DEEP LEARNING

**Assignment 1**

Linear and Logistic Regression

horizontal line

# Linear Regression

### Explanation:

* 1. Solves a regression problem (i.e maps input features to a continuous output scalar, through generating a polynomial f(Xs) that fits the data)

### Hyperparameters

* 1. Learning rate
  2. Optimizer used
  3. Epochs (not tuned but we could have tuned it)

### Loss functions

* 1. We can not use the “Accuracy” metrics for measuring Linear regression performance as Linear regression uses linear activation function with only 1 neuron.
  2. Metric used is the mean squared error.

### 

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Optm | RMSprop | | | Adagrad | | | Adam | | |
| LR | 0.1 | 0.01 | 0.001 | 0.1 | 0.01 | 0.001 | 0.1 | 0.01 | 0.001 |
| Val Loss | 3410 | 3600 | 3606 | 3648 | 440 | 153 | 3464 | 3728 | 3717 |
| Figure | [fig\_R1](#_lbda5f3ipscx) | [fig\_R2](#_iy22zzms5ua5) | [fig\_R3](#_qyxf3e49205t) | [fig\_G1](#_k25dxvd9m8ca) | [fig\_G2](#_br47su6klr6) | [fig\_G3](#_963z768n07kl) | [fig\_A1](#_oao3nyold09t) | [fig\_A2](#_29abjf9gsgru) | [fig\_A3](#_ttgyg7t2c6s0) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Optm | SGD | | | | | |
| LR | 0.01 | | | 0.001 | | |
| Mmtm | - | 0.01 | 0.001 | - | 0.01 | 0.001 |
| Val Loss | 5.8 | 6.002 | 5.7 | 6.32 | 6.25 | 6.9 |
| Train loss | 0.0124 | 0.0102 | 0.0101 |  |  |  |
| Figure | [fig\_S1](#_epkp94tw03zw) | [fig\_S2](#_e0jvm39y6wlt) | [fig\_S3](#_e0jvm39y6wlt) | [fig\_S4](#_e0jvm39y6wlt) | [fig\_S5](#_e0jvm39y6wlt) | [fig\_S6](#_e0jvm39y6wlt) |

### 

### Problems in Models

#### Overfitting: (RMSprop, Adagrad, Adam)

solved by early stopping (i.e decreasing the number of epochs) or by reducing the model complexity either by removing some layer/neurons -which doesn’t apply in our case- or by adding dropout in features layer. Also we can add more data to our model. Other methods exist that don’t apply to 1 neuron case.

#### Underfitting:

Solved by increasing the model complexity by adding more neurons/layers -which doesn’t apply to our case-, or by transforming the model and increasing the space dimension (i.e adding , Cos(X), etc), and finally we can increase the number of epochs.

#### Overshooting:

Overshooting the optimal value of the parameters is solved by decreasing  
 the learning rate.

# Logistic Regression

### Explanation:

* 1. Solves a regression problem (i.e maps input features to a continuous output scalar, through generating a polynomial f(Xs) that fits the data)

### Hyperparameters

* 1. Learning rate
     1. 0.1, 0.01, 0.001
  2. Optimizer
     1. SGD, RMSprop, Adagrad, Adam
  3. Pre-processing
     1. MinMaxScalar, StandardScaler
  4. Parameter Initializers
     1. RandomNormal, RandomUniform
  5. Loss function
     1. Binary- Cross Entropy, Categorical Hinge, Mean Squared Error
  6. Epochs
     1. 500, 1000
  7. Batch\_size (not tuned but we could have tuned it)
  8. Artificial features with and without interaction\_only.

