11077009 資訊碩一 林冠良 HW1 - libsvm

a. Using linear kernel and RGB color feature (400 × 3 dims for each image)

1. Use self-written txt_to_svm.py to convert the original feature values to svm file format.
2. Take the converted file and scale the data to be in [0, 1] using svm-scale.exe in libsvm-325/windows.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-scale.exe -l 0 -u 1
 ..\..\test\rgb_features > ..\..\test\rgb_features.scale
```

3. Use the scaled file as input and send it to svm-train.exe in the same directory.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-train.exe -t 0 ..\.
.\train\rgb_features.scale ..\..\train\rgb_linear.model
.......*.*
optimization finished, #iter = 2445
nu = 0.048622
obj = -7.009131, rho = -0.066248
nSV = 101, nBSV = 2
Total nSV = 101
```

4. Predict the test dataset using the model file from previous step and get an accuracy of 65.6527%.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-predict.exe ..\..\t
est\rgb_features.scale ..\..\train\rgb_linear.model ..\..\test\rgb_linear_predict
Accuracy = 65.6627% (109/166) (classification)
```

b. Using RBF kernel and RGB color feature

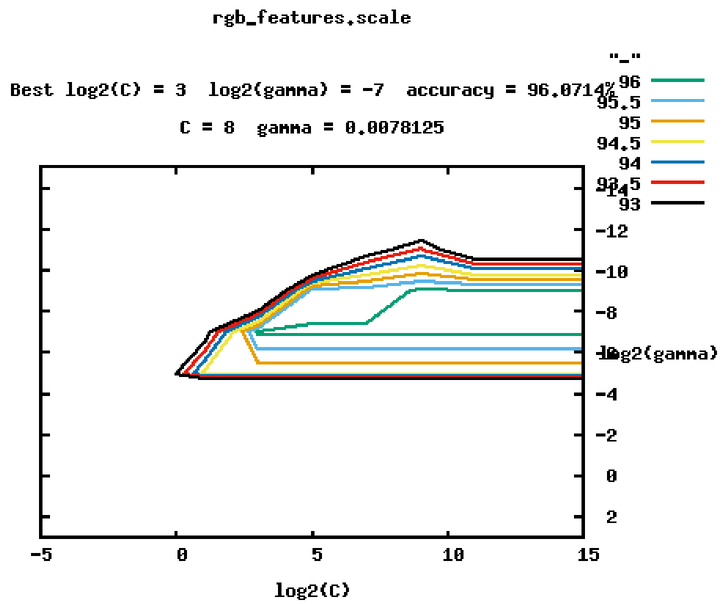
1. Use self-written txt_to_svm.py to convert the original feature values to svm file format.
2. Take the converted file and scale the data to be in [0, 1] using svm-scale.exe in libsvm-325/windows.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-scale.exe -l 0 -u 1
 ..\..\test\rgb_features > ..\..\test\rgb_features.scale
```

3. Use grip.py in the tools directory to find the best parameters.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> python ..\tools\grid.py -
out ../../train/rgb_RBF.out -png ../../train/rgb_RBF.jpg ..\..\train\rgb_features.sc
ale
[local] 5 -7 96.0714 (best c=32.0, g=0.0078125, rate=96.0714)
[local] -1 -7 87.1429 (best c=32.0, g=0.0078125, rate=96.0714)
[local] 5 -1 78.2143 (best c=32.0, g=0.0078125, rate=96.0714)
```

```
8.0 0.0078125 96.0714
```



4. Take the best c and gamma from previous step and use it as parameters in the svm-train.exe command.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-train.exe -t 2 -c 8
.0 -g 0.0078125 ..\..\train\rgb_features.scale ../../train/rgb_RBF.model
.*
optimization finished, #iter = 392
nu = 0.128949
obj = -147.648805, rho = 0.910830
nSV = 150, nBSV = 5
Total nSV = 150
```

