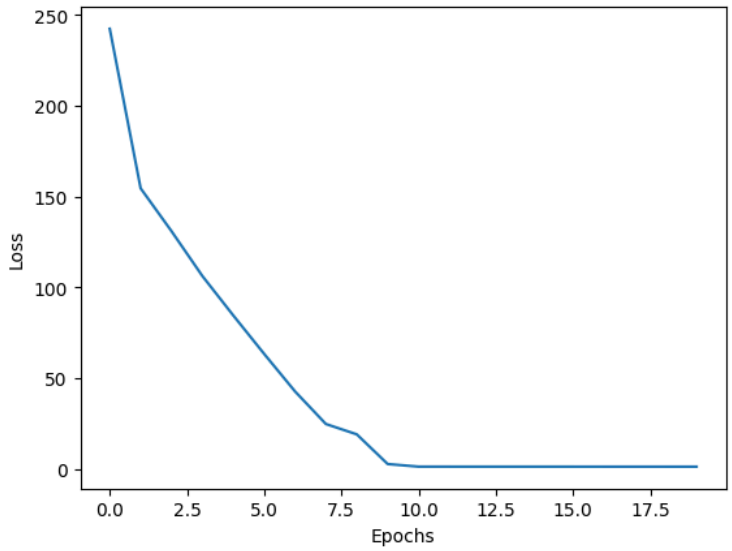


Example 1: SVM Implementation

In [56]:
```
def hinge_loss(x, y, w, b, C):
    N = len(y)
    loss = 0.5 * np.dot(w, w) + C * np.sum(np.maximum(0, 1 - y * (np.dot(x, w) + b)))
    return loss

def compute_gradient(x, y, w, b, C):
    N = len(y)
    grad_w = w.copy()
    grad_b = 0
    for i in range(N):
        margin = y[i] * (np.dot(x[i], w) + b)
        if margin < 1:
            grad_w -= C * y[i] * x[i]
            grad_b -= C * y[i]
    return grad_w, grad_b

def train_svm(x, y, C=1.0, learning_rate=0.01, epochs=1000):
    n_features = x.shape[1]
    w = np.zeros(n_features)
    b = 0
    losses = []
    for epoch in range(epochs):
        grad_w, grad_b = compute_gradient(x, y, w, b, C)
        w -= learning_rate * grad_w
        b -= learning_rate * grad_b
        loss = hinge_loss(x, y, w, b, C)
        losses.append(loss)
    return w, b, losses

def predict(x, w, b):
    return np.sign(np.dot(x, w) + b)

def plot_loss(epochs, losses):
    plt.plot(range(epochs), losses)
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.show()
```

In [57]:
```
x_numerical = boat.iloc[:, 1:5].values
x_categorical = boat.iloc[:, 5:].values
y = boat.iloc[:, 0].values
y = np.where(y == 1, 1, -1)

scaler = StandardScaler()
x_numerical_transformed = scaler.fit_transform(x_numerical)
x = np.hstack((x_numerical_transformed, x_categorical))

x = x.astype(float)
y = y.astype(float)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

C = 1.0
learning_rate = 0.001
epochs = 20
w, b, losses = train_svm(x_train, y_train, C, learning_rate, epochs)

y_pred = predict(x_test, w, b)

cm = confusion_matrix(y_test, y_pred)

make_cm(cm)
plot_loss(epochs, losses)
```





