VE 445 2019 Spring Lab 1 Support Vector Machine

I. Introduction

Support vector machine is a kind of linear classifier applied to solve binary classification problem. It has several modification models including soft margin SVM and kernel SVM to solve soft margin and linear non-separable problem. In this lab, we will implement the SVM and its modification models based on tensorflow library of python. This lab will help you understand SVM in the implementation perspective and get to know some powerful tool functions in tensorflow

II. SVM

1. **Hyperplane:**

   A hyperplane in an N-dimension Euclidean space is defined as following:

   $$\omega_1 x_1 + \omega_2 x_2 \cdots \omega_n x_n + b = 0$$

2. **SVM:**

   Given sample space $S = \{(\mathbf{x_1}, y_1)(\mathbf{x_2}, y_2), \cdots (\mathbf{x_m}, y_m), y_i = \{-1, 1\}\}$, we hope to find a hyperplane to divide the samples into two parts.

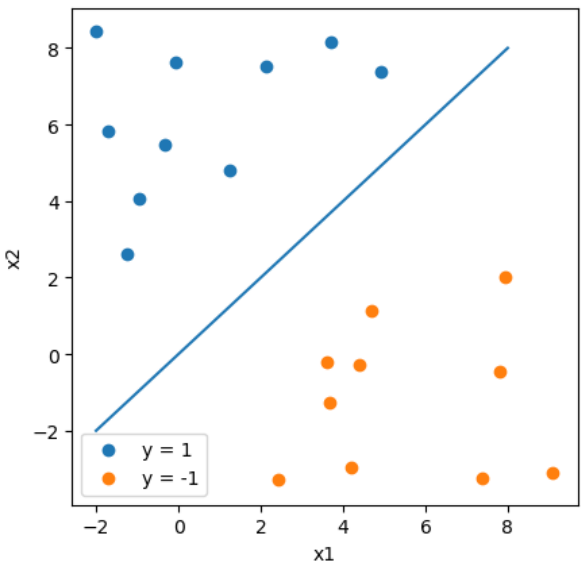

Figure 1. A division by hyperplane

Here the hyperplane can be defined as $\omega^{\mathbf{T}}\mathbf{x} + b = 0$, where $\omega = (\omega_1, \cdots, \omega_d)$, and b is the displacement. Assuming that the hyperplane (ω, b) can classify the sample correctly, then for each sample $(\mathbf{x_i}, y_i) \in S$, we have:

$$\begin{cases} \omega^T \mathbf{x} + b \geq 1, y_i = 1 \\ \omega^T \mathbf{x} + b \leq -1, y_i = -1 \end{cases}$$

The samples which make the equation achieved are called support vectors.