5. Predict the test dataset using the model file from previous step and get an accuracy of 76.506%

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-predict.exe ..\..\t
est\rgb_features.scale ..\..\train\rgb_RBf.model ../../test/rgb_RBF_predict
Accuracy = 76.506% (127/166) (classification)
```

c. Using RBF kernel and RGB color feature, 5-fold cross validation

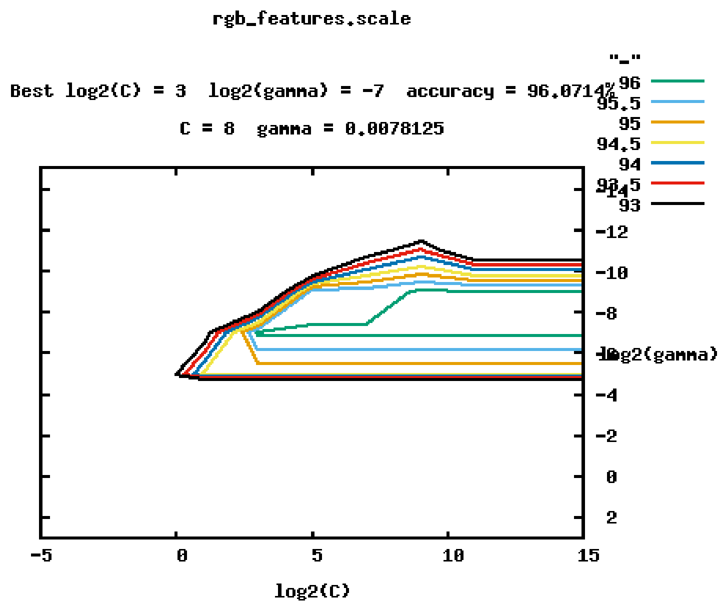
1. Use self-written txt_to_svm.py to convert the original feature values to svm file format.
2. Take the converted file and scale the data to be in [0, 1] using svm-scale.exe in libsvm-325/windows.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-scale.exe -l 0 -u 1
 ..\..\test\rgb_features > ..\..\test\rgb_features.scale
```

3. Use grip.py in the tools directory to find the best parameters.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> python ..\tools\grid.py -
out ../../train/rgb_RBF.out -png ../../train/rgb_RBF.jpg ..\..\train\rgb_features.sc
ale
[local] 5 -7 96.0714 (best c=32.0, g=0.0078125, rate=96.0714)
[local] -1 -7 87.1429 (best c=32.0, g=0.0078125, rate=96.0714)
[local] 5 -1 78.2143 (best c=32.0, g=0.0078125, rate=96.0714)
```

```
8.0 0.0078125 96.0714
```



4. Take the best parameters from previous step and add an option of 5-fold cross validation in the training step, in this situation, svm-train doesn't output a model, only a generalization performance.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-train.exe -t 2 -c 8
.0 -g 0.0078125 -v 5 ..\..\train\rgb_features.scale ../../train/rgb_RBF_5fold.model
.*
optimization finished, #iter = 366
nu = 0.136901
obj = -127.286185, rho = 0.974825
nSV = 135, nBSV = 7
Total nSV = 135
*.*
optimization finished, #iter = 328
nu = 0.134482
obj = -128.399837, rho = 0.850707
nSV = 128, nBSV = 6
Total nSV = 128
.*
optimization finished, #iter = 342
nu = 0.142434
obj = -129.970582, rho = 0.765601
nSV = 135, nBSV = 5
Total nSV = 135
.*
optimization finished, #iter = 357
nu = 0.143556
obj = -135.587044, rho = 0.706199
nSV = 129, nBSV = 7
Total nSV = 129
.*
optimization finished, #iter = 360
nu = 0.154023
obj = -142.836339, rho = 0.829481
nSV = 140, nBSV = 8
Total nSV = 140
Cross Validation Accuracy = 96.0714%
```

5. We can see that in the estimate generalization performance, the cross-validation accuracy is estimated to be 96.0714%.

d. Using linear kernel and gradient feature (400 2 dims for each image)

1. Use self-written txt_to_svm.py to convert the original feature values to svm

file format.

