2021

CSC503 Assignment 2

HAIJIA ZHU
V0083542

Table of Contents

This assignment contains two sections:

1. Experiments and analysis.
    a. Logistic regression.
    b. Support vector machines with linear kernel.
    c. K-fold cross-validation implementation and testing result.
    d. Support vector machine Gaussian kernel.
2. Logistic regression and SGD (CSC503 only).

Many figures use the index on the x-axis instead of the actual parameters, since the spacing increase nonlinearly, which makes it hard to present in the figures.

1. Pre-processing data

1.1. Rescaling features

All the features are rescaled to [0,1], as mentioned in [1], a large number, such as 255 in the original datasets, potentially result in an extremely small ***w***, and reduce the accuracy.

1.2. Reassign labels

Similar to Rescaling, depends on the implementation, some classifiers have difficulty dealing with non-scale labels, (e.g., Matlab build-in logistic regression struggle to fit the data sets if the label is 5 and 7). Thus, all the labels are assigned to 1 or 0.

1.3. Randomized the train sets

For k-fold cross-validation, the data set is randomized to ensure fairness before splitting into different “groups “.

2. Experiments and analysis

2.1. Logistic regression

This part will test the regularization parameter *C* using all (2000) test samples. The classifier is trained using all 12000 examples. However, the implementation in Matlab differs from sci-kit learn and us the following cost function:

$$\min_{w,b} \lambda \times \|w\|^2 + \sum_{i=i}^{n} \ell\big(y, \sigma(< w, x > +b)\big)$$

Overall, 21 different regularization parameters are tested and the red star in the figure represents the highest accuracy.

Table 1 Regularization Parameters for Logistic Regression with L2 Regularization

Regularization parameters

1e-16,1e-15,1e-14,1e-13,1e-12,1e-11,1e-10, 1e-9,1e-8,1e-7,1e-6, 1e-5, 1e-4, 1e-3, 1e-2,0.05, 1e-1, 1, 1e1, 1e2, 1e3

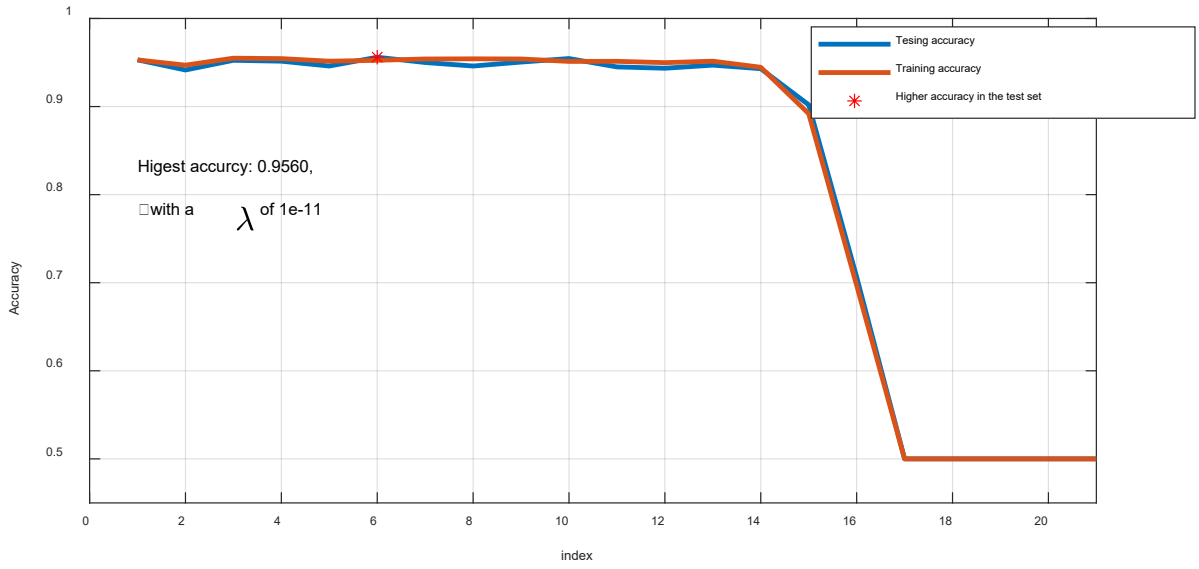


Figure 1 Logistic Regression

The tested regularization parameter starts from a small number (1e-16) to 1000. Even though it covers board range of parameters, it does not show a sign of overfitting (too little regularization). And the highest accuracy in the test dataset is achieved at 1e-15, which is extremely small. This suggests that this dataset could achieve a good performance using logistic regression without regularization. When increasing the λ, as we give more emphasise to the training errors, the accuracy both in training and testing datasets varies slightly, but overall performs well until it meets a threshold. When the λ exceeds *1e-2* (where the index is 15 in Figure 1), the accuracy drops significantly. This is mainly because we are not minimizing the training error but trying to construct a hyperplane with a small norm, which leads to underfitting. When the λ is too large (1e-1 in this case), the classifier is not able to classify the samples and classify all the input samples as 1 (or 0). As we continue increasing the regularization parameters, eventually, one could obtain a classifier with norm(***w***) close to *0*. The output of the classifier will mainly depend on the bias term, which leads to misclassification.

