# Machine Learning - Homework 3 Report

Lukas Johannes Ruettgers (2023372403)

October 18, 2023

## 1 Objective

The experiment's objective is to examine and evaluate the performance of a Multilayer Perceptron (MLP) on a multi-feature dataset. The dataset originates from https://www.kaggle.com/c/widsdatathon2020/data and contains medical information of patients at the Intensive Care Unit (ICU) of US-American hospitals. The patients $x_i$ were classified into two groups depending on their survival $s_i$ during their stay at the ICU.

The MLP is supposed to estimate the probability of survival ($s_i = 1$) or death ($s_i = 0$) respectively for any patient $x_i$ represented by the feature schema of the given dataset.

## 2 Experiment setup

### 2.1 Model

Multilayer Perceptrons are feedforward artificial neural networks comprise of an input layer, an output layer and at least one hidden layer. Each nodes $j$ receives values from all nodes $i$ in the previous layer. Node $j$ weighs the input of each node $i$ with a weight $w_{ij}$ and feeds the weighted sum into an activation function $f$, which is usually nonlinear and therefore constitutes one of the model's core advantages over linear models in its ability to represent more complex data patterns. Common choices are sigmoid, tanh and ReLU. The output of the activation function is then forwarded to each node in the next layer.
In the last layer finally yields an estimation $\hat{y}_j$ of the target value $y_j$ to the corresponding MLP input $x_j$ and the approximation error is determined using a loss function $E$.
The choice of this function depends on the specific learning task, namely whether the model is supposed to regress continuous values or classify between multiple classes.

In our case of binary classification, the estimate of the binary target value $y_j \in \{0, 1\}$ is continuous in $[0, 1]$ and can be interpreted by setting a cutoff-value which defines a hard threshold above which all predictions are interpreted as predicting $y_j = 1$ and vice versa.

To assess the accuracy of $N$ predictions on $K$ classes, the model avails to a provided loss function, which oftenly takes the shape of a mean squared error term

$$E = \frac{1}{N} \sum_{s=1}^{N} (y_s - \hat{y}_s)^2, \tag{1}$$

or a cross entropy loss to compare the probability distribution to the true distribution of class labels

$$E = -\frac{1}{N} \sum_{s=1}^{N} P(\hat{y}_s \text{ predicts } y_s \mid x_s, w_{i,j}). \tag{2}$$

The sole freedom of the MLP to improve its estimation is the adjustment of its weights $w_{ij}$. The gradient $\frac{\partial E}{\partial w_{i,j}}$ now indicates how strongly a weight $w_{i,j}$ contributes to the improvement or deterioration of the loss function and serves as foundation for many optimization

algorithms. One popular optimization method among them is the stochastic gradient descent, which in each training iteration $t$ randomly selects a sample $(x_s, y_s)$ and immediately updates the weights along the gradient direction.

$$w_{i,j}(t+1) = w_{i,j}(t) - \eta \frac{\partial E}{\partial w_{i,j}(t)}. \tag{3}$$

The update intensity is controlled by the parameter $\eta$ called the learning rate. Further adjustments seem to improve the learning procedure in practice. For example, a regularization term $\lambda \sum_{w_{i,j}} |w_{i,j}|$ is frequently added to the loss function $E$, which forces feature selection and thereby serves as an inductive bias towards simple models to avoid overfitting. Moreover, the update term is oftenly extended by a momentum to

$$w_{i,j}(t+1) = w_{i,j}(t) - (1-\alpha)\eta \frac{\partial E}{\partial w_{i,j}(t)} + \alpha(w_{i,j}(t) - w_{i,j}(t-1)) \tag{4}$$

to smoothen the learning curve and in that way reduce oscillations in the learning process. The hyperparameter $\alpha \in (0, 1)$ steers the "inertia" of the learning curve.

### 2.2 Preprocessing

The training dataset and the testing dataset comprises 5000 and 1097 samples respectively. Each sample is described by 108 features subject to binary, numeric and integer domains. While the binary values were solely converted into integers, the values of numeric and integer features were all normalized to $[0, 1]$, since their values varied largely in size and units. This was achieved by linearly rescaling each feature value $x$ to $\frac{x - x_{min}}{x_{max} - x_{min}}$, where $x_{min}$ and $x_{max}$ are the maximum and minimum values of the respective feature.

### 2.3 Hyperparameters

For the experiment, the MLP model provided by the `sklearn.neural_network` package is used. It is initialized with one hidden layer containing 100 nodes. The cutoff-value is straightforwardly chosen to be 0.5. As activation functions, both ReLU and logistic are examined. Moreover, the relation between the number of training iterations and the training, validation and test performance is assessed by incrementally increasing the iterations from 200 to 1000 by 200 for each model configuration. Here, an iteration references an epoch in which each feature vector of the dataset is sampled exactly once. First of all, the experiments alternate the learning rates between $0.1, 0.05, 0.01$ and $0.001$ to obtain a rough picture of the training and testing errors for each activation function. Furthermore, we scale $\alpha$ along $0, 0.1$ and $0.5$ to analyze whether the momentum term not only smoothens the learning curve, but also effectively improves the performance of training protocols with relatively high learning rates.