2. Take the converted file and scale the data to be in [0, 1] using svm-scale.exe in libsvm-325/windows.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-scale.exe -l 0 -u 1
 ..\..\test\gradient_features > ..\..\test\gradient_features.scale
```

3. Use the scaled file as input and send it to svm-train.exe in the same directory.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-train.exe -t 0 ..\.
.\train\gradient_features.scale ..\..\train\gradient_linear.model
.*
optimization finished, #iter = 462
nu = 0.010731
obj = -1.502291, rho = -1.819162
nSV = 134, nBSV = 0
Total nSV = 134
```

4. Predict the test dataset using the model file from previous step and get an accuracy of 77.7108%.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-predict.exe ..\..\t
est\gradient_features.scale ..\..\train\gradient_linear.model ..\..\test\gradient_li
near_predict
Accuracy = 77.7108% (129/166) (classification)
```

e. Using RBF kernel and gradient feature

1. Use self-written txt_to_svm.py to convert the original feature values to svm file format.

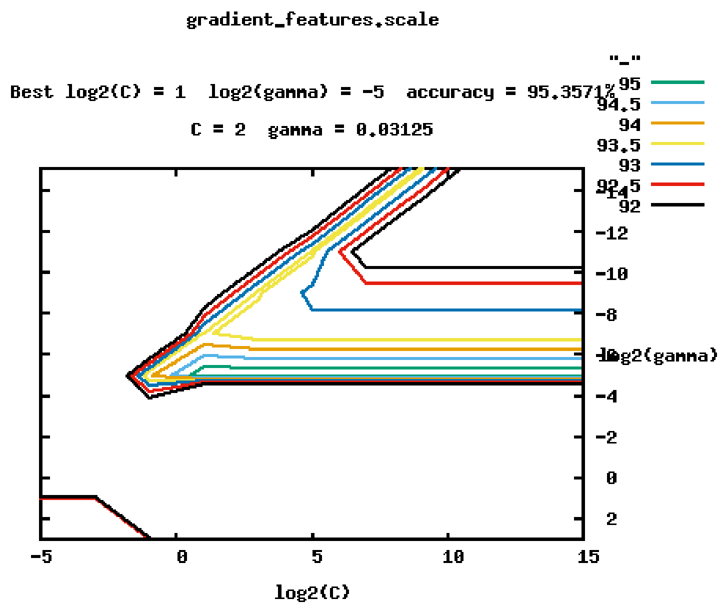
2. Take the converted file and scale the data to be in [0, 1] using svm-scale.exe in libsvm-325/windows.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-scale.exe -l 0 -u 1
 ..\..\test\gradient_features > ..\..\test\gradient_features.scale
```

3. Use grip.py in the tools directory to find the best parameters.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> python ..\tools\grid.py -
out ../../train/gradient_RBF.out -png ../../train/gradient_RBF.png ..\..\train\gradi
ent_features.scale
[local] 5 -7 93.2143 (best c=32.0, g=0.0078125, rate=93.2143)
[local] -1 -7 88.9286 (best c=32.0, g=0.0078125, rate=93.2143)
[local] 5 -1 76.7857 (best c=32.0, g=0.0078125, rate=93.2143)
```

```
2.0 0.03125 95.3571
```



4. Take the best c and gamma from previous step and use it as parameters in the svm-train.exe command.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-train.exe -t 2 -c 2
.0 -g 0.03125 ..\..\train\gradient_features.scale ..\..\train\gradient_RBF.model
*.*
optimization finished, #iter = 308
nu = 0.245009
obj = -68.602168, rho = 0.366688
nSV = 179, nBSV = 0
Total nSV = 179
```