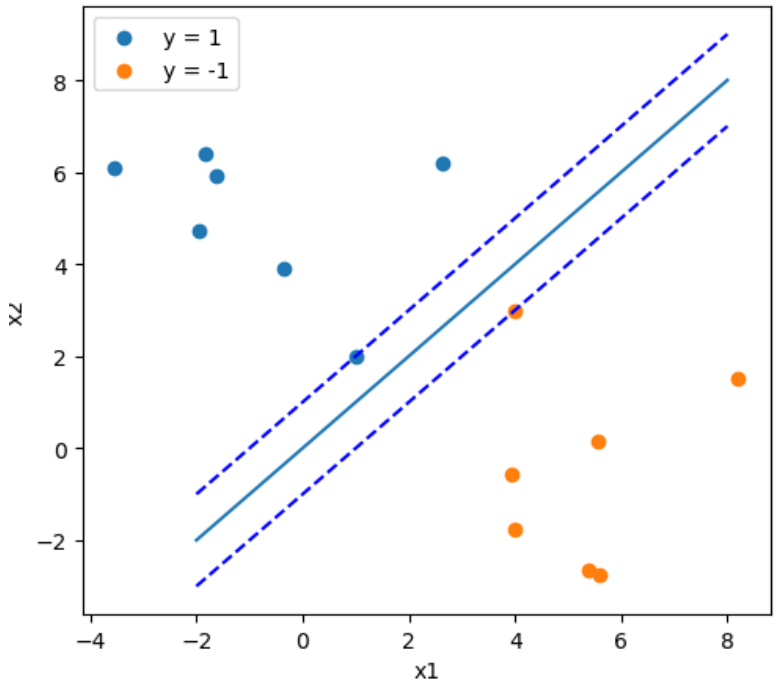

Figure 2. A margin to divide a sample space

The margin of the hyperspace is defined as the Euclidean distance between the hyperspace and two kinds of support vectors, which is given by:

$$\gamma = \frac{|\omega^T \mathbf{x} + b|_{y=1}}{||\omega||} + \frac{|\omega^T \mathbf{x} + b|_{y=-1}}{||\omega||} = \frac{2}{||\omega||}$$

Therefore the target function and constraint condition can be represented by:

$$\min_{\omega,b} \frac{1}{2}||\omega||^2 \qquad s.t. \quad y_i(\omega^T \mathbf{x_i} + b) \geq 1, \quad i = 1, 2, \cdots m.$$

By applying Lagrange multiplier method, its Lagrange function is given by:

$$\min_{\omega,b,\alpha} L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\omega^T \mathbf{x} + b)), \quad \alpha_i \geq 0, \quad i = 1, \cdots, m$$

Set the derivation of the Lagrange function to be 0:

$$\omega = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x_i}, \quad \sum_{i=1}^{m} \alpha_i y_i = 0$$

The dual problem can be hence obtained:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x_i^T x_j}$$

$$s.t. \quad \sum_{i=1}^{m} \alpha_i y_i = 0, \quad \alpha_i \geq 0, i = 1, 2, \cdots, m.$$

Using SMO algorithm, the function containing α can be optimized and the ω and b can be obtained by:

$$\omega = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x_i}, \quad y_s(\omega^T \mathbf{x_s} - b) \geq 1$$

where y_s and $\mathbf{x_s}$ is the label and feature of any support vector.

3. **Kernel SVM and Linear Non-separable Problem**:

   Sometimes in the original sample space it is impossible to find a hyperplane to divide the samples, which is known as linear non-separable.

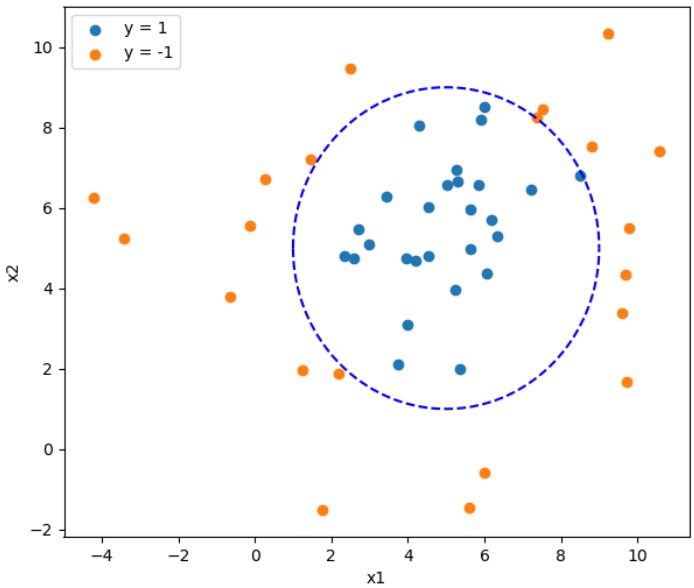

Figure 3. Linear Non-separable

For such case, a function $\phi(\mathbf{x})$ is expected to transfer the linear non-separable problem into a linear separable one, which maps the samples from lower dimension to higher dimensional Euclidean space, such that:

$$f(\mathbf{x}) = \omega^T \phi(\mathbf{x}) + b$$

And the target problem becomes:

$$\min_{\omega,b} \frac{1}{2}||\omega||^2 \qquad s.t. \quad y_i(\omega^T \phi(\mathbf{x_i}) + b) \geq 1, \quad i = 1, 2, \cdots m.$$