### This is the data collected, this part could be skipped since the analysis part contains the important details of these tests. (A total of 132 tests were executed)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OPT | SGD | | | | | | | |
| LR | 0.01 | | | | 0.001 | | | |
| PI | RU | | RN | | RU | | RN | |
| Epochs | 500 | 1000 | 500 | 1000 | 500 | 1000 | 500 | 1000 |
| Val Acc-B | 0.7895 | 0.7895 | 0.7895 | 0.7895 | 0.8158 | 0.7895 | 0.8158 | 0.7895 |
| Val Acc-C | 0.8421 | 0.8947 | 0.8421 | 0.8947 | 0.6842 | 0.5526 | 0.5263 | 0.7895 |
| Val Acc-M | 0.7895 | 0.7895 | 0.7895 | 0.7895 | 0.7632 | 0.8158 | 0.5263 | 0.7895 |
| Train Acc-B | 0.8502 | 0.8502 | 0.8458 | 0.8502 | 0.8414 | 0.8414 | 0.8458 | 0.8414 |
| Train Acc-C | 0.8722 | 0.8899 | 0.8767 | 0.8899 | 0.6652 | 0.5374 | 0.5374 | 0.8414 |
| Train Acc-M | 0.8458 | 0.8546 | 0.8414 | 0.8546 | 0.7797 | 0.8370 | 0.5374 | 0.8458 |
| Test Acc-B | 0.87 | 0.87 | 0.87 | 0.87 | 0.84 | 0.87 | 0.84 | 0.87 |
| Test Acc-C | 0.87 | 0.87 | 0.87 | 0.87 | 0.66 | 0.61 | 0.61 | 0.60 |
| Test Acc-M | 0.87 | 0.87 | 0.87 | 0.87 | 0.79 | 0.84 | 0.61 | 0.84 |
| Figure | [fig\_1](#_lzx14o7baae2) | [fig\_2](#_l8vpwodnkyra) | [fig\_3](#_ngktc9weur6k) | [fig\_4](#_l9m22qnu50is) | [fig\_5](#_3rafhuyhaxkw) | [fig\_6](#_6qnfuw6g06oh) | [fig\_7](#_nc6oh9x0vja7) | [fig\_8](#_m849x93anlgt) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OPT | RMSProp | | | | | | | | | | | |
| LR | 0.1 | | | | 0.01 | | | | 0.001 | | | |
| PI | RN | | RU | | RN | | RU | | RN | | RU | |
| Epochs | 500 | 1000 | 500 | 1000 | 500 | 1000 | 500 | 1000 | 500 | 1000 | 500 | 1000 |
| Val Acc-B | 0.78 | 0.78 | 0.78 | 0.81 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 |
| Val Acc-C | 0.89 | 0.89 | 0.84 | 0.81 | 0.89 | 0.86 | 0.89 | 0.86 | 0.89 | 0.86 | 0.89 | 0.86 |
| Val Acc-M | 0.86 | 0.86 | 0.86 | 0.86 | 0.78 | 0.78 | 0.81 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 |
| Train Acc-B | 0.84 | 0.84 | 0.84 | 0.84 | 0.85 | 0.85 | 0.85 | 0.84 | 0.85 | 0.85 | 0.85 | 0.85 |
| Train Acc-C | 0.89 | 0.88 | 0.88 | 0.87 | 0.90 | 0.88 | 0.90 | 0.88 | 0.88 | 0.88 | 0.88 | 0.88 |
| Train Acc-M | 0.9 | 0.9 | 0.9 | 0.9 | 0.85 | 0.85 | 0.85 | 0.85 | 0.85 | 0.85 | 0.85 | 0.85 |
| Test Acc-B | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 |
| Test Acc-C | 0.87 | 0.87 | 0.79 | 0.89 | 0.87 | 0.89 | 0.87 | 0.89 | 0.87 | 0.87 | 0.87 | 0.87 |
| Test Acc-M | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OPT | Adagrad | | | | | | | | | | | |
| LR | 0.1 | | | | 0.01 | | | | 0.001 | | | |
| PI | RN | | RU | | RN | | RU | | RN | | RU | |
| Epochs | 500 | 1000 | 500 | 1000 | 500 | 1000 | 500 | 1000 | 500 | 1000 | 500 | 1000 |
| Val Acc-B | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.84 | 0.84 |
| Val Acc-C | 0.89 | 0.89 | 0.89 | 0.89 | 0.78 | 0.84 | 0.81 | 0.84 | 0.87 | 0.81 | 0.76 | 0.78 |
| Val Acc-M | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.76 | 0.76 | 0.78 | 0.81 |
| Train Acc-B | 0.84 | 0.85 | 0.85 | 0.85 | 0.85 | 0.85 | 0.85 | 0.85 | 0.83 | 0.83 | 0.81 | 0.81 |
| Train Acc-C | 0.89 | 0.89 | 0.89 | 0.89 | 0.85 | 0.85 | 0.85 | 0.85 | 0.81 | 0.83 | 0.77 | 0.80 |
| Train Acc-M | 0.85 | 0.85 | 0.85 | 0.85 | 0.84 | 0.83 | 0.85 | 0.83 | 0.78 | 0.84 | 0.77 | 0.82 |
| Test Acc-B | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.82 | 0.82 | 0.84 | 0.82 |
| Test Acc-C | 0.87 | 0.87 | 0.87 | 0.87 | 0.84 | 0.87 | 0.84 | 0.87 | 0.82 | 0.82 | 0.82 | 0.82 |
| Test Acc-M | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.82 | 0.82 | 0.82 | 0.82 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OPT | Adam | | | | | | | | | | | |
| LR | 0.1 | | | | 0.01 | | | | 0.001 | | | |
| PI | RN | | RU | | RN | | RU | | RN | | RU | |
| Epochs | 500 | 1000 | 500 | 1000 | 500 | 1000 | 500 | 1000 | 500 | 1000 | 500 | 1000 |
| Val Acc-B | 0.78 | 0.78 | 0.78 | 0.81 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 |
| Val Acc-C | 0.84 | 0.84 | 0.84 | 0.86 | 0.89 | 0.89 | 0.89 | 0.86 | 0.89 | 0.89 | 0.89 | 0.89 |
| Val Acc-M | 0.81 | 0.89 | 0.89 | 0.86 | 0.78 | 0.78 | 0.81 | 0.78 | 0.78 | 0.78 | 0.78 | 0.78 |
| Train Acc-B | 0.86 | 0.84 | 0.86 | 0.83 | 0.85 | 0.85 | 0.84 | 0.84 | 0.85 | 0.85 | 0.85 | 0.84 |
| Train Acc-C | 0.88 | 0.86 | 0.86 | 0.87 | 0.89 | 0.89 | 0.89 | 0.90 | 0.88 | 0.89 | 0.88 | 0.89 |
| Train Acc-M | 0.91 | 0.88 | 0.88 | 0.9 | 0.85 | 0.85 | 0.86 | 0.85 | 0.85 | 0.85 | 0.85 | 0.85 |
| Test Acc-B | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 |
| Test Acc-C | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 |
| Test Acc-M | 0.84 | 0.87 | 0.84 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 | 0.87 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artificial Features | | | | |
| Degree | Poly Degree 2 | | Poly Degree 3 | |
| Interaction | Interaction only | All features | Interaction only | All features |
| Features total | 92 | 105 | 378 | 560 |
| Effect on best test Acc | Lower accuracy 0.89 -> 0.87 | Lower accuracy 0.89 -> 0.87 | Lower accuracy 0.89 -> 0.84 Higher on training | Lower accuracy 0.89 -> 0.87 |
| Effect on best validation Acc | Lower accuracy 0.89 -> 0.78 | Higher accuracy 0.89 -> 0.92 | Lower accuracy 0.89 -> 0.81 Higher on training | Lower accuracy 0.89 -> 0.81 Over-fitting |