2.2. SVM with linear kernel

Matlab does have its SVM toolbox, but the SVM toolbox used in this assignment is from libsvm [2].

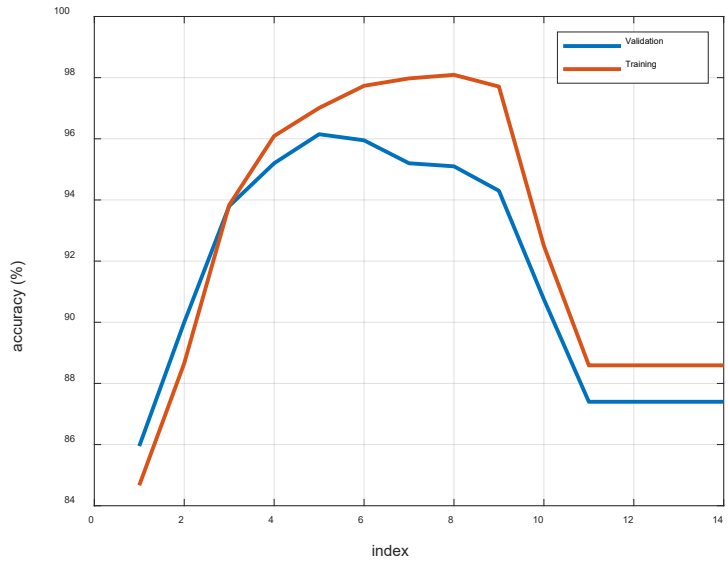
The selected tuning parameter is presented in Table 2. Since the accuracy does not change significantly between 1e-16 to 1e-8, only those two numbers are selected within range.

Table 2 Tunning Parameters for SVM with Linear Kernel.

Tuning parameter (*C*)

1e-16, 1e-4, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4, 1e5, 1e6,1e7,1e8

As we increase the *C*, more points are allowed for misclassification, thus we can find an increase of accuracy both in taring and testing datasets. However, if *C* is too large, that means too many points are allowed to misclassify, which leads to a decrease of accuracy. The accuracy in the train set differs from validation sets, which suggests there is overfitting.



2.3. Implementing K-fold cross-validation

The K-fold cross-validation is implemented in Matlab and is applied on both logistic regression and support vector machines.

In logistic regression, the objective is to find an optimal regularization parameter among the feasible input parameters as shown in Table 1.

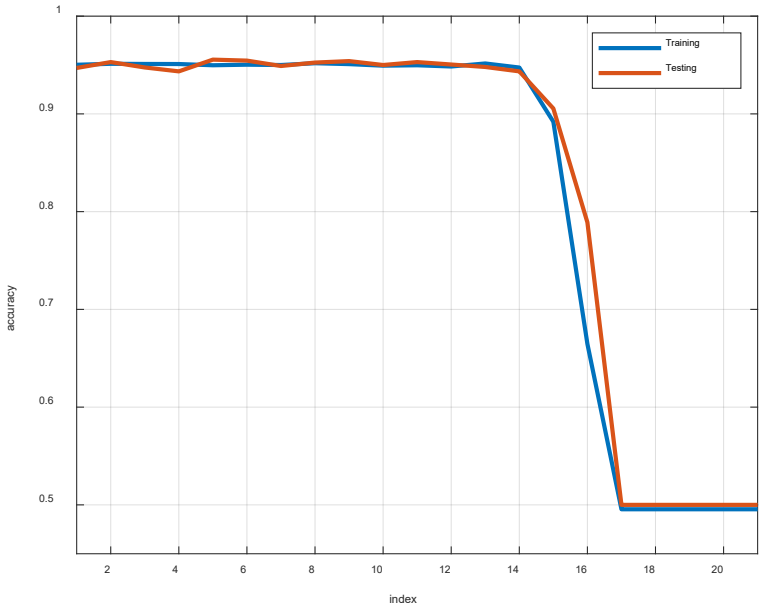


Figure 2 Logistic Regression Cross-validation with K =5.

The highest accuracy is achieved at 1e-12 and 1e-9 for testing dataset and training sets, with the accuracy of 0.9555 and 0.9518, respectively. It would be ideal if the parameters are the same on both datasets, however with such low fluctuation, as shown in Figure 3, it is likely that parameters within this range can achieve relatively high accuracy. The accuracy of each batch in the k-fold cross-validation, as shown in Figure 4, which have a similar pattern as shown in Figure 3.