And the dual problem becomes:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x_i})^T \phi(\mathbf{x_j})$$

Here a kernel function is expected to simplify the inner product of higher dimension function $\phi(\mathbf{x})$, where $k(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})^T \phi(\mathbf{x_j})$. Regardless of what $\phi(\mathbf{x})$ exactly is, we typically choose kernel function in the following:

Type	Expression	parameter
Linear kernel function	$k(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i}^T \mathbf{x_j}$	
Polynomial kernel function	$k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i}^T \mathbf{x_j})^d$	$d \geq 1$
Gaussian kernel function	$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\|\|\mathbf{x_i} - \mathbf{x_j}\|\|/2\sigma^2)$	$\sigma > 0$
Laplace kernel function	$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\|\|\mathbf{x_i} - \mathbf{x_j}\|\|/\sigma)$	$\sigma > 0$

Table 1. Common kernel functions

And the original problem can be represented as:

$$f(\mathbf{x}) = \omega^T \phi(\mathbf{x}) + b = \sum_{i=1}^{m} \alpha_i y_i k(\mathbf{x}, \mathbf{x_i}) + b,$$

according to which the sample can be classified.

In the previous example, the following sample space can be generated by applying Gaussian function.

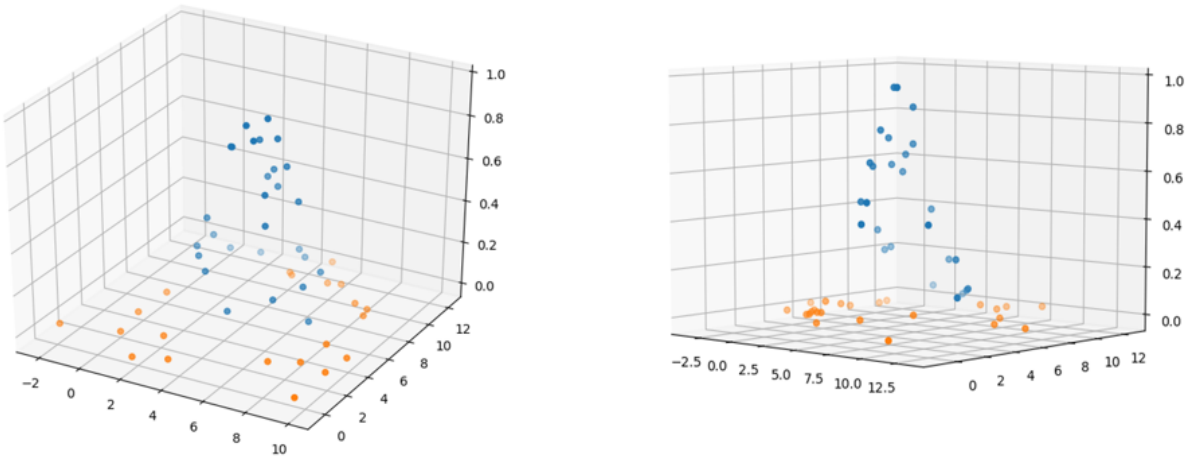

Figure 4. Higher dimensional space with Gaussian kernel function

4. **Soft Margin SVM**

   A soft margin SVM allows some samples that do not satisfy the hard margin constraint:

   $$y_i(\omega^T \mathbf{x_i} + b) \geq 1, i = 1, 2, \cdots, m.$$

   In order to maximize the margin and minimize the unsatisfactory sample number, the optimization function can be represented as:

   $$\min_{\omega,b} \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{m} l(y_i(\omega^T \mathbf{x_i} + b) - 1) := \min_{\omega,b} \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{m} \xi_i$$

   where $l(x)$ is a loss function and C is a constant. In this lab, we use the loss function:

   $$l(x) = \max(0, 1 - x)$$

   Its Lagrange function is given by:

   $$L(\omega, b, \alpha, \xi, \mu) = \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{m} \xi_i + \sum_{i=1}^{m} \alpha_i(1 - \xi_i - y_i(\omega^T \mathbf{x_i} + b)) - \sum_{i=1}^{m} \mu_i \xi_i$$

