# CSE574 Introduction to Machine Learning
# Programming Assignment 3
# Classification and Regression

Haowei Zhou/haoweizh@buffalo.edu

## 1. Implement Logistic Regression

### 1.1 Result

In this task, I implement 10 classifier from 0 to 9 to classify different digits. For each input image, we have to test for every classifier and choose the one which has the highest value as result.

The training set, validation set and testing set accuracy is below:

*Training set Accuracy:86.28%*
*Validation set Accuracy:85.35%*
*Testing set Accuracy:85.4%*

And the training set's total error is below:

| | Training set error |
|---|---|
| Error1 | 0.0207859810256 |
| Error2 | 0.0214371126371 |
| Error3 | 0.0622820361977 |
| Error4 | 0.0750297897609 |
| Error5 | 0.0453666468007 |
| Error6 | 0.0833743957504 |
| Error7 | 0.0345221003458 |
| Error8 | 0.0442047307379 |
| Error9 | 0.069314718056 |
| Error10 | 0.0962632480608 |

### 1.2 Analysis

From the table and chart above, we can conclude that the training set's total error is almost the same as the testing set's total error. Because the model is neither over-fitting nor under-fitting so the two curves are similar. So, we can predict that is the model is over-fitting, the error for testing data's total error will be much larger than training data's total error. On the other hand, if this model is under-fitting, both the testing data's total error and training data's total error will be very large.

## 2. Implement SVM

## 2.1 Result

In this project, I classify the MNIST dataset using SVM and I modify some parameters, the result is below:

| | Training set Accuracy | Validation set Accuracy | Testing set Accuracy |
|---|---|---|---|
| Kernel = linear | 97.286% | 93.64% | 93.78% |
| Gamma = 1 | 100.0% | 15.48% | 17.14% |
| Default | 94.294% | 94.02% | 94.42% |
| C = 1 | 94.294% | 94.02% | 94.42% |
| C = 10 | 97.132% | 96.18% | 96.1% |
| C = 20 | 97.952% | 96.9% | 96.67% |
| C = 30 | 98.372% | 97.1% | 97.04% |
| C = 40 | 98.706% | 97.23% | 97.19% |
| C = 50 | 99.002% | 97.31% | 97.19% |
| C = 60 | 99.196% | 97.38% | 97.16% |
| C = 70 | 99.34% | 97.36% | 97.26% |
| C = 80 | 99.438% | 97.39% | 97.33% |
| C = 90 | 99.542% | 97.36% | 97.34% |
| C = 100 | 99.612% | 97.41% | 97.4% |

## 2.2 Difference between kernel and radial basis

According to the result above, we can see the training dataset accuracy of using linear kernel method is higher than radial basis method, however, the validation set accuracy and testing set accuracy is slightly lower than radial basis method.

The function of kernel function is to mapping inputs to an higher dimensional space, if we use linear kernel method, I think we actually don't map inputs to higher dimensions. So we can conclude it is suitable for situations that data features is much larger than compared to the training sample.

In this project, the number of features is 716 and the number of training sample is 50000, so using linear kernel method is not appropriate. The feature space of RBF kernel has infinite number of dimensions, so it can classify MNIST better than using linear kernel method.

## 2.3 Influence of gamma setting

As we can see, if I set gamma = 1, the training set accuracy is 100% while the validation set accuracy and testing set accuracy is just less than 20%. So we can conclude that the training set is over-fitting. The default gamma value is equal to 1/n_features. The default gamma setting is much smaller than 1. Actually, the gamma is used as similarity measure between different data. If gamma is too big, which means the variance of Gaussian function is too small, so the data looks quite similar if they are close to each other.

## 2.3 Influence of C

C means the penalty parameter of the error term. As we can see in the result, as C increases, the

training set accuracy, validation set accuracy, testing set accuracy is increasing. When C set too small, for example, C equals to one, the training data set is under-fitting. Even if C equals to 100, the dataset is not over-fitting and when C greater than 40 and less than 100, the testing set accuracy is not changing too much. So we can get the best accuracy if we set C to 100 in this project.

# 3. Implement Multi-Class Logistic Regression

## 3.1 Result

This task is a little bit of different from the first task, In first task, we implement 10 sigmoid function to classify the MNIST data set, In this task, however, we only need to implement one classifier. In this model, we need 7160 parameters, which is 10 times of one of classifiers in first task. And In this task, we only have one error for this classifier rather than 10 error like the first task.

The training set, validation set and testing set accuracy is below:

*Training set Accuracy:93.484%*
*Validation set Accuracy:92.35%*
*Testing set Accuracy:92.6%*

And the training set's total error is below:

11792.5047108

## 3.2 Analysis

In this task , the training set accuracy, validation set accuracy, testing set accuracy is also almost the same. And the training set's total error and testing set's total error is also almost the same. So we can conclude that this model is neither over-fitting nor under-fitting.

## 3.3 Comparing to one-vs-all strategy

Comparing to the one-vs-all strategy, we can see the multi-class strategy is better and it has higher training set accuracy, validation set accuracy and testing data accuracy. In one-vs-all strategy, we training each classifier using the same digit, so this classifier can only “know” how similar the input image to this digit is. However, in multi-class strategy, the classifier is trained by different digits and it can “know” it compares each digits and get the most similar one. So multi-class logistic regression classifier is better than one-vs-all logistic regression.