### Problems in Models

#### Overfitting:

solved by early stopping (i.e decreasing the number of epochs) or by reducing the model complexity either by removing some layer/neurons -which doesn’t apply in our case- or by adding dropout in features layer. Also we can add more data to our model. Other methods exist that don’t apply to 1 neuron case.

#### Underfitting:

Solved by increasing the model complexity by adding more neurons/layers -which doesn’t apply to our case-, or by transforming the model and increasing the space dimension (i.e adding , Cos(X), etc), and finally we can increase the number of epochs.

#### Overshooting:

Overshooting the optimal value of the parameters is solved by decreasing  
 the learning rate.

### Performance Analysis

* 1. Best Accuracy
     1. Validation : 0.89 (colored green in the tables above)
     2. Train : 0.91 (colored green in the tables above)
     3. Test : 0.89 (colored green in the tables above)
  2. Hyperparameters effect
     1. We can see that Categorical Hinge gave the highest accuracy with different optimizers, followed by Mean Squared Error which gave slightly better performance Categorical Hinge when using learning rate of 0.1 in Adam and RMSProp.
     2. Using 500 / 1000 epochs gave different results based on the loss function and optimizers used but overall we can see that using 500 epochs is more efficient.

However in SGD, using 1000 epochs is more efficient.

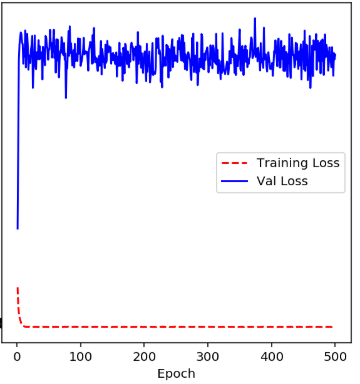
* + 1. Highest performance for validation, train and test sets was when using RMSProp, also for both validation and training data Adam achieved the same results.
    2. Parameter initialization didn’t affect much, but gave the best for tests with Random uniform and best for training with random normal.
    3. Best 2 optimizers were Adam and RMSprop, and worst was SGD. Adagrad achieved the same best result on validation set only.
  1. Artificial features
     1. Added polynomials of 2nd and 3rd degrees and 1st degree would only result in 1 new feature which most probably won’t affect the result
     2. Validation and training accuracy increased only when using polynomial of 2nd degree with 105 features

