Data Mining HW5

LIBSVM

Name: 張緣彩 Student ID: R07922141

5.1 Iris dataset: Testing label is provided.

a. Comparison of performance with and without scaling. [5%]

Kernel/Scaling	With scaling (-1, 1)	Without scaling
Linear	**100%**	**100%**
Polynomial	69.333%	98.6667%
RBF	97.3333%	97.3333%
Sigmoid	97.3333%	33.3333%

(Testing set accuracy)

From the table above, we can see that with or without scaling have the greatest performance which 100% accuracy on testing set. Moreover, scaling the dataset increased the performance on sigmoid kernel whereas hurting the performance of polynomial kernel.

b. Comparison of different kernel functions. [5%]

Kernel/Scaling	With scaling (-1, 1)	Without scaling
Linear	**100%**	**100%**
Polynomial	69.333%	98.6667%
RBF	97.3333%	97.3333%
Sigmoid	97.3333%	33.3333%

(Testing set accuracy)

From the table above, we can see that using linear kernel has the greatest performance in this dataset which 100% accuracy on testing set.

c. Parameter set and performance of your best model. (Report training accuracy and testing accuracy) [5%]

Kernel = Linear

Training accuracy = 98.6667%

Testing accuracy = 100%

I only change the change the kernel to linear, all other parameters are the default values used by LIBSVM.

d. More discussions are welcome. [Bonus 1%]

I found out that if you scale the data into [0,1.0] will hurting the performance, training accuracy 96%; testing accuracy 97.3333%.

5.2 News dataset: Testing label is provided.

a. Comparison of performance with and without scaling. [5%]

Kernel/Scaling	With scaling (0, 1)	Without scaling
Linear	80.4895%	**84.1958%**
Polynomial	27.7622%	27.7622%
RBF	27.7622%	27.7622%
Sigmoid	27.7622%	27.7622%

(Testing accuracy)

From the table above, we can see that the data without scaling will has better performance than with scaling. I also tried to scale to range [0, 2], the performance is worse than [0, 1]. So, we can conclude that without scaling has the best performance.

b. Comparison of 5-1-a and 5-2-a. [5%]

By comparing the dataset between 5-1 and 5-2, we are able to know that scaling works in 5-1 because when we scaled the data in 5-1, some kernels actually have improved its performance and some kernels get worse. But while we scaled the data in 5-2, kernels other than Linear didn't improved nor get worse. So, we can conclude that scaling has higher affect in data 5-1.

c. Comparison of different kernel functions. [5%]

Kernel/Scaling	With scaling (0, 1)	Without scaling
Linear	80.4895%	**84.1958%**
Polynomial	27.7622%	27.7622%
RBF	27.7622%	27.7622%
Sigmoid	27.7622%	27.7622%

(Testing accuracy)

From the table above, we can see that Linear kernel performs the best. Polynomial, RBF and Sigmoid kernel have the same performance which is worst than Linear kernel. I also ran every kernel in scaled data, it results are the same.

d. Parameter set and performance of your best model. (Report training accuracy and testing accuracy) [5%, Surpass baseline 5%]

Kernel = Linear

C = 0.675

Training accuracy = 97.1149%, 86.6915 (5-fold cross validation)

Testing accuracy = 84.5455%

I only change the change the kernel to linear and c to 0.675, all other parameters are the default values used by LIBSVM

e. We know that the curse of dimensionality causes overfitting. How does it influence Naive Bayesian, Decision Tree and SVM separately? [5%]

	Naïve Bayesian (alpha = 0.05)	Decision Tree (max_depth = 55)	SVM (C = 0.675)
CV 10-fold	91.896%	61.514%	87.4826%
Testing Acc.	89.441%	62.867%	84.5455%

From the table above, we can see that Naïve Bayesian have the best performance in this dataset. And from 10-fold cross validation accuracy and testing accuracy, we know that it has slightly overfitting the data. Moreover, SVM model is better than decision tree but slightly worse than Naïve Bayes model which also overfit the data. Finally, decision tree model has the worst performance of the three model but since we constrained the depth of the tree, it wasn't overfit the data.

f. More discussions are welcome. [Bonus 1%]
This data maybe not suitable for kernels other than linear. The performance of polynomial, RBF and sigmoid are worst for this data. Maybe this data is hardly separable using polynomial, RBF and sigmoid kernel.