### 2.4 Cross Validation

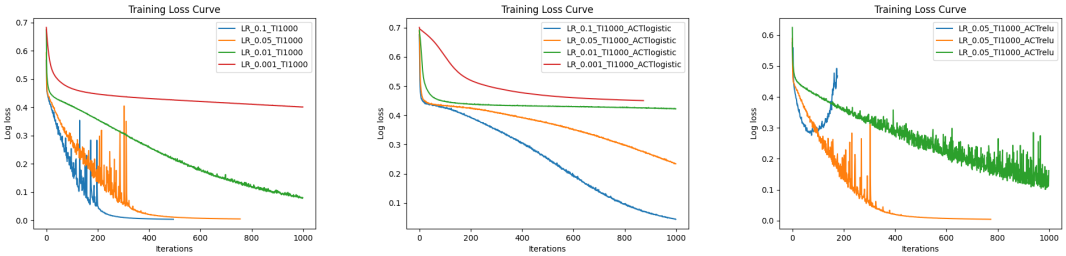
Besides training the model with the entire training dataset, the bias of the models was assessed via 10-fold cross validation (CV). The training data set was split into ten equally sized intervals and the 10-fold CV error was computed as the mean accuracy of the predictions of the models trained on their respective testing interval.

## 3 Results

### 3.1 Training

For the model equipped with the ReLU activation function, the learning curve comparison for different learning rates might indicate on the first glance that the lower learning rates

hinder the model from converging to a low error region and that the higher learning rates of 0.1 and 0.05 are more preferable (Fig. 1a). However, we will see in the later sections that the opposite is the case, since the higher learning rates lead the model to overfitting the training data. We also observe that high oscillations in the learning curve correlate with high learning rates, which meets our expectations since a higher learning rate results in sharper weight change directions. The training loss curve comparison for the MLP




(a) MLP with ReLU activation functions

(b) MLP with logistic activation functions

(c) Influence of momentum on loss curve

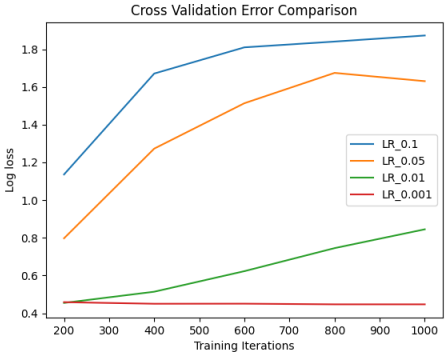
Figure 1: Training error curves for different learning rates (LR) 0.1 (blue), 0.05 (yellow), 0.01 (green) and 0.001 (red). Curve cutoffs represent early learning stops because of convergence. In Figure c), the influence of three different momentum factors $\alpha$ on the loss curve are juxtaposed: 0 (blue), 0.1 (yellow), 0.5 (green).

with logistic activation functions (Fig. 1b) conveys a similar impression on the training rate, which is also disproved later by the validation results. Conversely to the ReLU model, the learning curves preserve their smoothness for increasing learning rates. The similar convergence pattern of the $\eta = 0.001$ learning curve in the ReLU model and the $\eta = 0.001, 0.01$ learning curves in the logistic model strengthen the conjecture that this error region incorporates reasonable and generalized classification hypotheses.

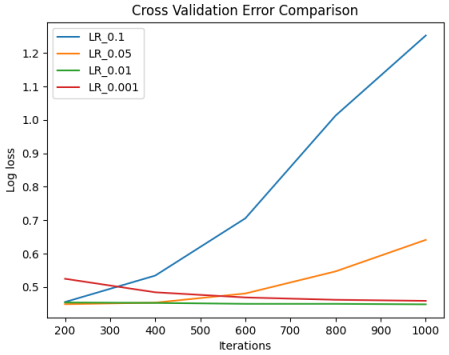
Especially the learning curve of the ReLU model with learning rate 0.05 suffers from oscillations despite a momentum with factor 0.1 is applied. If the momentum is completely omitted, the learning curve even turns sharply, deteriorating the log loss after few iterations. Surprisingly, the application of a momentum with rate 0.5 does not significantly reduce the oscillations in the learning curve in comparison to a momentum of rate 0.1. Though the amplitudes shrink, the length over which the oscillations occur even increases a little (Fig. 1c). An explanation might be that the gradients which evoke such strong oscillations maintain relevant in the weight update term for more iterations and therefore lengthen the learning curve oscillation. On the other side, the test accuracy improves slightly from 0.7247 when using no momentum to 0.7511 when using momentum with scale 0.5.