# Appendix

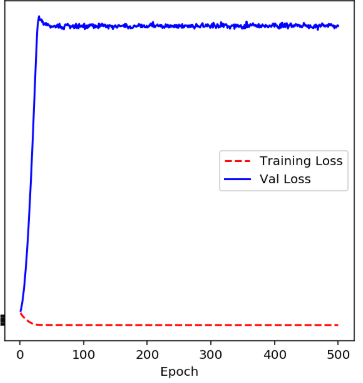
## Linear Regression Sample Run

### RMSprop

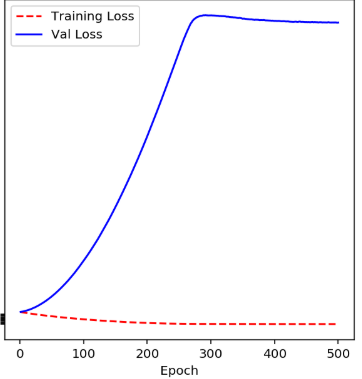
#### R\_LR 0.1



#### R\_LR 0.01

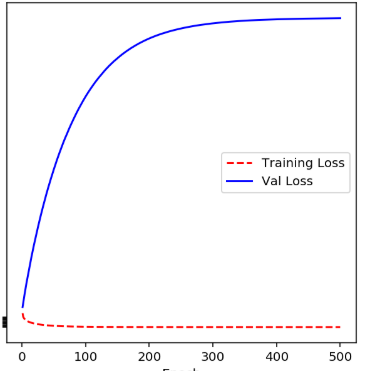


#### R\_LR 0.001

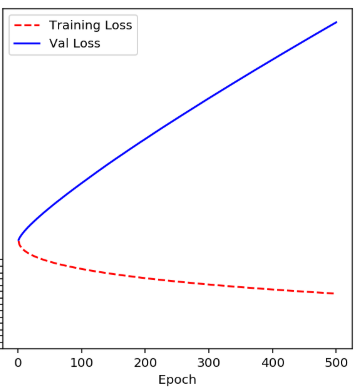


### Adagrad

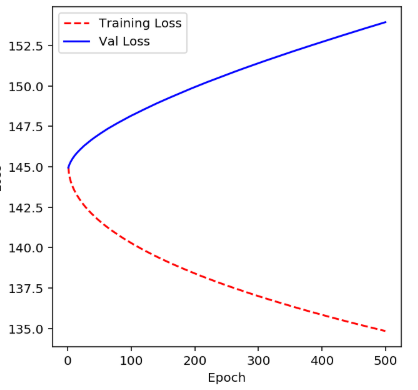
#### Adagrad LR 0.1



#### Adagrad LR 0.01

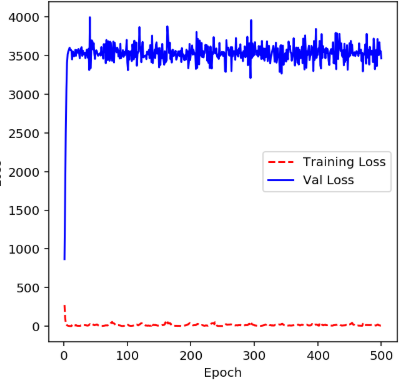


#### Adagrad LR 0.001

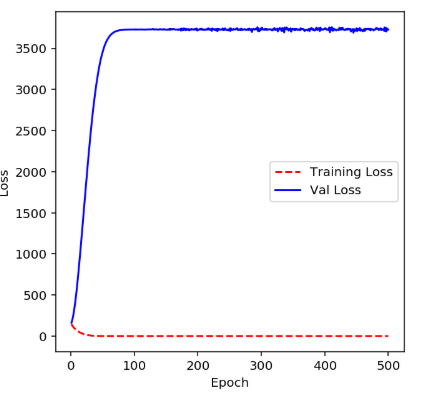


### Adam

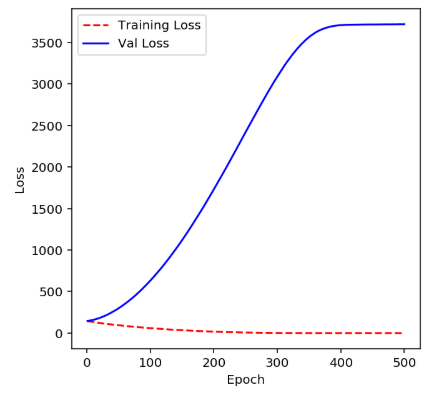
