# Machine Learning - Exercise 2: Classification or Regression

Apostolescu Bianca, Chen Lu, Glinzner Matthias

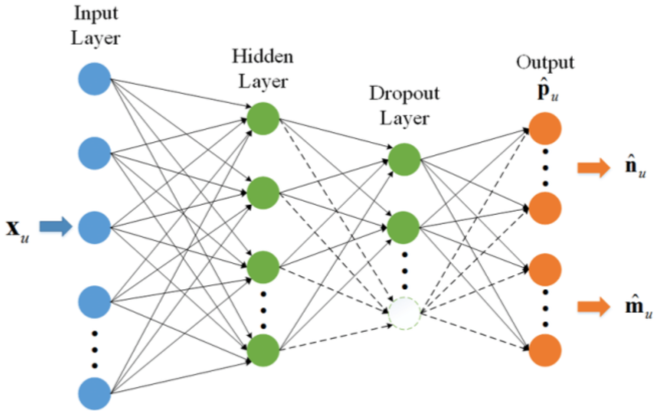
## Summary

## Introduction

- **SimpleNN**: A flexible and modular architecture that allows it to adapt to different configurations, implemented using PyTorch.
- Implementation of two different methods for the search of configurations.
- Comparison with existing methods: an NN approach (using MLP) and a tree-based method (LightGBM).

## Model Descriptions - SimpleNN

- Type of feedforward neural networks
- Modular design
- Dynamic Activation Functions
- Batch Normalization
- Dropout Regularization
- Weight Normalization



## Model Descriptions - SimpleNN

**Initialization (`__init__`):**

- The `SimpleNN` class inherits from `nn.Module`
- It accepts four key parameters:
  - `input_size`: The number of features in the input data.
  - `output_size`: The number of output neurons.
  - `layer_sizes`: A list containing the size (number of neurons) of each hidden layer.
  - `activation_funcs`: A list of PyTorch activation functions to be used in each hidden layer.

**Layers (`nn.ModuleList`):**

- `self.layers`: containing the **linear layers** of the network, with each linear layer connecting one layer's output to the next layer's input. The first layer connects the input to the first hidden layer, and each subsequent layer connects one hidden layer to the next.
- `self.batch_norms`: stores **batch normalization layers** corresponding to each hidden layer.
- `self.dropouts`: containing **dropout layers**.

## Model Descriptions - SimpleNN

**First Layer Special Handling:**

- an explicit batch normalization layer (`self.batch_norm1`) applied to the input features before the first hidden layer
- It helps in stabilizing the learning process by normalizing the distribution of the inputs.

**Output Layer:**

- an output layer (`self.output`) defined as a linear transformation after the last hidden layer
- It maps the final hidden layer's outputs to the desired output size.

**Forward Pass (`forward` method):**

- The input data is first passed through the initial batch normalization layer.
- Then, for each hidden layer, the data is processed by the linear layer, followed by batch normalization, dropout, and the specified activation function.
- Finally, the processed data is passed through the output linear layer to produce the network's output.

## Model Descriptions - MLP

**Description:**

- Supervised learning algorithm that learns a function by training on a dataset.

**Advantages:**

- Capability to learn non-linear models.
- Capability to learn models in real-time.

**Disadvantages:**

- Hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy.
- Requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
- Sensitive to feature scaling.

## Model Descriptions - LightGBM

**Description:**

- Gradient boosting framework that uses tree based learning algorithms

**Characteristics:**

- Optimal split for categorical features
- Optimal feature parallelization
- Optimal speed and memory usage

**Advantages:**

- Fast training speed
- Low memory usage
- Better performance metrics - focuses on optimizing the accuracy score
- Capable of dealing with high-scale data

## Model Descriptions - LightGBM

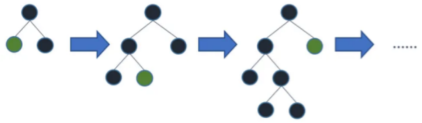


Figure: LightGBM uses leaf-wise tree growth

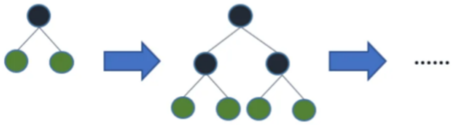


Figure: Other boosting algorithms use level-wise tree growth

- LightGBM grows trees vertically - leaf-wise
- It chooses to grow the leaf with the maximum delta loss
- It reduced the loss more abruptly than a level-wise method

## Model Comparison

| | **SimpleNN** | **MLPClassifier** | **LightGBM** |
|---|---|---|---|
| **Model Type** | Neural networks (Multi-Layer Perceptron) | Neural networks (Multi-Layer Perceptron) | A gradient boosting framework based on decision trees |
| **Learning Approach** | Learning **weight parameters** | Learning **weight parameters** | Sequentially improving predictions using an ensemble of weak decision tree models |
| **Optimization** | Utilizes **backpropagation** and **gradient-based** optimization techniques. Suitable for both CPU and GPU computation | Typically optimized for CPU use | Designed for distributed and efficient training, particularly effective for large datasets |
| **Use Case** | Complex, deep learning tasks where customization of the network architecture is required | Standard machine learning tasks for smaller to medium-sized datasets | Structured/tabular data |
| **Customization and Complexity** | High customization | Limited compared to custom implementations | |

## Hyperparameter Tuning Methods

**Grid Search Approach**

- Tune hyperparameters by **exhaustively searching** through a specified subset of hyperparameter space.

