Exercise 8 - Support Vector Machines

Task 1

Show that the criterion

$$\delta_i(w^\intercal x_i + b) \geq 1, \quad i \in \{1, 2, \dots, n\}$$

corresponds to correct classification for all n samples in a binary classification problem where

$$\delta_i = \begin{cases} +1, & i \in \mathcal{I}_1, \\ -1, & i \in \mathcal{I}_2, \end{cases}$$

and $\mathcal{I}_k$ denotes the index set of class k.

Task 2

Given a binary data set:

$$\text{Class -1:} \begin{bmatrix} 1 & 9 \\ 5 & 5 \\ 1 & 1 \end{bmatrix}, \quad \text{Class +1:} \begin{bmatrix} 8 & 5 \\ 13 & 1 \\ 13 & 9 \end{bmatrix}.$$

Plot the points. Sketch the support vectors and the decision boundary for a linear SVM classifier with maximum margin for this data set.

Task 3

Given the binary classification problem:

$$\text{Class -1:} \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \\ 4 & 6 \\ 3 & 7 \\ 4 & 8 \\ 5 & 9 \\ 6 & 10 \end{bmatrix}, \quad \text{Class +1:} \begin{bmatrix} 6 & 2 \\ 7 & 3 \\ 8 & 4 \\ 9 & 5 \\ 8 & 6 \\ 7 & 7 \\ 7 & 8 \\ 7 & 9 \\ 8 & 10 \end{bmatrix}.$$

a)

b)

In the plot, sketch the mean values and the decision boundary you would get with a Gaussian classifier with covariance matrix $\Sigma = \sigma^2 I$, where I is the identity matrix.

c)

What is the error rate of the Gaussian classifier on the training data set?

d)

Sketch on the plot the decision boundary you would get using a SVM with linear kernel and a high cost of misclassifying training data. Indicate the support vectors and the decision boundary on the plot.

e)

What is the error rate of the linear SVM on the training data set?

Task 4

a)

Download the two datasets mynormaldistdataset.mat and mybananadataset.mat.

You can use a library for SVM, e.g. scikit-learn in python.

```python
from sklearn import svm

# In this case we assume 2d features.
# X: numpy array with shape (n, 2), all n data points.
# Y: numpy array of shape (n), labels in {0, 1, 2, 3, ...} corresponding to X

classifier = svm.SVC(C=10, kernel='linear')
classifier.fit(X, Y)

print("The feature [1, 2] is classified as:", classifier.predict([[1, 2]]))
```

Familiarize yourself with the data sets by studying scatterplots.

b)

Rerun the experiments a couple of times, and visualize the data using something like the following:

```
import numpy as np
import matplotlib.pyplot as plt

def make_meshgrid(X, h=.02):
    """Make a meshgrid covering the range of X. This will be used to draw cla

    Args:
        X: numpy array with shape [n, 2] containing 2d feature vectors.
        h: parameter controlling the resolution of the meshgrid
    """
    x = X[:, 0]
    y = X[: ,1]
    x_min, x_max = x.min() - 1, x.max() + 1
    y_min, y_max = y.min() - 1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max,
    return xx, yy

def scatter(X, Y, xx, yy, Z):
    """
    Scatter plot with classification regions

    Args:
        X: numpy array of shape [n, 2] where n is the total number of datapoi
        Y: numpy array of shape [n] containing the labels {1, 2, 3, ...} of X
        xx: meshgrid x
        yy: meshgrid y
        Z: The result of applying some prediction function on all points in x
    """
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

    plt.figure()
    # Color class regions
    plt.gca().contourf(xx, yy, Z, alpha=0.7)
    # Data points
    plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.viridis, marker='o', edgec
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
    plt.axes().set_aspect('equal')
    plt.grid()
    plt.tight_layout()

    plt.show()

# Given X and Y as explained above, we can display the data and classificatio
classifier = svm.SVC(C=100.0, kernel='linear')
classifier.fit(X, Y)

xx, yy = make_meshgrid(X)
Z = classifier.predict(np.c_[xx.ravel(), yy.ravel()])
```

How does the support vectors and the boundary change with the parameter?

c)

Try to remove some of the non-support-vectors and rerun. Does the solution change?

d)

Load `mybananadataset.mat`. Try various values of the $C-$parameter with a linear SVM. Can the linear SVM classifier make a good separation of the feature space?

e)

Change kernel to a RBF (*Radial Basis Function*), and rerun. Try changing the σ-parameter.

In the lecture we defined the rbf kernel as

$$K(x, y) = \exp\{-\frac{1}{\sigma^2}||x - y||^2\}.$$

In `sklearn`, σ is expressed through a γ parameter, and the rbf kernel is given as

$$K(x, y) = \exp\{-\gamma||x - y||^2\}.$$

```python
# svm.SVC() with its default parameter values:
classifier = svm.SVC(C=1.0, kernel='rbf', gamma=1/num_features)
```

Make sure you know why we now get non-lenear decision boundaries.

f)

Implement a grid search of the C- and σ- parameters based on 10-fold cross-validation of the training data (the A-dataset). Find the best values of C and σ, retrain on the entire A–dataset, and the test on the B-dataset. Does the average 10-fold cross-validation estimate of the overall classification error match the result we get when testing on the independent B-dataset?

You can for example use the parameter ranges suggested in the lecture slides:

$$C \in \{2^{-5}, 2^{-3}, 2^{-1}, \ldots, 2^{13}, 2^{15}\}$$
$$\sigma \in \{2^{-15}, 2^{-13}, 2^{-11}, \ldots, 2^{1}, 2^{3}\}.$$

-