#### Adam LR 0.1



#### Adam LR 0.01

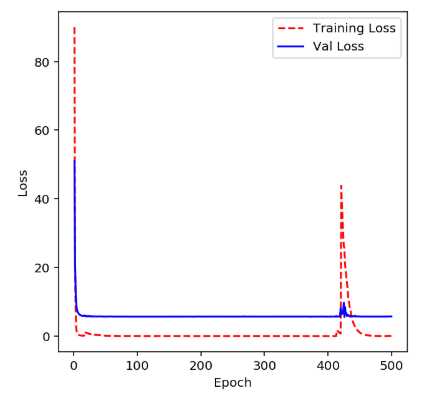


#### Adam LR 0.001

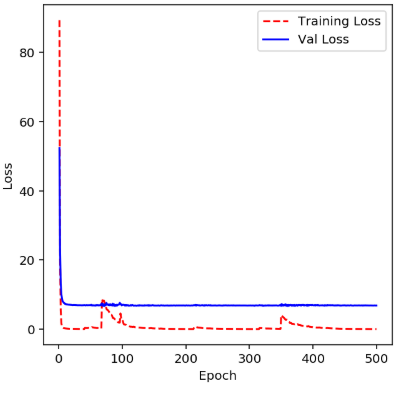


### SGD

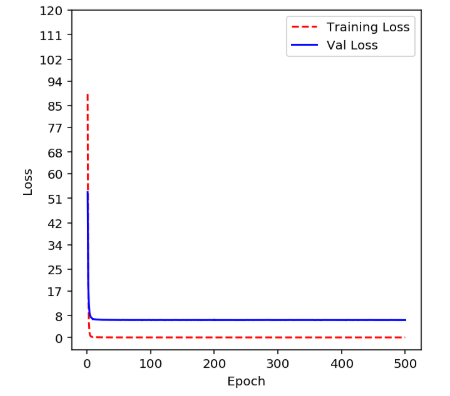
#### SGD LR 0.01, no Momentum



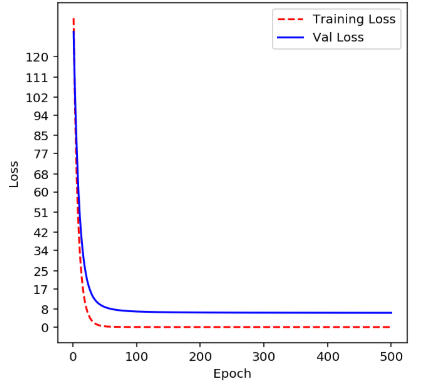
#### SGD LR 0.01, Momentum 0.01



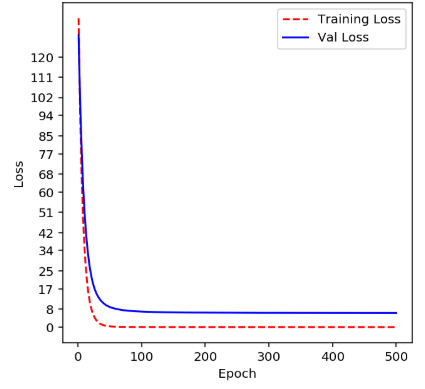
#### SGD LR 0.01, Momentum 0.001



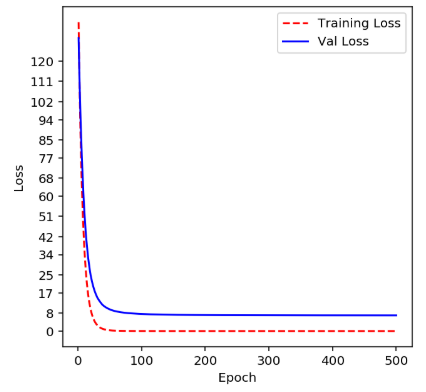
#### SGD LR 0.001, no Momentum



#### SGD LR 0.001, Momentum 0.01



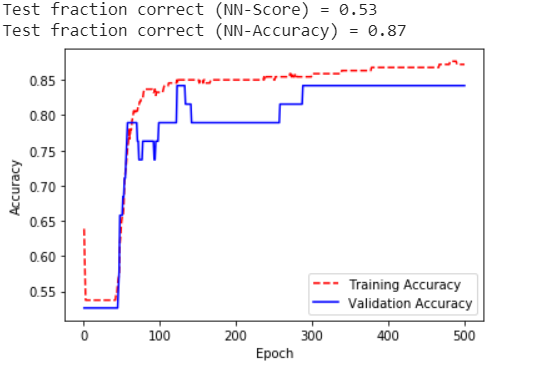
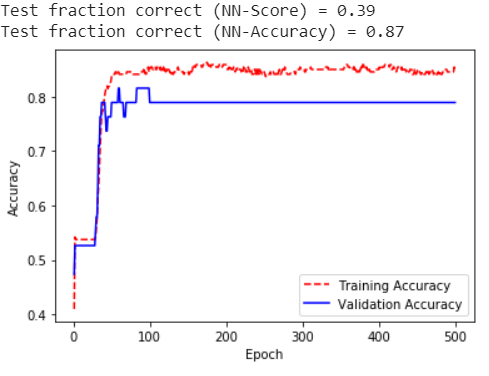
#### SGD LR 0.001, Momentum 0.001