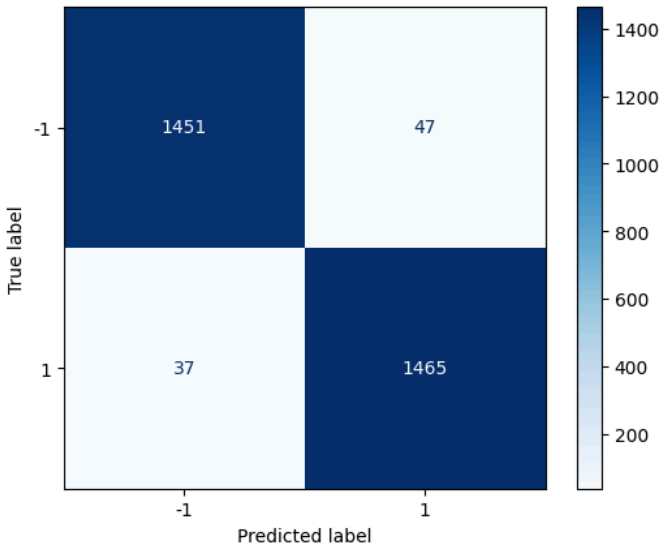
Example 2: Generated Moon Data

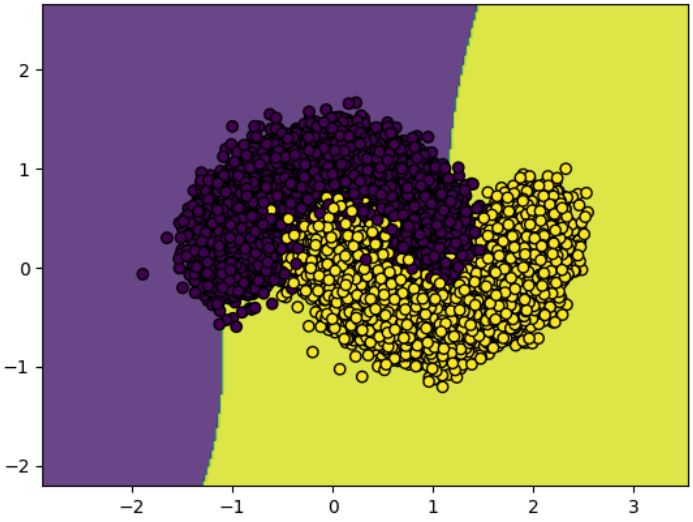
Example 2: Use sklearn SVC packge

In [58]:
```
def plot_decision_boundary_model(x, y, model):
    x_min, x_max = x[:, 0].min() - 1, x[:, 0].max() + 1
    y_min, y_max = x[:, 1].min() - 1, x[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300), np.linspace(y_min, y_max, 300))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.8, cmap="viridis")
    plt.scatter(x[:, 0], x[:, 1], c=y, edgecolor="k", cmap="viridis")
    plt.show()

def plot_decision_boundary_svm(x, y, w, b):
    x_min, x_max = x[:, 0].min() - 1, x[:, 0].max() + 1
    y_min, y_max = x[:, 1].min() - 1, x[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300), np.linspace(y_min, y_max, 300))
    Z = np.sign(np.dot(np.c_[xx.ravel(), yy.ravel()], w) + b).reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.8, cmap="viridis")
    plt.scatter(x[:, 0], x[:, 1], c=y, edgecolor="k", cmap="viridis", s=10)
    plt.show()
```

In [61]:
```
x, y = make_moons(n_samples=20000, noise=0.2, random_state=0)
y = np.where(y == 1, 1, -1)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15, random_state=0)

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

svm_model = SVC(kernel='rbf', C=1.0, random_state=0)
svm_model.fit(x_train, y_train)

y_pred = svm_model.predict(x_test)

cm = confusion_matrix(y_test, y_pred)
make_cm(cm)
```

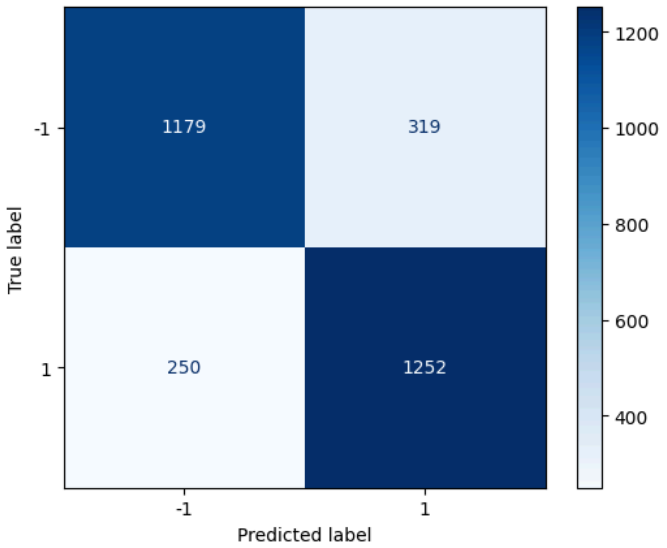
```
plot_decision_boundary_model(x, y, svm_model)
```

