# Programming Homework 1 Binary Classification

Yuheng ZHAO

October 2022

## 1 Contents & File Structure

According to the requirements, this program includes:

1. **binary_classification.py**, source code of the image classification pipeline. We imply function **runModel** and by configuration, supports two types of classifier, logistic regression or the SVM. Multiple step sizes are included for detailed analysis.

2. **ploy.py** is used to plot the results of the classifier, including losses of each epoch (iteration) with different hyper parameters.

3. **./img** folder contains the out put of **plot.py**.

4. **/log** folder stores the log data for the training process, including the average loss and accuracy of each iteration. The log files are organized by model type and hyper parameters, see **./log/README.md** for more details. In this approach, epoch size is set to 20, and the last piece of the log data contains the final accuracy of the model.

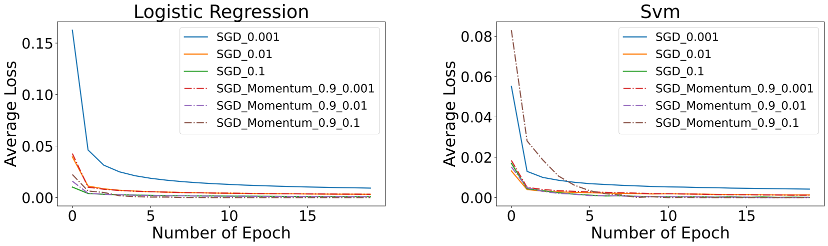
## 2 Results

In this part we illustrate the result of model training, including the average loss of each epoch and the final accuracy. You can find the data in **./log** folder.

### 2.1 Losses and accuracy

We report average loss after each epoch of the training data. Figure 1 shows the change of loss in each epoch. With more iterations of the whole training data set, the loss of each iteration decrease and finally converge at a low value.

Table 1 shows the final accuracy of the model after 20 iterations, as the binary classification of hand written 0s and 1s is highly distinguishable, all the approaches reach a high level of accuracy.



(a) Logistic Regression

(b) Support Vector Machine

Figure 1: Average loss of each epoch using different optimizer and step size.

| | Logistic Regression | SVM |
|---|---|---|
| SGD & lr=0.001 | 99.8582% | 99.9527% |
| SGD & lr=0.01 | 99.9054% | 99.9527% |
| SGD & lr=0.1 | 99.9054% | 99.8582% |
| SGD & m=0.9 & lr=0.001 | 99.9054% | 99.9527% |
| SGD & m=0.9 & lr=0.01 | 99.9054% | 99.9054% |
| SGD & m=0.9 & lr=0.1 | 99.9527% | 99.9054% |

Table 1: Final accuracy of the model

## 2.2 SGD & SGD-Momentum

According to general knowledge, SGD with high momentum makes the decrease of loss faster. In our project, we set momentum = 0 or 0.9 to simulate SGD and SGD-Momentum respectively. As we see in Figure 1, when using SGD-Momentum optimizer (dotted lines), the loss decrease faster in the training process.

## 2.3 Step Size

Generally, larger step size accelerate the converge process, but may cause loss to increase if step size is too large. In this project, we test learning rate 0.1, 0.01 and 0.001 to compare the different outcomes.

Generally, as we can see in Figure 1, larger step size (e.g., 0.1) do cause a fluctuation in the curve compared to the smaller one (e.g., 0.001). And with SGD-Momentum, higher step size's impact becomes more clear as we can see distinguishable flucuation in the curve in both Figure 1(a) and Figure 1(b).