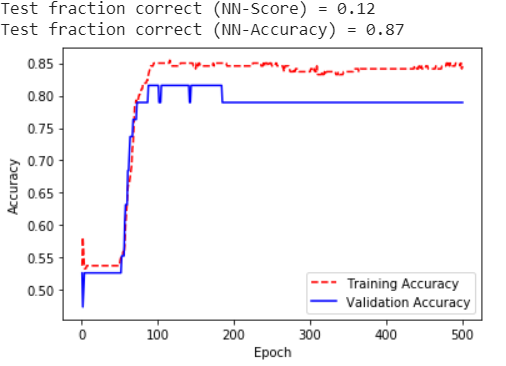


## Logistic Regression

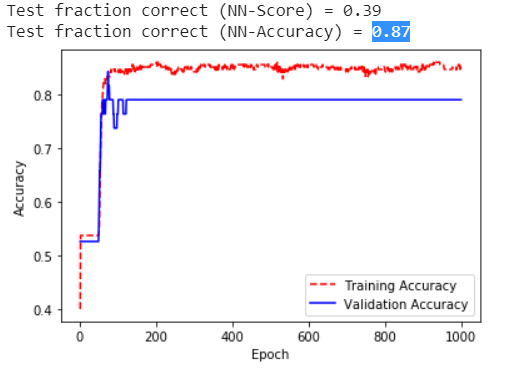
### SGD-SIGMOID

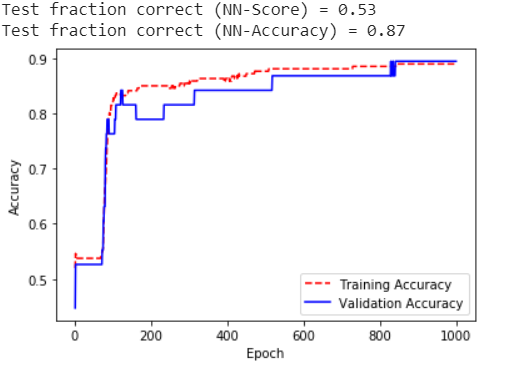
#### 0.01-RU-500

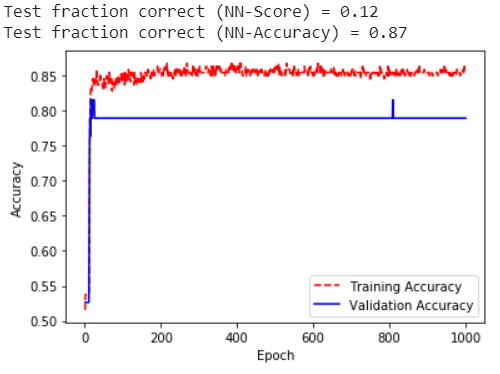




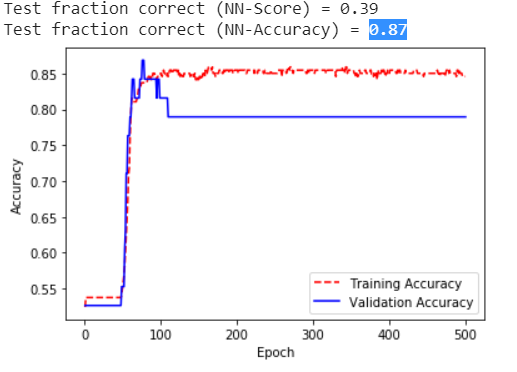
#### 0.01-RU-1000

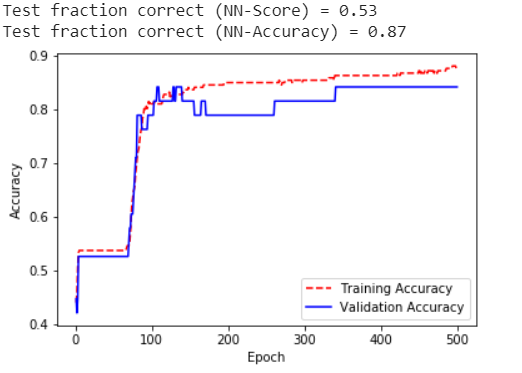


