Homework 2

2020 Spring CSCI 5525: Machine Learning

Due on March 1st 11:59 p.m.

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems, but each person must write up the final solutions individually. You need to fill in above to specify which problems were a collaborative effort and with whom. (2) Regarding online resources, you should **not**:

- Google around for solutions to homework problems,
- Ask for help on online.
- Look up things/post on sites like Quora, StackExchange, etc.

Submission. Submit a PDF using this LaTeX template for written assignment part and submit .py files for all programming part. You should upload all the files on Canvas.

Written Assignment

Instruction. For each problem, you are required to write down a full mathematical proof to establish the claim.

Problem 1. Separability.

(**3 points**) Formally show that the XOR data set (see Lecture 4 note) is not linearly separable. **Hint:** A data set $\{(x_i, y_i)_{i=1}^N\}$ where $y_i \in \{-1, 1\}$ is linearly separable if $\exists \mathbf{w} \in \mathbb{R}^d$ and $\exists b \in \mathbb{R}$ s.t.

$$\text{sign}(\langle \mathbf{w}, x_i \rangle + b) = y_i \qquad \forall i$$

Your answer. For the XOR data set, we know that:

$$x = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \text{ While } y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \text{, but in this case, we suppose } y = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \text{ to do classification.}$$

As for the equation

$$\text{sign}(\langle \mathbf{w}, x_i \rangle + b) = y_i \qquad \forall i$$

or

$$\langle \mathbf{w}, x_i \rangle + b = y_i^t \qquad \forall i$$

we can transform it into a matrix multiplication:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = y_i^t = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Where $a < 0$, $b > 0$, $c > 0$, $d < 0$.

We do matrix elementary transformation to the matrix A : $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ and the augmented matrix $B = (A|y_i^t) = \begin{bmatrix} 0 & 0 & 1 & a \\ 0 & 1 & 1 & b \\ 1 & 0 & 1 & c \\ 1 & 1 & 1 & d \end{bmatrix}$, we get:

$$A \to \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B \to \begin{bmatrix} 1 & 0 & 0 & c-a \\ 0 & 1 & 0 & b-a \\ 0 & 0 & 1 & a \\ 0 & 0 & 0 & b+c-a-d \end{bmatrix}$$

Apparently, $rank(A) = 3$, while $a + d - b - c > 0$, $rank(B) = rank(A|y_i^t) = 4$, so $rank(A) < rank(A|y_i^t)$, there is no solution for the equation $\langle \mathbf{w}, x_i \rangle + b = y_i^t$.

Problem 2. Kernels.

(**4 points**) As you learned in the class, Kernels provide a powerful method to traverse between kernel space and feature space. Using Kernel's properties (mention the properties used):

- (**2 points**) Show that $K(x, y) = K_1(x, y)K_2(x, y)$ is a valid kernel where K_1 and K_2 are valid kernels.
- (**2 points**) Show that the function $K(x, y) = K_1(x, y) + K_2(x, y)$ is a valid kernel function where $x, y \in \mathbb{R}$.

Your answer.

(a) Let $K_1(x,y) = f(x)^\intercal f(y)$ and $K_2 = g(x)^\intercal g(y)$ while $f(x) \in \mathbb{R}^m$ and $g(x) \in \mathbb{R}^n$

$$\begin{aligned} K(x,y) &= K_1(x,y)K_2(x,y) \\ &= (\sum_{i=1}^{m} f_i(x)f_i(y))(\sum_{i=1}^{n} g_i(x)g_i(y)) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} f_i(x)f_i(y)g_j(x)g_j(y) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n}[f_i(x)g_j(x)][f_i(y)g_j(y)] \end{aligned}$$

Apparently, let $h(x) \in \mathbb{R}^{mn}$ where $h_{ij}(x) = f_i(x)g_j(x)$, we have $K(x,y) = h(x)^\intercal h(y)$.