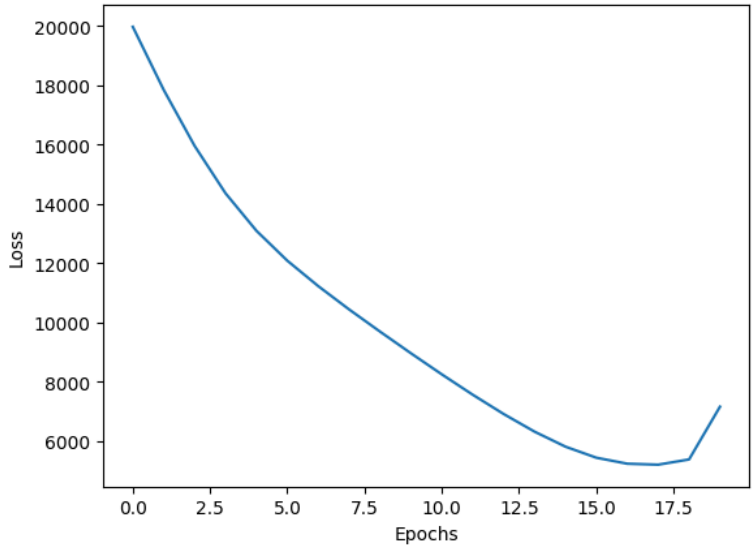


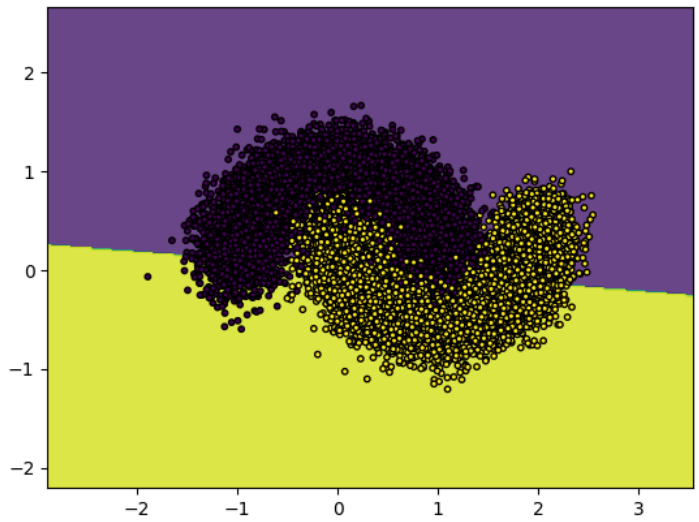


Example 2: SVM Implementation

In [60]:
```
x, y = make_moons(n_samples=20000, noise=0.2, random_state=0)
y = np.where(y == 1, 1, -1)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15, random_state=0)

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

C = 1.0
learning_rate = 0.001
epochs = 20
w, b, losses = train_svm(x_train, y_train, C, learning_rate, epochs)

y_pred = predict(x_test, w, b)
cm = confusion_matrix(y_test, y_pred)

make_cm(cm)
plot_loss(epochs, losses)

plot_decision_boundary_svm(x, y, w, b)
```


1179
319
250
1252
-1
1
True label
-1
1
Predicted label
1200
1000
800
600
400


20000
18000
16000
14000
12000
10000
8000
6000
Loss
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs


2
1
0
−1
−2
−2
−1
0
1
2
3