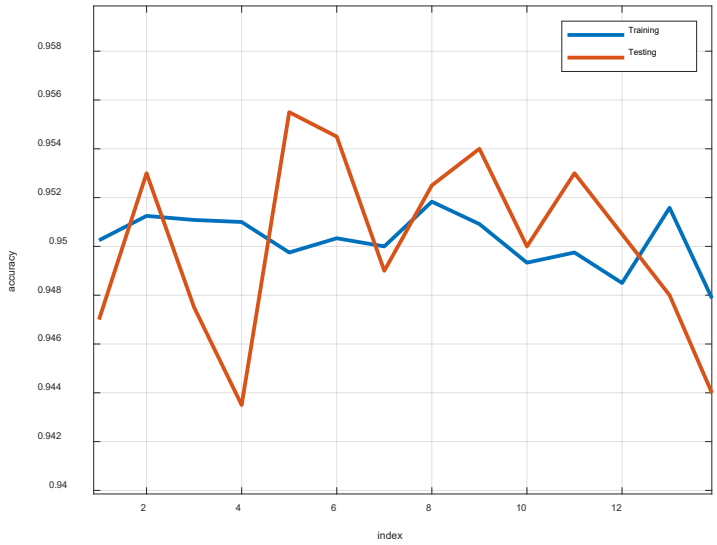


Figure 3 Magnification View

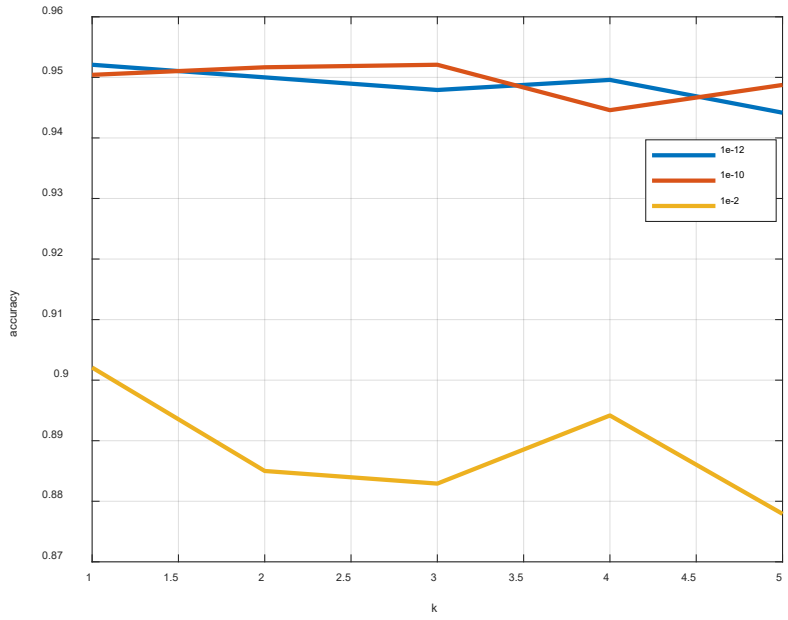


Figure 4 Accuracy of each fold using different parameters.

Changing the *k* does not significantly affect the accuracy in both training sets and test set. The simulation result is presented in Figure 5.

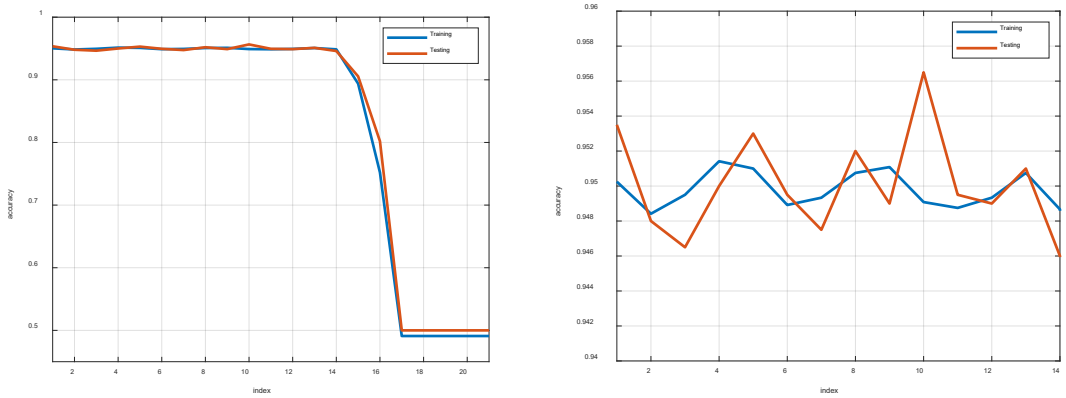


Figure 5 Cross-validation with K = 10.

The k-fold cross-validation SVM is also implemented and the result is presented below: as we can see in Figure 6, when the tuning parameter is small, SVM suffers underfitting. As we increase the *C*, the accuracy increases as well. Eventually, the classifier will overfit (and achieve an accuracy of 100% in the train set.), which is expected as a larger *C* allows more misclassification.

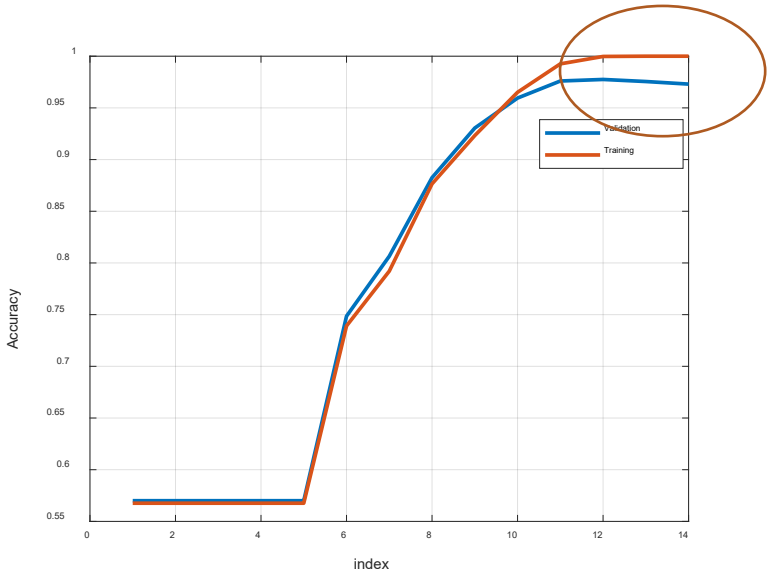


Figure 6 SVM Accuracy

The test errors (using MSE as loss function) on the given test dataset for logistic regression and SVM are shown in Figure 7.