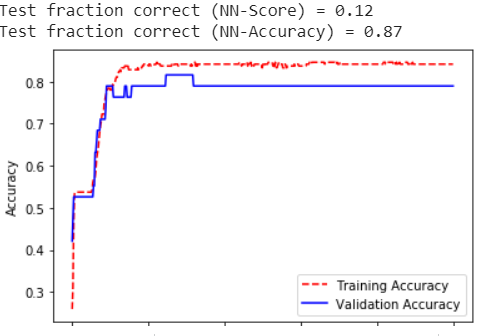




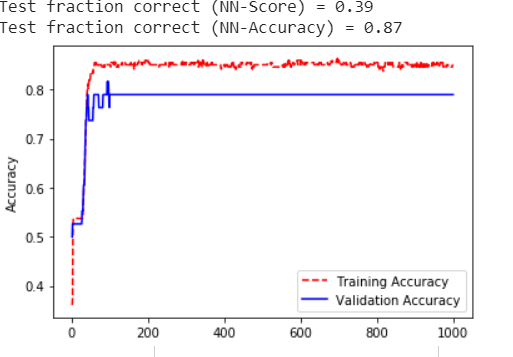
#### 0.01-RN-500

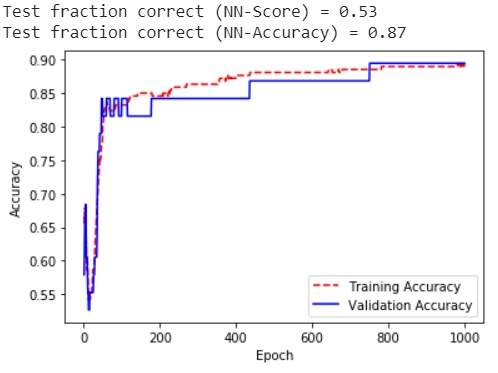


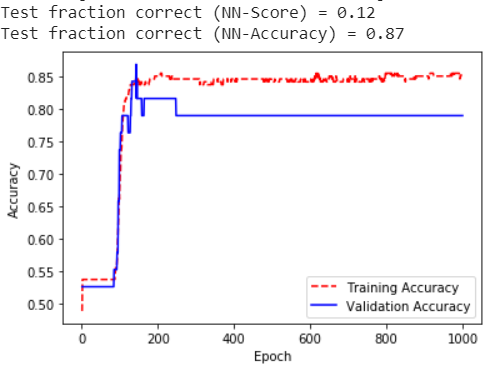




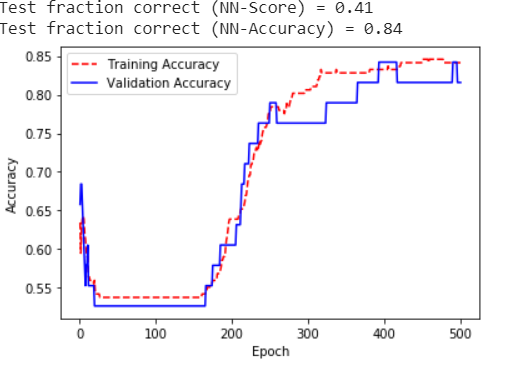
#### 0.01-RN-1000

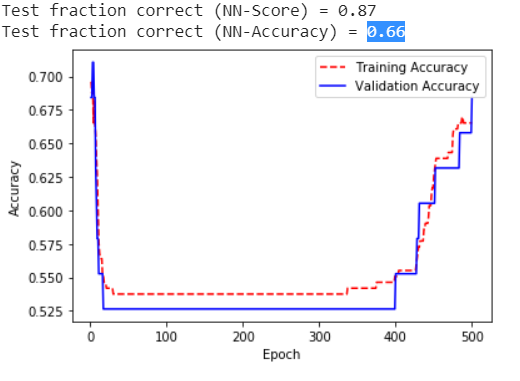


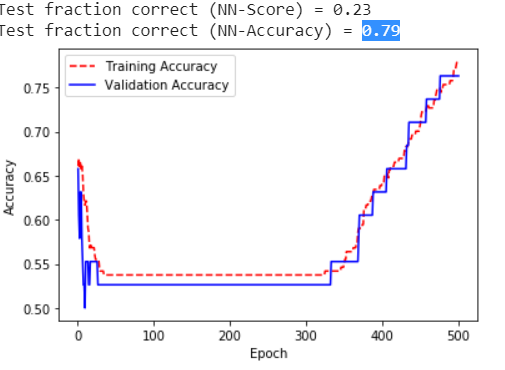




