- Execution description:
  在 colab 按全部執行就可以了
- Experimental results

1. Use the linear model y = 2x +ε with zero-mean Gaussian noise ε~ N(0, 1)
to generate 500 data points with (equal spacing) x ∈ [−100, 100].

```
torch.manual_seed(35)

# y = 2x +ε
def generate_data_points(max, min, num_points):
  x = torch.linspace(max, min, num_points)
  epsilon = torch.randn(num_points)
  y = 2 * x + epsilon

  return x,y
```

```
x, y = generate_data_points(-100, 100, 500)
```

2. 訓練一個 RBF SVM

-t 2 代表用 radial basis function

```
# c = 2 to the power of -5
# gamma = 2 to the power of -15
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.000030517578125 -w1 1 -w-1 1 -v 5')
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.000030517578125 -w1 1 -w-1 1')
p_labels, p_acc, p_vals = svmutil.svm_predict(test_label, test, svm_model)
```

```
Cross Validation Accuracy = 83%
Accuracy = 84% (84/100) (classification)
```

3. 利用 5-fold cross validation，至少找三組 C and γ 參數，挑一組最好的

第一個：

```
# c = 2 to the power of -5
# gamma = 2 to the power of -15
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.000030517578125 -w1 1 -w-1 1 -v 5')
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.000030517578125 -w1 1 -w-1 1')
p_labels, p_acc, p_vals = svmutil.svm_predict(test_label, test, svm_model)
```

```
Cross Validation Accuracy = 83%
Accuracy = 84% (84/100) (classification)
```

第二個：

```
# c = 2 to the power of -5
# gamma = 2 to the power of -13
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.0001220703125 -w1 1 -w-1 1 -v 5')
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.0001220703125 -w1 1 -w-1 1')
p_labels, p_acc, p_vals = svmutil.svm_predict(test_label, test, svm_model)
```

```
Cross Validation Accuracy = 88.5%
Accuracy = 89% (89/100) (classification)
```

第三個：

```
# c = 2 to the power of -5
# gamma = 2 to the power of -11
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.00048828125 -w1 1 -w-1 1 -v 5')
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.00048828125 -w1 1 -w-1 1')
p_labels, p_acc, p_vals = svmutil.svm_predict(test_label, test, svm_model)
```

```
Cross Validation Accuracy = 89%
Accuracy = 90% (90/100) (classification)
```

第三個準確率最高！

4. 用上題找到的參數，比較 scaling 方法、或不用 scaling，影響準確率的差異為何。

沒有 scaling:

```
# c = 2 to the power of -5
# gamma = 2 to the power of -11
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.00048828125 -w1 1 -w-1 1 -v 5')
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.00048828125 -w1 1 -w-1 1')
p_labels, p_acc, p_vals = svmutil.svm_predict(test_label, test, svm_model)
```

```
Cross Validation Accuracy = 89%
Accuracy = 90% (90/100) (classification)
```

有 scaling(scale 到 range [-1; +1]):

```
def scaling(data, max, min):
  x_tensor = torch.tensor(data[:,0]).clone().detach()
  # y_tensor = torch.tensor(data[:,1])
  # Linearly scaling x train to the range [-1; +1].

  x_current_min = torch.min(x_tensor)
  x_current_max = torch.max(x_tensor)

  x_new_min = min
  x_new_max = max

  x_scaling_factor = (x_new_max - x_new_min) / (x_current_max - x_current_min)

  x_shifted_tensor = x_tensor - x_current_min

  x_scaled_tensor = x_shifted_tensor * x_scaling_factor

  x_scale_data = x_scaled_tensor + x_new_min

  epsilon = data[:,1] - 2 * data[:,0]
  y_scale_data = 2 * x_scale_data + epsilon

  return  x_scale_data, y_scale_data

x_train_scale_data, y_train_scale_data = scaling(train, 1, -1)
x_train_scale_data = x_train_scale_data .unsqueeze(1)
y_train_scale_data = y_train_scale_data.unsqueeze(1)


x_test_scale_data, y_test_scale_data = scaling(test, 1, -1)
x_test_scale_data = x_test_scale_data .unsqueeze(1)
y_test_scale_data = y_test_scale_data.unsqueeze(1)

train = torch.cat((x_train_scale_data , y_train_scale_data), dim=1).numpy()
test = torch.cat((x_test_scale_data, y_test_scale_data), dim=1).numpy()
```

```
# c = 2 to the power of -5
# gamma = 2 to the power of -11
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.00048828125 -w1 1 -w-1 1 -v 5')
svm_model = svmutil.svm_train(train_label, train, '-s 1 -t 2 -d 2 -c 0.03125 -g 0.00048828125 -w1 1 -w-1 1')
p_labels, p_acc, p_vals = svmutil.svm_predict(test_label, test, svm_model)
```

```
Cross Validation Accuracy = 94.75%
Accuracy = 95% (95/100) (classification)
```

- Conclusion:
  1. 還沒有 scale 之前設-c 0.03125 -g 0.00048828125 會得到最好的準確率
  2. Scale 到 range [-1; +1]之後準確率會更高
- Discussion
  讀懂 github 的程式碼細節後，就會知道各個參數的意義，也更能調整出讓準確率更高的參數，然而要讀懂當中的細節有一定的困難度。