When the SVM achieves high accuracy in train sets (as shown in the red box in Figure 6), that classifier performs poorly in the test sets, although the train errors are still relatively small. This suggests this classifier is overfitting.

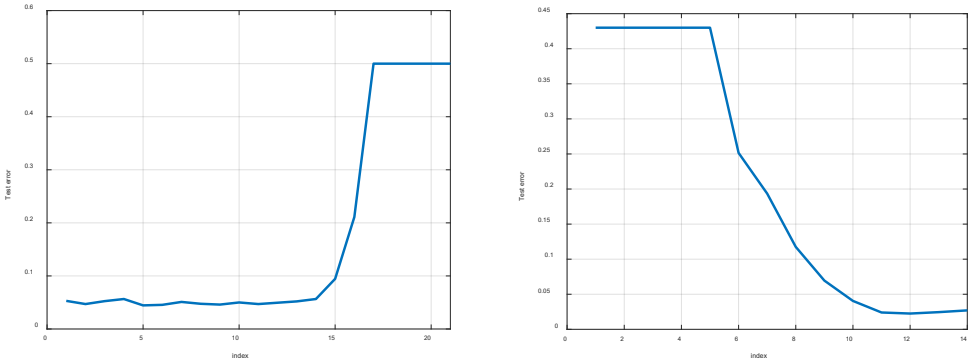


Figure 7 Test errors (left: logistic regression; right: SVM)

To evaluate both algorithms, one can calculate the confidence interval of SVM and logistic regression respectively as in Figure 8 and Figure 9. For logistic regression, the smallest confidence interval is located at 1e-12, and there is a 95% likelihood that the confidence interval [0.9465, 0.9645] covers the true classification error of the model on unseen data. For SVM, the smallest confidence interval is located at C=100000, and there is a 95% likelihood that the confidence interval [0.9710, 0.9840] covers the true classification error of the model on unseen data.

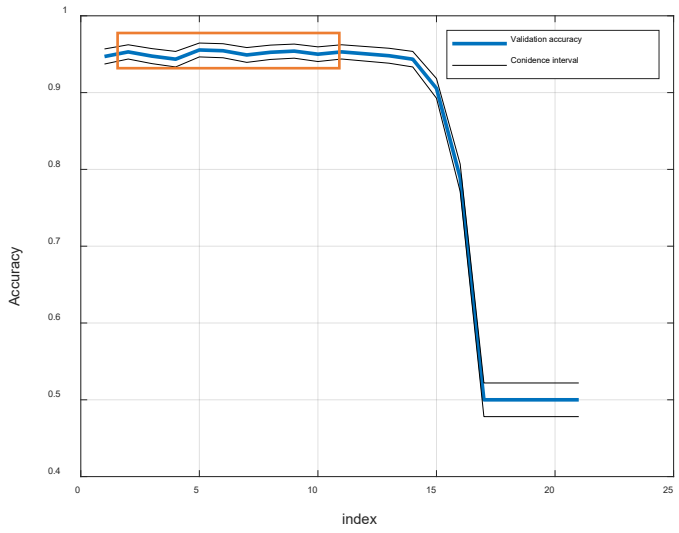


Figure 8 Confidence interval for logistic regression.

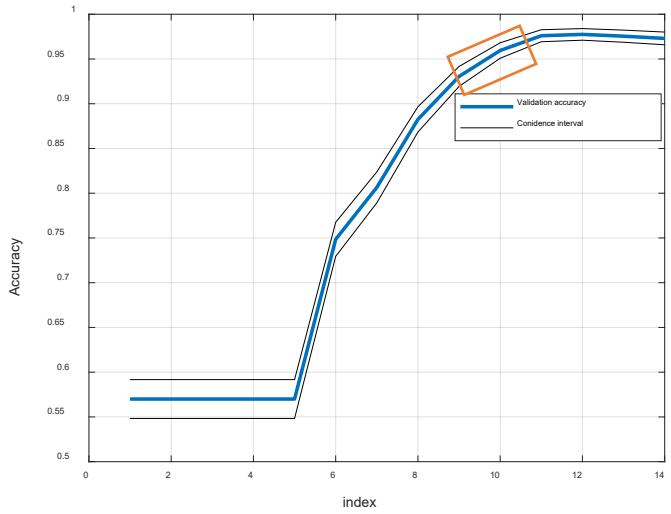


Figure 9 Confidence interval for SVM.

Compare both algorithms, we could see that both algorithms are capable of achieving high accuracy with an optimal parameter. However, for this dataset, logistics is more robust and not as sensitive as C-SVM, as shown in the red box in Figure 8. There is a huge difference in train time. The logistic regression in Matlab is capable of vectorizing the training process but libsvm is hard to vectorize. Although parallel computing is applied on SVM, a 4.3-hour training time is needed to train for all 14 parameters with K=5 on a workstation equipped with a 12 core intel E5-2620 processor. While the logistic regression classifier only needs 3 minutes to train all 21 parameters and find an optimal parameter.