(b) Suppose the size of K, K_1, K_2 is $d * d$, let $V \in \mathbb{R}^d$, we have:

$$\begin{aligned} V^\intercal K(x,y)V &= V^\intercal[K_1(x,y) + K_2(x,y)]V \\ &= V^\intercal K_1(x,y)V + V^\intercal K_2(x,y)V \\ &\geq 0 + 0 \geq 0 \end{aligned}$$

So, $K(x,y)$ is positive semidefinite matrix. $K(x,y)$ is a valid kernel function.

Problem 3. Derivation of SVM Solution.

(**5 points**) In the lecture, we derived the soft-margin SVM solution without the intercept term. Now derive the solution with the intercept term b by going through Lagrange duality. Here the predictor will be $\hat{f}(x) = \text{sign}(\mathbf{w}^\intercal x + b)$. You should use the same conventions for the scalar and vector variables as used in the course.

- What is the Lagrange dual objective?
- State the two relevant KKT conditions: stationarity and complementary slackness.
- What is the condition for which one would get support vectors?
- What is the optimum $\mathbf{w}$ and b?

Note: Be concise in answering the questions above.

Your answer. (a) For this question, the problem is equivalent to the SVM problem:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i \quad suchthat \tag{1}$$

$$\forall i \quad y_i(\mathbf{w}^\intercal x_i + \mathbf{b}) \geq 1 - \xi_i \tag{2}$$

$$\forall i \quad \xi_i \geq 0 \tag{3}$$

The Lagrange dual objective is:

$$L(\mathbf{w}, \mathbf{b}, \xi, \lambda, \alpha) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\lambda_i(1 - \xi_i - y_i\mathbf{w}^\intercal x_i - y_i\mathbf{b}) - \sum_{i=1}^{n}\alpha_i\xi_i$$

(b) With the KKT conditions, the stationarity is:

$$\nabla_{\mathbf{w},\mathbf{b},\xi} L(\mathbf{w}^*, \mathbf{b}^*, \xi^*, \lambda^*, \alpha^*) = 0$$

When $\frac{\partial L}{\partial \mathbf{w}} = 0$:

$$\mathbf{w}^* = \sum_i y_i \lambda_i^* x_i$$

When $\frac{\partial L}{\partial \mathbf{b}} = 0$:

$$\sum_i \lambda_i^* y_i = 0$$

When $\frac{\partial L}{\partial \xi_i} = 0$:

$$C - \lambda_i^* - \alpha_i^* = 0 \quad \forall i$$

With the KKT conditions, the complementary slackness is:

$$\forall i,\ \alpha_i^* \xi_i^* = 0,\ \lambda_i^*(1 - \xi_i^* - y_i(\mathbf{w}^*)^\intercal x_i - y_i \mathbf{b}^*) = 0$$

(c)As support vectors satisfy $\lambda_i^* > 0$, we have:

$$1 - \xi_i^* = y_i((\mathbf{w}^*)^\intercal x_i + \mathbf{b}^*)$$

(d)With the KKT conditions and the conclusion in (a), we have equations below, where x_i is support vector:

When $\frac{\partial L}{\partial \mathbf{w}} = 0$:

$$\mathbf{w}^* = \sum_i y_i \lambda_i^* x_i$$

For $\mathbf{b}$, with the formula in (c), we have:

$$\mathbf{b}^* = y_i^{-1}(1 - \xi_i^*) - (\mathbf{w}^*)^\intercal x_i$$

Programming Assignment

Instruction. For each problem, you are required to report descriptions and results in the PDF and submit code as python file (.py) (as per the question).

- **Python** version: Python 3.
- Please follow PEP 8 style of writing your Python code for better readability in case you are wondering how to name functions & variables, put comments and indent your code
- **Packages allowed**: numpy, pandas, matplotlib, cvxopt
- **Submission**: Submit report, description and explanation of results in the main PDF and code in .py files.
- Please PROPERLY COMMENT your code in order to have utmost readability
- Please provide the required functions for each problem.
- There shall be NO runtime errors involved.
- There would be PENALTY if any of the above is not followed

Problem 4. Linear SVM dual form.