5. Predict the test dataset using the model file from previous step and get an accuracy of 78.9157%

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-predict.exe ..\..\t
est\gradient_features.scale ..\..\train\gradient_RBF.model ../../test/gradient_RBF_p
dict
Accuracy = 78.9157% (131/166) (classification)
```

f. Using RBF kernel and gradient feature, 5-fold cross validation

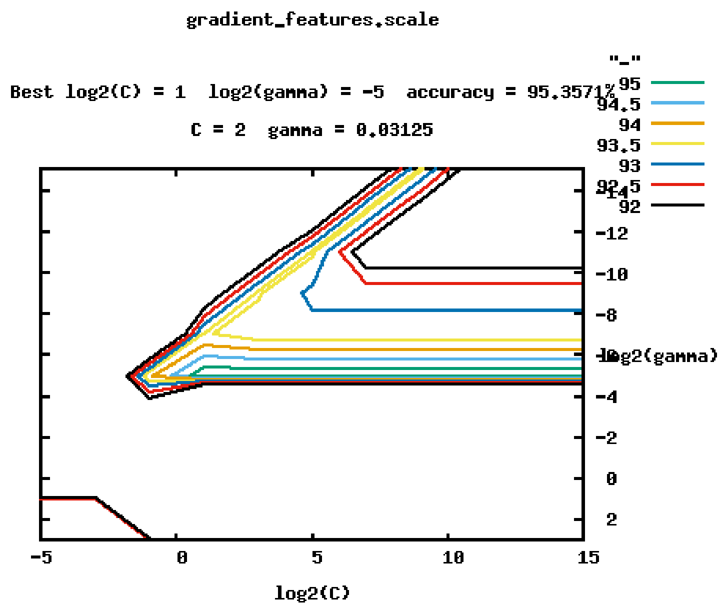
1. Use self-written txt_to_svm.py to convert the original feature values to svm file format.
2. Take the converted file and scale the data to be in [0, 1] using svm-scale.exe in libsvm-325/windows.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-scale.exe -l 0 -u 1
 ..\..\test\gradient_features > ..\..\test\gradient_features.scale
```

3. Use grip.py in the tools directory to find the best parameters.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> python ..\tools\grid.py -
out ../../train/gradient_RBF.out -png ../../train/gradient_RBF.png ..\..\train\gradi
ent_features.scale
[local] 5 -7 93.2143 (best c=32.0, g=0.0078125, rate=93.2143)
[local] -1 -7 88.9286 (best c=32.0, g=0.0078125, rate=93.2143)
[local] 5 -1 76.7857 (best c=32.0, g=0.0078125, rate=93.2143)
```

```
2.0 0.03125 95.3571
```



4. Take the best parameters from previous step and add an option of 5-fold cross validation in the training step, in this situation, svm-train doesn't output a model, only a generalization performance.

```
(base) PS C:\Users\popshia\Desktop\HW1\libsvm-325\windows> .\svm-train.exe -t 2 -c 2
.0 -g 0.03125 -v 5 ..\..\train\gradient_features.scale ..\..\train\gradient_RBF_5fol
d.model
*.*
optimization finished, #iter = 277
nu = 0.272818
obj = -61.110054, rho = 0.333772
nSV = 160, nBSV = 0
Total nSV = 160
*.*
optimization finished, #iter = 297
nu = 0.271695
obj = -60.858146, rho = 0.328159
nSV = 160, nBSV = 0
Total nSV = 160
*.*
optimization finished, #iter = 254
nu = 0.278536
obj = -62.392260, rho = 0.355260
nSV = 156, nBSV = 0
Total nSV = 156
*.*
optimization finished, #iter = 283
nu = 0.283284
obj = -63.455251, rho = 0.286191
nSV = 163, nBSV = 0
Total nSV = 163
*.*
optimization finished, #iter = 271
nu = 0.281738
obj = -63.107843, rho = 0.311386
nSV = 159, nBSV = 0
Total nSV = 159
Cross Validation Accuracy = 95.3571%
```

5. We can see that in the estimate generalization performance, the cross-validation accuracy is estimated to be 95.3571%.

Conclusion

As for the three training options, RBF with cross validation shows the best accuracy, the second accurate is RBF kernel, with linear kernel being the worst accurate model. Looking at feature choosing, gradient feature trained model perform better than RGB feature models.