2.4. SVM with Gaussian kernel

C	γ
1e-6, 1e-5, 1e-3, 1e-2, 1e-1,0.5, 1, 1e1, 1e2, 1e3,1e4,1e5,1e6,1e7	1e-6, 1e-5, 1e-3, 1e-2, 1e-1, 0.5, 1, 1e1, 1e2, 1e3

γ significantly affect the train. When this parameter is small, the training of the classifier is extremely slow. Typically, the γ can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. A smaller number means further; if γ is too large, that means no amount of regularization with C will be able to prevent overfitting.

Although it is hard to see in Figure 15, the accuracy for both training and validation accuracy is low when γ is small, which is expected. As we increase it, the training error decreases significantly, while the validation accuracy improves slightly. The model suffers underfitting (between 1e-5 to 0.1) and resulting in a model that behaves similar to a linear model. As we increase the γ, the power of SVM is fully utilized, with the highest accuracy of 97.65% in the validation set.

Finally, one can also observe that for some intermediate values of γ we get equally performing models, as shown in Figure 11. *C* will not affect the performance as it has already been optimized using k-fold cross-validation. Which suggest that the trained model does not change (no better support vector can be found) as we cross some threshold.

Compare this classifier to an SVM with a linear kernel. The RBF-SVM performs better (lower error) when the parameters are properly selected. As mentioned, the RBF-SVM behaves similarly to an SVM with linear kernel when γ is small. Compare the test error with SVM with linear kernel, both algorithms can achieve a low test error, however, the test error starts to increase as the *C* become too large, while in SVM with RBF, the training error remain constant, as shown in Figure 12 and Figure 7. When the SVM with linear kernel achieves a similar test error as in RBF-SVM, the accuracy in the testing set perform poorly as shown in Figure 7 and Figure 6. While the accuracy and the test error perform well in RBK-SVM.

