Data Mining HW5

LIBSVM

Name: 劉廷緯 Student ID: R07942089

5.1 Iris dataset : Testing label is provided.

Kernel Type	Testing Accuracy	Testing Accuracy with Scaling
Linear	100.00%	97.33%
Polynomial	98.66%	70.66%
Radial Basis Function	97.33%	98.66%
Sigmoid	33.33%	96.00%

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

   From the table above, we see that scaling improves the classification performance of the Sigmoid kernel by a great margin, and the Radial Basis function kernel is also improved slightly. On the other hand, the Linear and polynomial kernel did not improve by adding scaling.

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

   Comparison of different kernel functions is shown in the table above, we see that the Linear kernel achieves the best performance in this dataset. Polynomial and Radial Basis Function also performs well, however Sigmoid performs poorly without scaling.

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

- Best model training accuracy: 98.66%
- Best model testing accuracy: **100%**
- Parameter and setting for best model: **svm-train -s 0 -t 0**

d. **More discussions is welcome. [Bonus 1%]**

We know that in this dataset one class is linearly separable from the other 2, the latter are not linearly separable from each other. However, we can observe that the Linear kernel performs the best in this dataset.

5.2 News dataset : Testing label is provided.

Kernel Type	Testing Accuracy	Testing Accuracy with Scaling
Linear	83.36%	79.86%
Polynomial	49.51%	35.59%
Radial Basis Function	70.91%	69.02%
Sigmoid	70.91%	67.62%

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

From the table above, we see that none of the kernel type benefits from scaling, adding scaling results in a decline in performance, we can conclude that this dataset does not require scaling. Each kernel is ran under the same settings for fair comparison.

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

For the Iris dataset (5-1-a), some kernel functions can benefit from scaling (Radial Basis function and Sigmoid), whereas for the News dataset (5-2-a), none of the kernel functions benefit from scaling. We observe that, in the News dataset

TF-IDF features are provided, this type of feature doesn't need scaling.

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

Comparison of different kernel functions is shown in the table above, we see that the Linear kernel achieves the best performance in this dataset. Radial Basis Function and Sigmoid also performs well, however Sigmoid performs poorly without scaling.

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

- Best model training accuracy: 97.63%
- Best model testing accuracy: **84.97%** (baseline: 84.3%)
- Parameter and setting for best model: **svm-train -s 0 -t 0 -e 0.01 -w3 2.5**

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

The News dataset is a small dataset, hence models will suffocate from overfitting problems, from hw-4 I achieved a best performance of 89.51% and 64.90% for the Naive Bayesian and Decision Tree models, respectively. (Testing set accuracy)

We can conclude that the degree of overfitting effect of the three classifiers can be ranked as follow: Decision Tree > SVM > Naive Bayes. Where Decision Tree models suffers the most from overfitting, and Naive Bayes classifiers suffers the least. SVM classifiers are slightly less robust than Naive Bayes, however it still performs pretty well (84.97%).

For some theoretical support and explanation: SVM is good at dealing with small data samples, since only support vectors are used to construct the separating hyperplane (i.e., classifier, hence suffocate less from overfitting issues). On the other hand, Decision Tree is the only classification technique (out of the 3) which does not suit small datasets. This is caused by its greedy characteristic, small data set may cause significant overfitting problems, hence explaining its poor performance (64.90%) on the News dataset.

f. **More discussions is welcome. [Bonus 1%]**

By setting the parameter C of class 3 to weight*C (with the -w3 2.5 parameter) performance can be increased by a small margin (~1%). This is because there are more class 3 samples in the testing set.

5.3 Abalone dataset : Testing label is provided.

Kernel Type	Testing Accuracy	Testing Accuracy with Scaling
Linear	66.63%	57.81%
Polynomial	61.84%	57.91%
Radial Basis Function	66.25%	55.90%
Sigmoid	56.28%	54.94%

a. **Your data preprocessing and scaling range. Please state clearly. [10%]**

Preprocessing pipeline:
- Specify each entry to either one of the data type: (`int`, `str`)
- Change the first column
    (the sex attribute which is categorical and in `str` date type) into one-hot encoding vectors.
- Write the resulting feature into LibSVM format.
- Scaling range is set to **[0, 1]** for the results generated in the above table, however scaling is not applied in the best model.

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

Comparison of different kernel functions is shown in the table above, we see that the Linear kernel achieves the best performance in this dataset. Radial Basis Function also performs well. Polynomial and Sigmoid kernels performed not as

well as the previous two kernels.

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

- Best model training accuracy: 65.16%
- Best model testing accuracy: **66.63%** (baseline: 65.1%)
- Parameter and setting for best model: **svm-train -s 0 -t 0 -e 0.01 -c 20**

d. **More discussion is welcome. [Bonus 1%]**

Classification performance did not benefit from scaling on this dataset, since the original features are already in a small range: [0, 1].

5.4 Income dataset

Kernel Type	Cross Validation Accuracy	Cross Validation Accuracy with Sklearn Scaling	Cross Validation Accuracy with LibSVM Scaling
Linear	53.67%	85.07%	84.99%
Polynomial	44.37%	82.91%	75.96%
Radial Basis Function	75.68%	84.76%	83.47%
Sigmoid	75.96%	83.11%	83.41%

a. **Your data preprocessing / data cleaning. Please state clearly. [10%]**

Preprocessing pipeline:
- Specify each entry to either one of the data type: (`int`, `str`)
- Identify all missing entries `'?'` and replace them with `np.nan`

- Impute and estimate all missing entries:
    1. If dtype is `int`: impute with mean value of the feature column
    2. If dtype is `str`: impute with most frequent item in the feature column
- Split data into categorical and continuous and process them separately:
    1. categorical features index = [1, 3, 5, 6, 7, 8, 9, 13]
    2. continuous features index = [0, 2, 4, 10, 11, 12]
- For categorical data:
    1. 8 categorical attributes are transformed into a 99 dimension one-hot feature vector
- Normalize each attribute to zero mean and unit variance.
- Write the resulting feature into LibSVM format.

b. **How do you choose parameters set and kernel function ? [5%]**
I first choose the best kernel function that has the highest performance on this dataset by running all 4 kernels with default parameters. After finding out which kernel functions performs the best, I fine tune other parameters one at a time with that kernel function, trying all reasonable values using several "for" loops. All performances is measured by N-fold cross validation accuracy.

c. **Report cross validation accuracy, and testing accuracy. [5%]**
- Best model training accuracy: 85.23%
- Best model 3-Fold cross validation accuracy: **85.10%**
- Best model Testing accuracy: Unknown, testing label was not provided (baseline: 85.0%)

d. **Parameter set of your best model. [Surpass baseline 5%, Top 20% in class: 5%]**
- Parameter and setting for best model: **svm-train -s 0 -t 0 -c 20 -e 0.1 -m 1000**

e. **More discussion or observation are welcome. [Bonus 1%]**
From the above table, we can easily observe that without scaling, performance dropped greatly. Sklearn's `StandardScaler` results in a higher performance than libsvm's scaling, hence I used Sklearn's `StandardScaler` in my best model to scale each feature column to zero mean and unit variance. This is because the attribute columns varies from each other by a large scale, without scaling or some sort of normalization SVM fails to find a suitable separating hyperplane.