5.3 Abalone dataset: Testing label is provided.

a. Your data preprocessing and scaling range. Please state clearly. [10%]
For data preprocessing, since there is a categorical feature (sex), so I used one-hot encoder to encoder the feature. Then output the data into LIBSVM data format which is label <index>:<value> <index>:<value>… index 1 will be the second feature in the csv; index 2 will be the third feature in the csv following on until the last feature, then for the first feature I append in the end of the index which is the sex category. The category has 3 classes – I, F and M which replaced by 8:1.0, 9:1.0 and 10:1.0 respectively. For example, if a data has feature of F and class of 3 then the output will look like '3 1:value 2:value … 7:value 9:1.0'
And for the scaling range, I used a for loop to loop all the lower and upper bound from -10 to 10 and found that when the scaling range is in [-5, 5] get the best performance.

b. Comparison of different kernel functions. [5%]

Kernel/Scaling	With scaling (-5, 5)	With scaling (-1, 1)	Without scaling
Linear	66.443%	65.1965%	64.5254%
Polynomial	**67.5935%**	57.047%	57.047%
RBF	66.1553%	62.4161%	59.8274%
Sigmoid	50.0479%	58.9645%	57.4305%

(Testing accuracy)
From the table above, we can see if scaling the data into range [-5, 5], Linear and RBF kernel have almost the same accuracy but Sigmoid has worst performance on this dataset. Polynomial kernel outperforms other three kernels. But for data in scaling range [-1, 1] and without scaling, the best performance is Linear kernel. RBF kernel is better than Polynomial and Sigmoid kernel.

c. Parameter set and performance of your best model. (Report training accuracy and testing accuracy) [5%, Surpass baseline 5%]
Kernel = Polynomial
Scaling range = [-5, 5]
Training accuracy = 69.2725%
Testing accuracy = 67.5935%
I only change the change the kernel to polynomial, all other parameters are the default values used by LIBSVM

d. More discussion is welcome. [Bonus 1%]
Scaling this data into a suitable range really is a must. In general, we scale data values into range of [0, 1] but in this data it actually has really better performance while scaling in the range of [-5, 5].

5.4 Income dataset

a. Your data preprocessing / data cleaning. Please state clearly. [10%]
Since there are numerical and categorical features in income dataset, so I decided to scale numerical feature using MinMaxScaler (sklearn.preprocessing library) and using OneHotEncoder (sklearn.preprocessing library) to encode categorical features.
For the missing value, I have tried 3 kinds of method.

    a. First method is removing those columns which have missing value.
    b. Second method is encoding the missing value as a new category of the columns.
    c. Third method is replacing the missing value with the most frequent category of the columns.

It turns out that 3 methods have the almost the same accuracy. I used the second method for the homework.
Then I need to export csv into LIBSVM format. This data after pre-processing(scaling and one-hot encoding) has about 108 features, most of the features are one-hot then it will be very sparse. For example,

```
0 1:0.5890410958904109 2:0.07833913821394783 3:0.5333333333333333 6:0.346938775510204 14:1.0 27:1.0 38:1.0 40:1.0 55:1.0 64:1.0
65:1.0 106:1.0
```

label <index>:value <index>:value <index>:value … as mention in the README. For testing data, since we don't have true label, so I decided to place 0 into the label. Every testing data will look like

```
0 1:0.0821917808219178 2:0.1638296906945608 3:0.6 6:0.24489795918367346 11:1.0 31:1.0 36:1.0 45:1.0 55:1.0 62:1.0 66:1.0 106:1.0
0 1:0.4931506849315069 2:0.05327058689679412 3:0.5333333333333333 6:0.39795918367346933 11:1.0 27:1.0 34:1.0 42:1.0 54:1.0 64:1.0 66:1.0 106:1.0
0 1:0.1232876712328767 2:0.04281841570439737 3:0.6 6:0.37755102040816324 11:1.0 31:1.0 36:1.0 46:1.0 55:1.0 62:1.0 65:1.0 106:1.0
0 1:0.3972602739726027 2:0.10265056063430877 3:0.8666666666666667 6:0.07142857142857142 11:1.0 28:1.0 32:1.0 43:1.0 58:1.0 64:1.0 65:1.0 106:1.0
0 1:0.4520547945205479 2:0.10973289874044362 3:0.8 6:0.39795918367346933 11:1.0 25:1.0 34:1.0 51:1.0 54:1.0 64:1.0 66:1.0 106:1.0
```

b. How do you choose parameters set and kernel function? [5%]

```
for i in $(seq 0 1 3)
do
for c in $(seq 1.0 1.0 400.0)
do
# Calculate cross validation of each combination, 10-fold
$1/svm-train -t $i -c $c -v 10 $2 $2.model &>> $2.log

# Retrain model
$1/svm-train -t $i -c $c $2 $2.model &>> $2.log

# Calculate training accuracy
$1/svm-predict $2 $2.model $4 &>> $2.log
done
done
```

I used shell to do grid search. I looped all the combination of kernels (0, 1, 2, 3) and c (1.0 to 400.0, step 1.0). I logged all the output into a file then find the best combination.

c. Report cross validation accuracy, and training accuracy. [5%]
Cross validation accuracy (5-fold): 85.219%
Training accuracy: 86.1065%

d. Parameter set of your best model. [Surpass baseline 5%, Top 20% in class: 5%]
Kernel = RBF
C = 220
Scaling range = [0, 1]
I only change the change the kernel to RBF and c to 220, all other parameters are the default values used by LIBSVM

e. More discussion or observation are welcome. [Bonus 1%]
In this data, I tried not to scale the value, the SVM train very slow and hard to converge. Scaling is important in data which values have high standard deviation.