### 3.2 Cross Validation

The two activation functions cause notable differences regarding their CV error development for increasing training iterations. While the ReLU models seem to logarithmically approach an upper limit for the logarithmic loss, the increase of iterations exarcerbates the error of the logistic model with learning rate 0.1 exponentially. Nonetheless, the more important candidates with low learning rates both reasonably approach the same error level. With a cross validation 10-fold error of 0.4475 for the best ReLU model (Fig. 2a) and 0.4498 for the best logistic model (Fig. 2b), we find no significant dominance of the one activation function over the other when considering the validation error.




(a) MLP with ReLU activation functions




(b) MLP with logistic activation functions

Figure 2: Cross validation errors developments along increasing training iterations (IT). The error is measured as the logarithmic loss for various learning rates (LR): 0.1 (blue), 0.05 (yellow), 0.01 (green) and 0.001 (red).

### 3.3 Testing

A similar impression is generated by the comparison of the testing results. Both the logistic and the ReLU model obtain an overall accuracy of around 0.78 for their best learning rate and training iteration choices (Fig. 3).

| **Model Parameters** | | | **Prediction Results** | | | | |
|---|---|---|---|---|---|---|---|
| ACT | LR | TI | TP | FP | FN | TN | Accuracy |
| ReLU | 0.01 | 200 | 448 | 99 | **134** | **416** | **0.7876** |
| ReLU | 0.01 | 400 | **484** | **63** | 205 | 345 | 0.7557 |
| ReLU | 0.01 | 600 | 461 | 86 | 189 | 361 | 0.7493 |
| ReLU | 0.01 | 800 | 444 | 103 | 169 | 381 | 0.7521 |
| ReLU | 0.01 | 1000 | 464 | 83 | 193 | 357 | 0.7484 |
| ReLU | 0.001 | 200 | 431 | 116 | 128 | 422 | 0.7776 |
| ReLU | 0.001 | 400 | 437 | 110 | 128 | 422 | 0.7830 |
| ReLU | 0.001 | 600 | 444 | 103 | 131 | 419 | 0.7867 |
| ReLU | 0.001 | 800 | **455** | **92** | 141 | 409 | **0.7876** |
| ReLU | 0.001 | 1000 | 434 | 113 | **123** | **427** | 0.7849 |
| Logistic | 0.01 | 200 | 448 | 99 | **134** | **416** | **0.7876** |
| Logistic | 0.01 | 400 | **484** | **63** | 205 | 345 | 0.7557 |
| Logistic | 0.01 | 600 | 461 | 86 | 189 | 361 | 0.7493 |
| Logistic | 0.01 | 800 | 444 | 103 | 169 | 381 | 0.7521 |
| Logistic | 0.01 | 1000 | 464 | 83 | 193 | 357 | 0.7484 |
| Logistic | 0.001 | 200 | 392 | 155 | **116** | **434** | 0.7530 |
| Logistic | 0.001 | 400 | 417 | 130 | 118 | 432 | 0.7739 |
| Logistic | 0.001 | 600 | **428** | **119** | 125 | 425 | 0.7776 |
| Logistic | 0.001 | 800 | 424 | 123 | 126 | 424 | 0.7730 |
| Logistic | 0.001 | 1000 | 425 | 122 | 121 | 429 | **0.7785** |

Figure 3: Test results of the investigated models. Models are categorized by their activation functions (ACT), learning rate (LR) and number of training iterations (IT).

While the learning curves for the logistic model did not show significant dominance of the learning rate 0.001 over the learning rate 0.01 in convergence, the test results depict something different. While the models with learning rate 0.01 achieve their best

performance after few iterations and then only exacerbate their accuracy, this trend is reversed for the models with learning rate 0.001. Models with higher learning rates than 0.01 also showed the same pattern. This may firstly underscore that the models become victim of overfitting for higher training iterations. But secondly, we also observe that the models with higher learning rates achieve not as balanced predictions as their counterparts with lower learning rate. While the logistic model with learning rate 0.001 achieves almost a perfect balance after 1000 iterations, the models with higher learning rates tend to bias towards negative labels and in particular achieve a poor false omission rate (FOR). For example, the ReLU model with learning rate 0.01 culminates in a FOR of 0.3727 after 400 iterations.

All in all, the experiments reflects the significance of the learning rate and how too large rates can quickly lead into one-way roads towards overfitting. We observed how the application of momentum to a learning curve with strong oscillations resembles a lengthening of the oscillations while reducing their amplitude. Momentum also effectively contributes to the improvement of the model's accuracy by a few percent, but cannot account for the weaknesses owing to the inadequately large choice of the learning rate, which therefore proves to be a fundamental hyperparameter for obtaining successful hypotheses.