(**15 Points**)
SVM can be implemented in either primal or dual form. In this assignment, your goal is to implement a linear SVM in dual form using slack variables. **The quadratic program has a box-constraint on each Lagrangian multiplier α_i, $0 \leq \alpha_i \leq C$.**

For doing this you would need an optimizer. Use an optmizer cvxopt which can be easily installed in your environment either through pip or conda. You can refer to the cvxopt document for more details about quadratic programming: https://cvxopt.org/userguide/coneprog.html#quadratic-programming.

1. (**7 Points**) Implement SVM.

   Please at least include the following functions:

   - weight = svmfit(X, y, c) to train your model
   - label = predict(X, weight) to predicts the labels using the model
   - train_accuracy, cv_accuracy, test_accuracy = k_fold_cv(train_data, test_data, k, c) to realize the k-fold cross validation

   You have to use this "hw2data.csv" dataset for this assignment. The labels are in the last column. It is a 2 class classification problem. Split this dataset into 80-20 % split and hold out the last 20% to be used as test dataset. Then, you have to implement k=10 fold cross validation on the first 80% of the data as split above.

   Report the test performance (performance on held out test data) of your trained model for the following cases and provide your reasoning (describing the result is not explanation but you must explain the variation 'why' the result is the way it is):

2. (**2 Points**) Summarize and explain the methods and equations you implemented.

3. (**3 Points**) Vary C in the range [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000] and report the average train, validation and test accuracy as C varies. Provide the plots.

4. (**3 Points**) Explain the train, validation and test performance with C? Reason about it. As a designer, what value of C (and on what basis) would you choose for your model - explain.

Submission: Submit all plots requested and generated while performing hyperparameter tuning and explanations in latex PDF. Submit your program in a file named (hw2_svm.py).

Your answer. 1. The test performance is not as good as train or validation performance but no big differences, shown in Figure 1. Because we train the model with the training data and valid the model with the validation data which is part of the training data, we get a better performance in training and validation accuracy than the test data.
2. In the linear SVM, I use cvxopt to find $\mathbf{w}$. The X and y are training set and labels, the size of which are $n * m$ and $n * 1$. The P, q, G, h for training the linear SVM are:

$$\begin{aligned}
P &= (y * X)(X^\intercal * y^\intercal) \\
q &= -numpy.ones((n, 1)) \\
G &= numpy.vstack((-numpy.eye(n), numpy.eye(n))) \\
h &= numpy.hstack((numpy.zeros(n), numpy.ones(n) * c))
\end{aligned}$$

I implemented:
$svmfit(X, y, c)$ function to find $\mathbf{w}$
$predict(X, w)$ function to predict
$compute\_accuracy(y, y\_pred)$ to compute the accuracy of the predicted data
$k\_fold\_cv(traindata, testdata, k, c)$ to train the data with cross validation
$get\_next\_train\_valid(X\_shuffled, y\_shuffled, k, part\_num)$ to get the next training sets and validation set
$read\_data\_rd(data\_label\_file)$ to read data randomly
$split\_data\_rd(data\_label, test\_percent)$ to separate data and label.

3. The average loss in shown in Figure 1. X axis refers to the value C while Y axis refers to accuracy.

4. From the Figure 1 we find that when C changes, the accuracy doesn't change much because in this case, the data isn't linear separable, so the C doesn't affect the predictions much. As a designer, how to choose C depends on the dataset. If C is too small, the SVM allows more misclassifications, if C is too large, it may lead to overfitting.