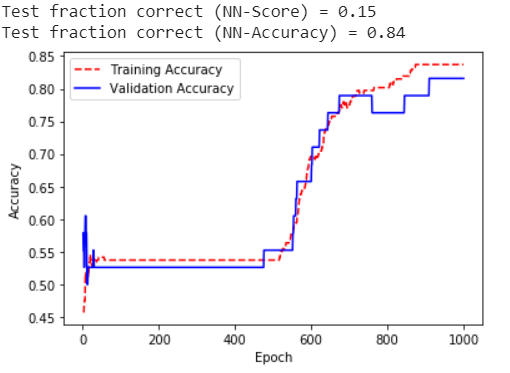
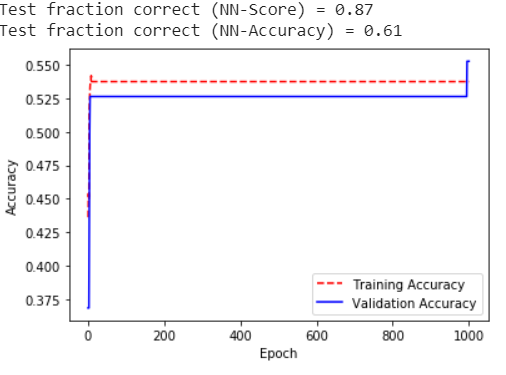
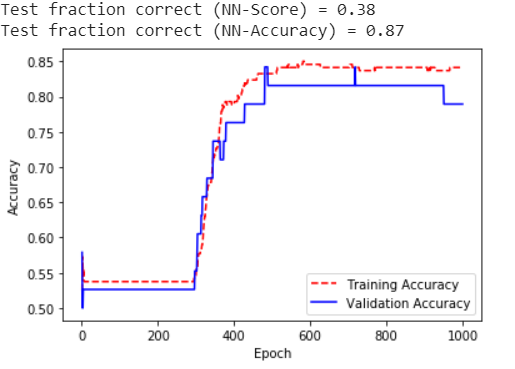
#### 0.001-RU-500



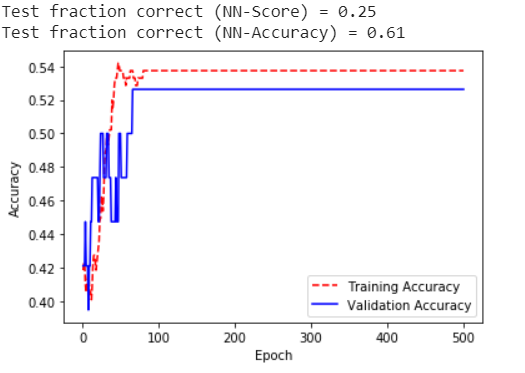
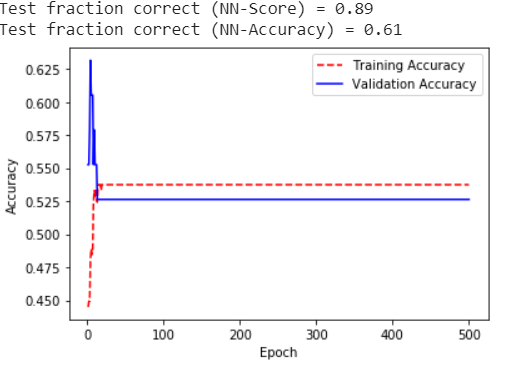
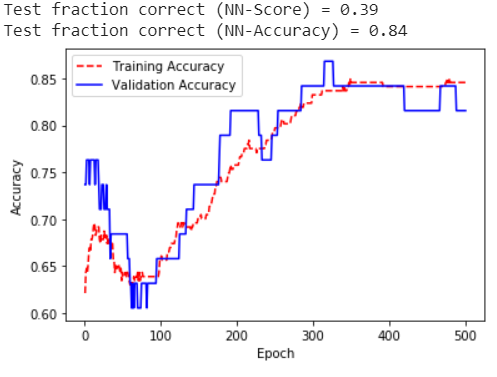




#### 0.001-RU-1000



#### 0.001-RN-500



#### 0.001-RN-1000