   Set the derivation of the Lagrange function to be 0:

   $$\omega = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x_i}, \quad \sum_{i=1}^{m} \alpha_i y_i = 0, \quad C = \alpha_i + \mu_i$$

   The dual problem is given by:

   $$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x_i^T x_j}$$

   $$s.t. \quad \sum_{i=1}^{m} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, i = 1, 2, \cdots, m.$$

It can be seen that the only difference between soft margin SVM and hard margin SVM is just the constraint of α . Similarly, this dual problem can be easily solved by SMO algorithm and any other optimization algorithm.

III. Specifics

There are 100 points for this lab in total. And a template python file is given. You should follow the specifics in the file and implement the three classes. Some of the function is given and not allowed to modify. There are several sets of arbitrary data and you should train different models based on them.

[Tips:] Your don't need to implement SMO algorithm on your own because the implementation might be exhausting and tedious. Instead, you can use the optimizer of Tensorflow. However, if you implement the optimize algorithm on your own, some bonus will be considered for you.

1. [20%] Implement the class SVM.
2. [10%] Implement the class KernelSVM.
3. [10%] Implement the class SoftMarginSVM.
4. [10%] Train the model based on data1 using basic SVM.
5. [10%] Train the model based on data2 using kernel SVM.
6. [10%] Train the model based on data3 using soft margin SVM.
7. [20%] Write report about your training procedure and compare your result with the result by using library function provided by **sklearn** module or other python modules.
8. [10%] Train your SVM based on a big data set.

IV. Implementation.

The implementation environment is suggested to be Python(version 3.5.2 +). Considering that some of students might not be familiar with tensorflow, thus some examples will be given here to help you implement the lab.

1. Tensorflow is an open source library for python and you can easily install it using pip3 on Linux or Mac OS, or Anaconda3 for Windows OS.
2. In tensorflow, all the data and parameters being calculated and operated are stored as tensors.

   For example:

   ```
   import numpy as np
   import tensorflow as tf                                  # import the module
   A = tf.Variable(tf.random_normal(shape=[5, 1]))          # define a random variable
   sess = tf.Session()                                      # create a session
   sess.run(tf.global_variables_initializer())              # initialize variables
   print(sess.run(A))                                       # run the session
   ```

   The result is given by:

   ```
   [[-1.527929  ]
    [-1.2407476 ]
    [-1.1900425 ]
    [-0.3022918 ]
    [-0.04236453]]
   ```

The data can be represented as placeholder in Tensorflow, and **feed_dict** can be used to feed the data in the session of your tensor graph:

```
x = tf.placeholder(shape=[None, 5], dtype=tf.float32)   # define a place holder
                                                        # with 5 features
result = tf.matmul(x, A)                                # define result = x * A
B = np.array([[1, 2, 3, 4, 5]])                         # given data B
print(sess.run(result,feed_dict = {x:B}))               # feed the session and run
```

The output is the $xA =$ [[-9.000542]]

Apart from the basic examples above, you can also apply some other functions or classes in tensorflow such as **tf.train, tf.reduce_mean, tf.reduce_sum, tf.subtract, tf.add**. Your could get the usage and example from the Tensorflow official website: https://tensorflow.google.cn/api_docs/python/.

V. Submission

You should submit a .zip file to Canvas by April 3rd 11:59 pm, which is expected to contain the python file named SVM.py and your lab report.