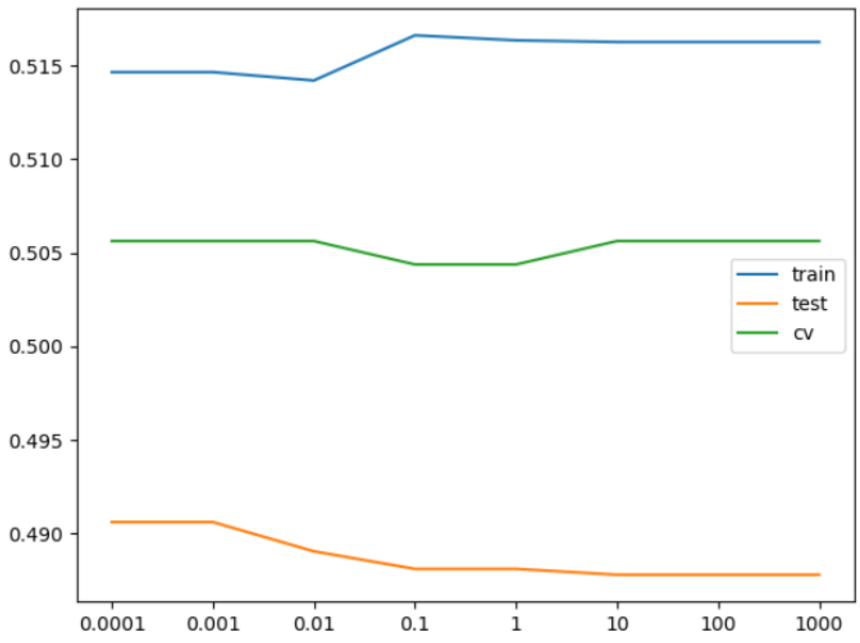

Figure 1: The loss

Problem 5. Kernel SVM.

(**10 Points**)

1. (**7 Points**)Implement a Kernel SVM for a generic kernel. Please at least include the following functions:
   - α = rbf_svm_train(X, y, c, σ) to train your model
   - label = predict(test_X, train_X, train_y, α, σ) to predicts the labels using the model
2. (**3 Points**)Now use the linear SVM implemented in Problem 4 and RBF-SVM (by making use of aforementioned Kernel SVM code in part 1) on the data set ("hw2data.csv"). Implement rbf_svm_train and rbf_svm_predict functions. Use the cross validation similar to Problem 4 to select the best hyper-parameters. Please report the error rate as C and σ vary. Since RBF kernel involves the second hyper-parameter you may consider using heat map to plot it. Then, report validation and test error for each fold - plot it. Compare the accuracies achieved using linear SVM and RBF-SVM, briefly explain the difference.

Submission: Submit all plots requested and generated and explanations in latex PDF. Submit your program in files named (hw2_kernel_svm.py).

Your answer. 2. I test 4 σ with 8 C, and the heat map of error rate is shown in Figure 2. The validation and test error for each fold with the best C and σ is shown in Figure 3. The $C = 1000$ while $\sigma = 0.5$.

The highest test accuracy for RBF SVM in this experiment is more than 98%, but when it comes to linear SVM, the accuracy is less than 50%. The reason is the dataset is not linear