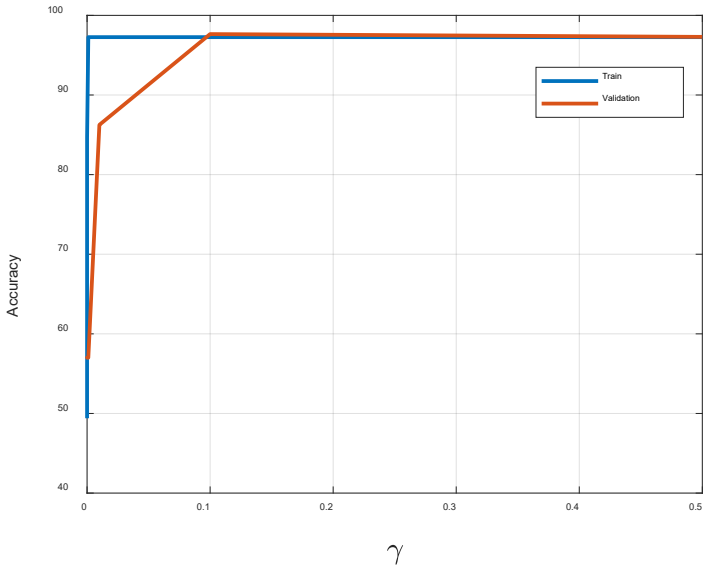


Figure 10 Part of RBF-SVM Training and validation

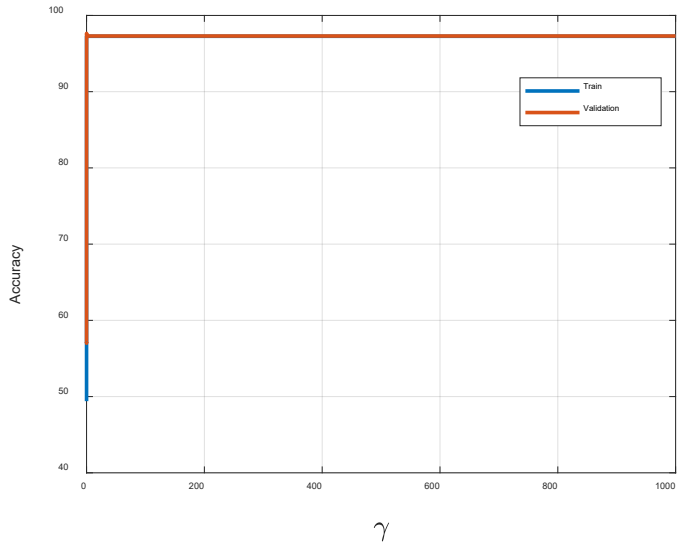


Figure 11 RBF-SVM Training and validation

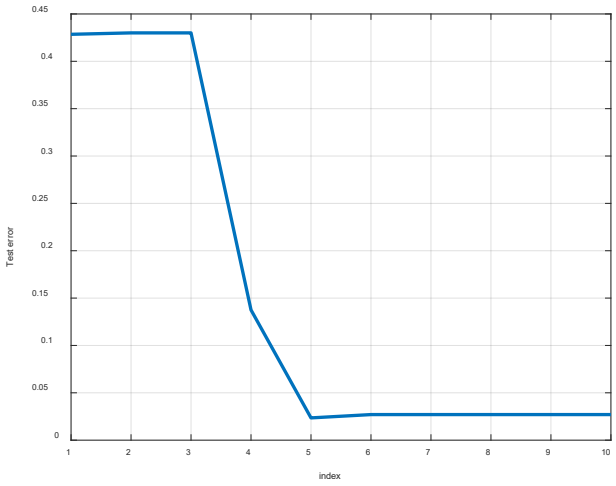


Figure 12 Test error

One interesting thing found during the testing is that *C* does not affect the training time as large as expected (compare to γ). Typically, larger C will lead to a longer fitting time, but in this case, smaller γ leads to extremely long fitting time, although *C* does affect the train time. This issue should be investigated in future studies.

3. Logistic regression and SGD

3.1. Implement SGD

The steps of logistic regression with SGD can be summarized as below:

Loop over the following process until some stop condition is met:

1. Calculate the cost.
2. Calculate the gradient.
3. Update the gradient.

The cost is easy to calculate and is defined as:

$$J = \frac{1}{2}\|w\|^2 + C \times \ell\big(y, \sigma(< w, x > + b)\big),$$

where:

- $\ell(y, \hat{y})$ is the cross-entropy loss;
- $\sigma(z)$ is the sigmoid function;
- C is the regulation parameter.

The gradient is calculated and is presented below:

$$\frac{\partial J}{\partial w_i} = C \times (\sigma(< w, x > + b) - y) \times x_i + w_i$$

$$\frac{\partial J}{\partial b} = C \times (\sigma(< w, x > + b) - y)$$

Three different learning rate schedules are tested, they are: constant learn rate, time-based learning rate, and step-based learning rate. A time-based schedule drops the learning rate at each time. A step-based schedule reduces the learning rate by a factor every few iterations. Different learning rate schedules can be found in Figure 13.

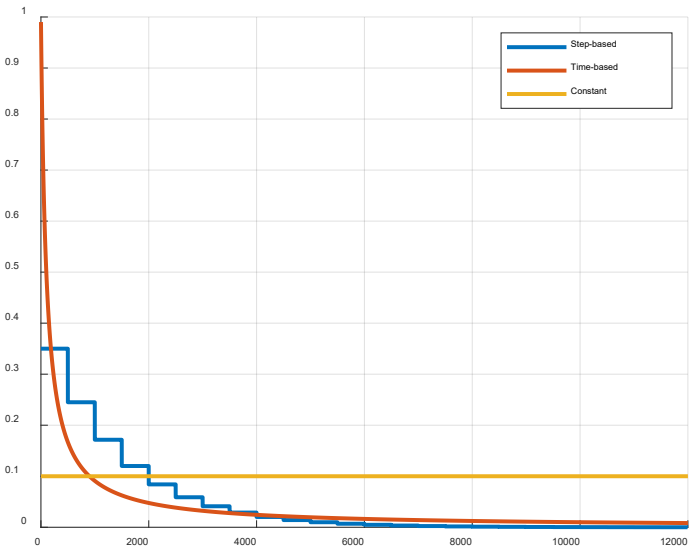


Figure 13 Leaning rate schedules.

For logistic regression with SDG, the implemented code can be found in APENDIX A.

All three schedulers are test and the time-based scheduler is select as it is not performed better in the initial test.

3.2. Searching optimal parameters

Table 3 Schedule Parameters

α	t_0
1e-1, 1, 5, 1e1, 1e2, 5e2, 1e4, 5e4	1e-3, 1e-2, 1e-1, 1, 5, 1e1, 1e2, 5e2, 1e4, 5e4

To find the optimal parameter for the time-based schedule, k-fold cross-validation is performed. The candidates are illustrated in Table 3. A contour of the accuracy in the train sets using k-fold cross-validation is presented in Figure 14, where the contour represents the accuracy. The red circle on the top left is the parameter set with higher accuracy (0.9427) in the CV training set. The selected learning rate and its changes are shown in Figure 15. It is interesting that if the change of learning rate is not as aggressive as other parameter sets and the final learning rate is relatively large, it still achieves a relatively good performance (achieve

accuracy of 94.15% both in train and validation, highlighted in red). This suggests that there might exist some overfitting when using some parameter sets (who has a smaller final learning rate). The selected α, t_0 also achieve a test accuracy of 94.35% when training with all the training sets (and tested on the test set), which is also the highest. The training accuracy and validation accuracy are presented in Table 6 and Table 5 respectively.

Table 4 Tuning C using k-fold CV.