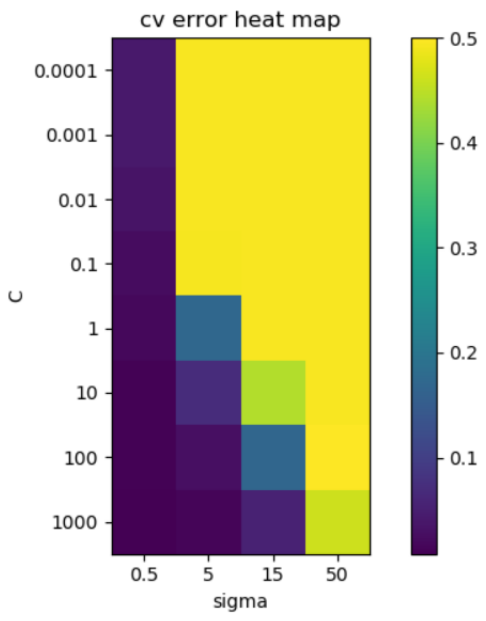

Figure 2: Cross validation error heat map

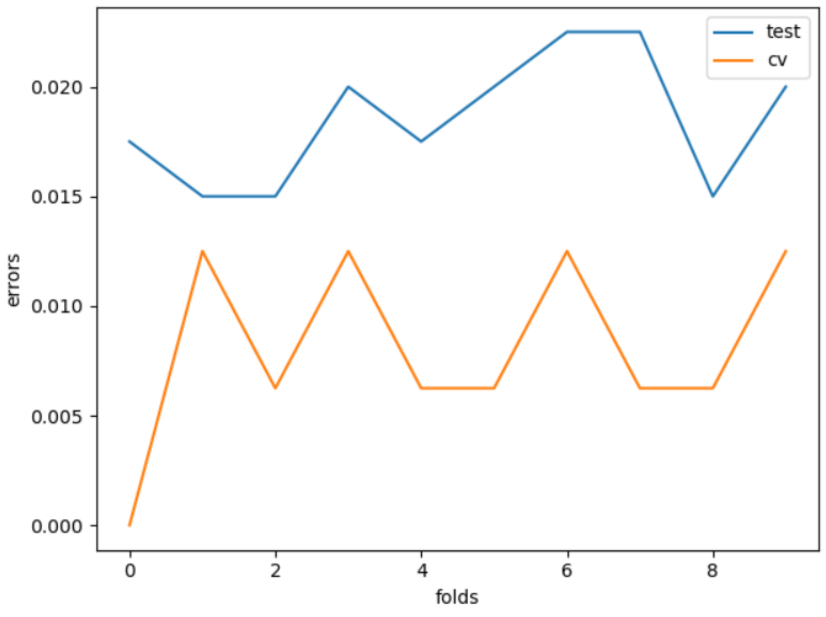

Figure 3: Best test and validation error

separable, so we use the RBF kernel to transform the dataset into a higher dimension to make the SVM perform better.

Problem 6. Multi-class logistic regression.

(**13 Points**) This problem is about multi class classification and you will use MNIST dataset. In the hw2 folder, we have put the mnist files in csv format. There are two .zip files: mnist_train and mnist_test. Use mnist_train for training your model and mnist_test to test. First column in these files are the digit labels.

1. (**7 Points**) Implement a multi-class logistic regression for classifying MNIST digits. Refer the class notes on classifying multi classes. You should use mini-batches for training the model. Save the final trained weights of your model in a file and submit it. Please at least include the following functions:
   - weight = mnist_train(X, y, learning_rate) to train your model
   - label = mnist_predict(weight, X) to predicts the labels using the model
2. (**4 Points**) Report and briefly explain the loss function, how you train update weights and how you train with mini-batches progresses. Please describe the pseudocode of mini-batches progresses and show the derivation of loss and gradient.
3. (**2 Points**) Report confusion matrix and accuracy for the test data.

Submission: Submit all plots requested and generated along with explanations in latex PDF. Submit your program in files named (hw2_multi_logistic.py).

Your answer. 2. For the minibatches, in our case, we have 720 data samples. I shuffle the data and divide the data into 24 minibatches, so there are 30 data samples for each minibatch. In my code, the training data can be seen as 30 small blocks. For example, in the first block, there are 24 elements, I shuffle the elements in it and put these 24 shuffled elements one by one into every minibatches. And the pseudocode is:

```
Algorithm 1: Get minibatch
  Input: Training set: $X$, dimension: (64, 720)
  Output: $Minibatches$, dimension: (24, 64, 30)
1 for i in range(30) do
2     random_seed = range(24)
3     shuffle(random_seed)
4     for j in range(24) do
5         Minibatches[j].append(X[i * 24 + random_seed[j]])
```

I designed an single layer perceptron with Softmax and cross entrop loss, p_i is the output of the Softmax:

$$L(y, p) = -\sum_i y_i \log(p_i)$$

Let a_i be the input of the Softmax, so the derivative of Cross Entropy Loss with Softmax is:

$$\frac{\partial L}{\partial a_i} = p_i - y_i$$

And a_i is defined as:

$$a_i = < \mathbf{w}_i, x_j > + \mathbf{b}_i$$

So the derivation of gradient is:

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{w}_{ij}} &= \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial \mathbf{w}_{ij}} \\
&= (p_i - y_i)x_j \\
\frac{\partial L}{\partial \mathbf{b}_i} &= \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial \mathbf{b}_i} \\
&= p_i - y_i
\end{aligned}$$

3. The confusion matrix and accuracy are shown in Figure 4.

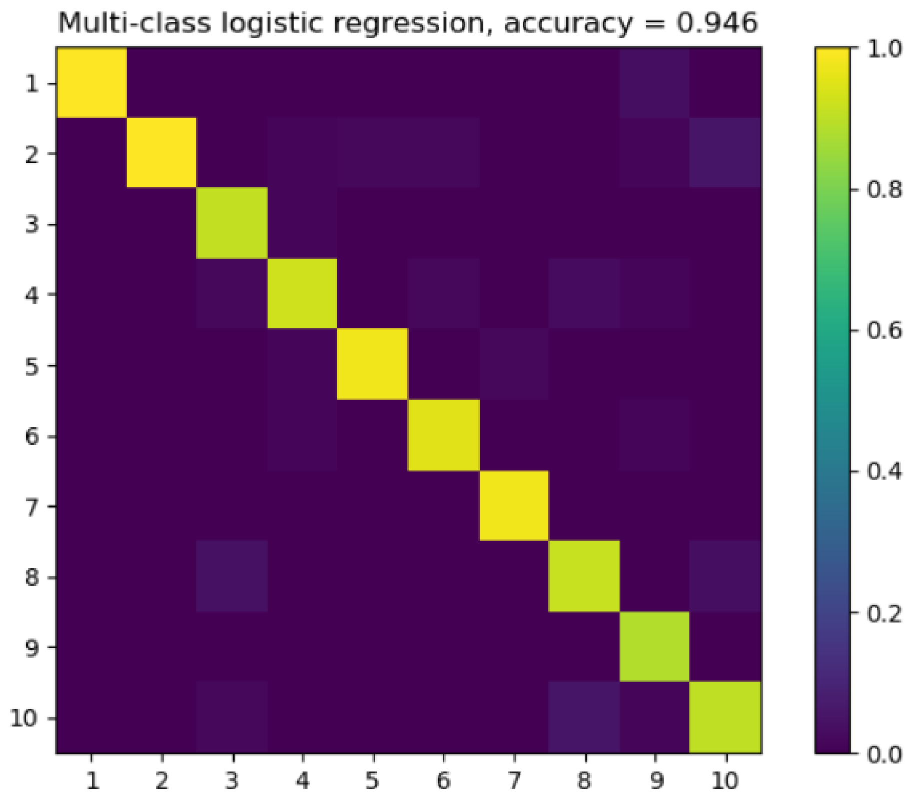

Figure 4: Confusion matrix and accuracy

Problem 7. Multi-class SVM classifier.

(**10 Points**) This problem is about multi-class classification using SVM and you will use MNIST dataset.

One way to let binary classifier be capable to deal with multi-class classification task is One v.s. All strategy. The pseudocode is shown below:

1. (**7 Points**) Implement a multi-class SVM for classifying MNIST digits using One v.s. All strategy. Refer the class notes on classifying multi classes. You should use kernel SVM

Algorithm 2: One-versus-All

```
   Input: Training set: $S = (\mathbf{x}_i, y_i)$, Binary classification algorithm $A$
   Output: The multicalss hypothesis defined by $h(\mathbf{x}) \in \text{argmax}_{i\in\mathcal{Y}}\ h_i(\mathbf{x})$
1  for each $i \in \mathcal{Y}$ do
2      $S_i \leftarrow (\mathbf{x}_j, (-1)^{\mathbb{1}[y_j \neq i]})$.
3      $h_i \leftarrow A(S_i)$
4  return $h(\mathbf{x}) \in \text{argmax}_{i\in\mathcal{Y}}\ h_i(\mathbf{x})$
```

implemented in Problem 5. Save the final trained weights of your model in a file and submit it.