Regularization parameters
1e-3,1e-2,1e-1, 1, 5, 1e1,5e1,1e2,5e2,1e3,5e3,1e4,1e5,1e6,1e7

Table 5 k-fold Cross-validation Training Accuracy (k=5)

t_0 / α	1.00e-03	1.00e-02	1.00e-01	1	5	1.00e+01	1.00e+02	5.00e+02	1.00e+04	5.00e+04
1.00e-01	89.65	89.74	90.26	91.72	92.82	92.79	91.17	89.23	80.44	72.19
1	93.36	93.29	92.93	91.16	91.23	91.88	93.53	93.98	94.28	91.30
5	93.36	93.35	93.29	92.85	92.47	92.65	93.30	93.36	93.56	94.15
1.00e+01	93.16	93.15	93.14	93.16	93.15	93.15	93.17	93.19	93.25	93.68
1.00e+02	85.86	85.86	85.86	85.86	85.86	85.86	85.87	85.99	87.75	90.94
5.00e+02	87.28	87.13	88.16	87.70	88.34	88.04	88.42	88.06	87.34	89.12
1.00e+04	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	60.35	77.98
5.00e+04	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.03

Table 6 Validation Accuracy

t_0 / α	1.00e-03	1.00e-02	1.00e-01	1	5	1.00e+01	1.00e+02	5.00e+02	1.00e+04	5.00e+04

1.00e-01	90.15	89.55	90.15	92.60	93.75	93.40	92.45	90.50	82.70	71.55
1	93.30	93.30	93.15	91.65	91.75	92.25	93.85	94.20	94.35	93.25
5	93.65	93.65	93.60	93.45	93.20	93.50	93.65	93.65	94.00	94.15
1.00e+01	93.55	93.55	93.55	93.55	93.55	93.55	93.55	93.55	93.65	93.95
1.00e+02	91.05	91.05	91.05	91.05	91.05	91.05	91.05	91.15	91.95	93.60
5.00e+02	89.20	89.75	90.25	82.05	86.50	90.95	90.40	90.80	88.90	90.50
1.00e+04	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	66.45	60.40
5.00e+04	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00

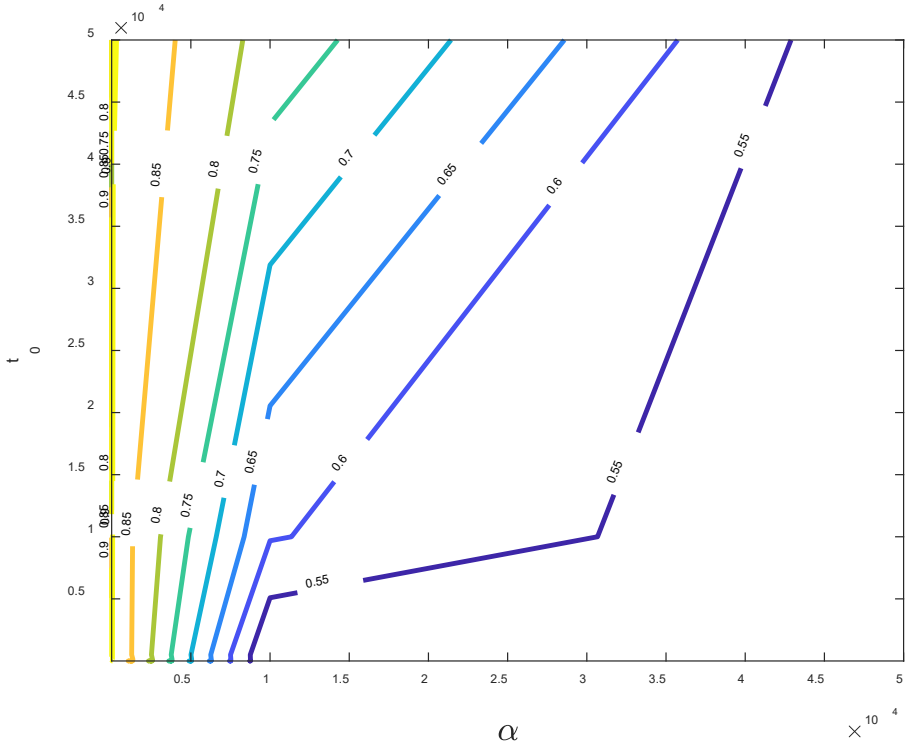


Figure 14 Accuracy

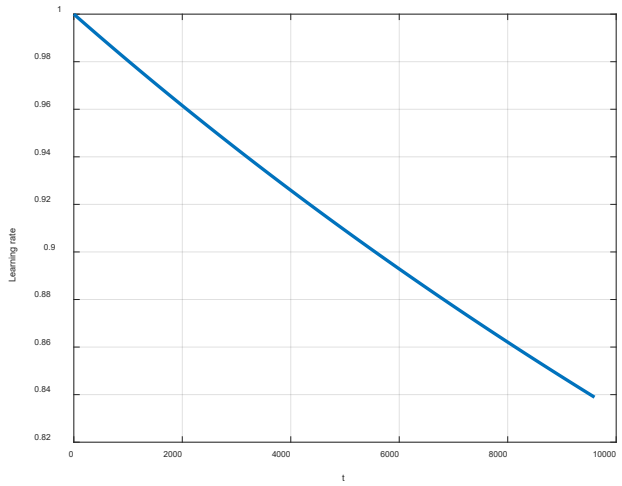


Figure 15 selected schedule of a parameter set that near the optimal set (Accuracy 94.15%, Validation accuracy: 94.15%,).