- weights = mnist_svm_train(X, y, learning_rate) to train your model
- label = mnist_svm_predict(weights, X) to predicts the labels using the model

2. (**3 Points**) Report confusion matrix and accuracy for the test data. Compared with the result in Problem 6, does the performance improved? Please explain why or why not.

Submission: Submit all plots requested and generated along with explanations in latex PDF. Submit your program in files named (hw2_multi_svm.py).

Your answer. 3. The confusion matrix and accuracy are shown in Figure 5.

In my case, the C and σ of multi-class SVM are 1 and 0.5. The multi-class SVM performs better than the multi-class Softmax regression. I think the reason is I only design one layer of perceptron of the Softmax regression, and for the multi-class SVM, there are 10 predictors which can help the final prediction.

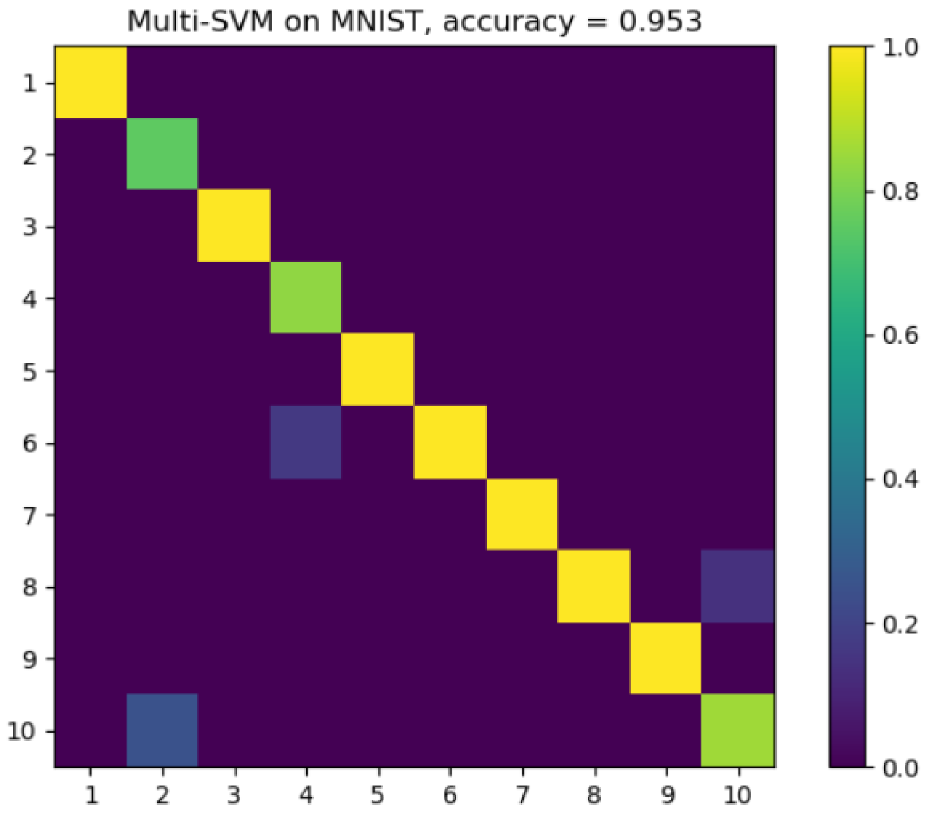

Figure 5: Confusion matrix and accuracy