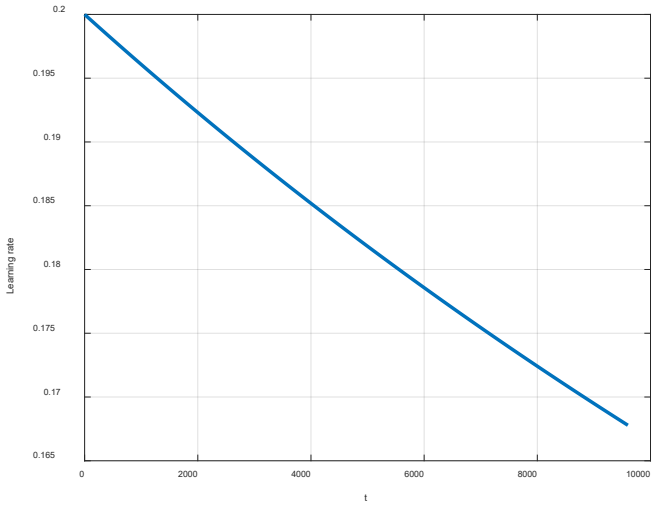


Figure 16 Selected schedule.

The parameter ($\alpha = 1; t_0 = 1e4$) is selected based on the cross-validation result. Next, we perform cross-validation to find optimal *C*. The candidacy parameters are presented in Table 4. And the accuracy of the validation and train is presented in Figure 17. When *C* is small, the accuracy is low as expected, since the error takes a limited less effort to fit the correct ***w, b***. As we increase the regularization parameter, the accuracy increases both in testing and training sets. However, it suffers underfitting, until *C* is in an ideal range (such as index between 6 and 10 in Figure 17). If too little regulation is applied, the classifier suffers overfitting, which results a decrease in accuracy (as shown in Figure 17, where index is greater than 11 (C= 5e3))

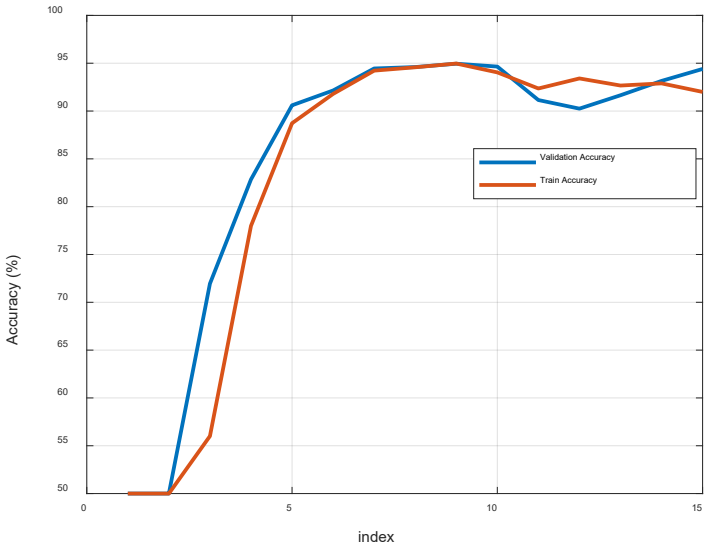


Figure 17 Accuracy with different C

[1] R. Berwick, “An Idiot’s guide to Support vector machines (SVMs),” p. 28.

[2] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, “A Practical Guide to Support Vector Classification,” p. 16.

```
% load
filenameImagesTrain = 'train-images-idx3-ubyte';
filenameLabelsTrain = 'train-labels-idx1-ubyte';
filenameImagesTest = 't10k-images-idx3-ubyte';
filenameLabelsTest = 't10k-labels-idx1-ubyte';

XTrain_all = processImagesMNIST(filenameImagesTrain);
YTrain_all = processLabelsMNIST(filenameLabelsTrain);
XTest_all= processImagesMNIST(filenameImagesTest);
YTest_all = processLabelsMNIST(filenameLabelsTest);
% Select label 5 and 7
XTrain = XTrain_all(YTrain_all==5|YTrain_all==7,:);
YTrain = YTrain_all(YTrain_all==5|YTrain_all==7,:);
XTest = XTest_all(YTest_all==5|YTest_all==7,:);
YTest = YTest_all(YTest_all==5|YTest_all==7,:);
YTest(YTest==5)=1;
YTrain(YTrain==5)=1;
YTest(YTest==7)=0;
YTrain(YTrain==7)=0;

%
clearvars -except XTrain YTrain XTest YTest
global t alpha t0 d r C learning_rate_init
close all


%%  main
% Cost function parameter
C = 20;
% number of epoch
no_epoch = 5;
% %% learn rate update
% For learn rate:
% 1:Time-based
% 2:Step-based;
% any other number: constant learn rate. eg: 1e-3
lr_setup = 2;
t = 1;
t0 = 0.1;
alpha = 0.1;

% another schulder
d = 0.7; % learning rate should change at each drop
r = 500; % often the rate should be dropped
learning_rate_init = 0.5;

%%%%%%%%%%nothing need to be change from here%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
loss = [];
learn_rate = [];
w=rand(size(XTrain(1,:)));
```

```
b= rand(1);
for i =1:no_epoch


    [w,b,l,lr] = LR_SGD_fit(w, b, XTrain, YTrain, lr_setup);
    loss = [loss,l];
    learn_rate = [learn_rate, lr];
end

[y_hat,prob,accuracy]=LR_SGD_predict(w,b,XTest,YTest);
accuracy
figure
plot(loss)
title('loss')
figure
plot(learn_rate)
title('learn rate')
%
```