**Local Search Technique**

- **Starting Point**: It begins with a base configuration (e.g., a set of hyperparameters).
- **Neighbor Solutions**: In each iteration, the algorithm explores "neighboring" configurations. These neighbors are slight variations of the current configuration (e.g., adding or removing a layer, changing the number of neurons).
- **Iterative Improvement**: If a neighbor shows improvement over the current configuration (e.g., higher F1 score), the algorithm moves to this new configuration.
- **Termination**: This process repeats until no further improvements are found or a certain number of iterations are completed.
- **Comparison to Grid Search**: Unlike grid search, which evaluates all possible configurations in a predefined grid, local search navigates the hyperparameter space more fluidly, potentially finding good solutions with fewer evaluations. However, it may also converge to a local optimum and miss the global optimum that grid search could find.

## Experimental Setup - Dataset

**Loan Application Dataset**

- High Dimensionality : over 10k records and 92 features
- Class Imbalance
- Complex Interactions and Non-Linearity

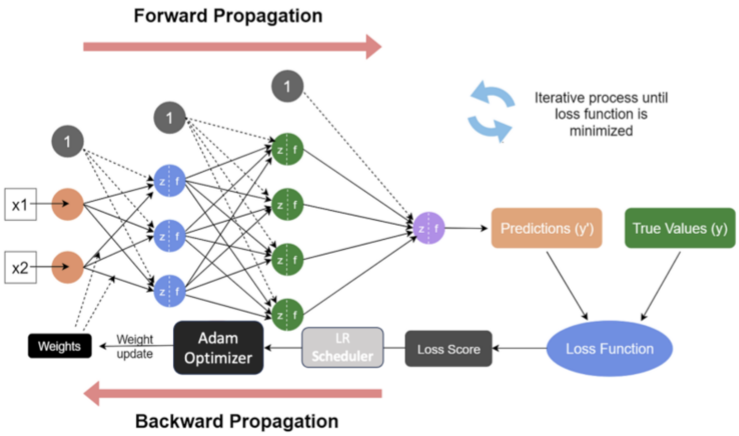
**College US News**

- High Dimensionality : 1300 records and 34 features
- Missing Values: 18
- Outliers

**Breast Cancer Dataset**

- Low Dimensionality
- No missing values
- Few Outliers

## Experimental Setup - Training Process



## Result on Loan Dataset

| | Layer | Activation | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | average |
|---|---|---|---|---|---|---|---|---|
| **0** | [64] | [ReLU] | 0.81 | 0.83 | 0.85 | 0.87 | 0.86 | 0.84 |
| **1** | [64, 64] | [ReLU, ReLU] | 0.80 | 0.80 | 0.73 | 0.69 | 0.67 | 0.74 |
| **2** | [128, 64] | [Tanh, ReLU] | 0.82 | 0.84 | 0.81 | 0.84 | 0.83 | 0.83 |
| **3** | [128, 64] | [Tanh, Tanh] | 0.82 | 0.82 | 0.82 | 0.86 | 0.86 | 0.84 |
| **4** | [64, 64, 32] | [ReLU, Tanh, ReLU] | 0.79 | 0.82 | 0.79 | 0.84 | 0.83 | 0.81 |
| **5** | [128, 128, 64] | [ReLU, ReLU, ReLU] | 0.79 | 0.82 | 0.79 | 0.84 | 0.83 | 0.81 |
| **6** | [128, 128, 64] | [Tanh, ReLU, ReLU] | 0.81 | 0.83 | 0.84 | 0.85 | 0.84 | 0.83 |

- Configuration 0 has an average F1 score of 0.84, which is relatively stable, indicating that good results can be achieved even with a simpler network.
- The average scores of configuration 4 and configuration 5 are both 0.81, indicating a deeper network structure did not bring the expected performance improvements.
- Configuration 6 has an average F1 score of 0.83, showing consistent and relatively high performance, which may indicate that **a deep network with a hybrid activation function is suitable**.

**Depth vs Width**: Compared with increasing the depth of the layers, a proper balance of depth and width (such as configuration 2 and configuration 3) may be more important.
**Impact of activation functions**: different activation functions seems to help capture different types of data features and patterns.

## Result on Loan Dataset

| | Model | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | average | Testset |
|---|---|---|---|---|---|---|---|---|
| **0** | MLP | 0.84 | 0.82 | 0.83 | 0.84 | 0.82 | 0.83 | 0.86 |
| **1** | SimpleNN | 0.84 | 0.85 | 0.84 | 0.85 | 0.85 | 0.85 | 0.85 |
| **2** | LightGBM | 0.99 | 0.98 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |

**Model performance**:

- LightGBM significantly outperforms MLP and SimpleNN on this task. (LightGBM generally performs better when processing tabular data and large-scale datasets, especially when feature relationships can be well represented by decision trees.)

**Model selection**:

- Neural network models (MLP and SimpleNN) performed relatively well in cross-validation, they failed to reach the performance level of LightGBM.

**Overfitting and generalization**:

- The performance of MLP and SimpleNN on the test set is similar to the cross-validation results, which indicates they maintain good generalization ability on unseen data.

# Result on College US News Dataset

| | Layer | Activation | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | average |
|---|---|---|---|---|---|---|---|---|
| **0** | [64] | [ReLU] | 0.80 | 0.83 | 0.89 | 0.87 | 0.85 | 0.85 |
| **1** | [64, 64] | [ReLU, ReLU] | 0.84 | 0.81 | 0.84 | 0.83 | 0.84 | 0.83 |
| **2** | [128, 64] | [Tanh, ReLU] | 0.82 | 0.80 | 0.83 | 0.81 | 0.86 | 0.82 |
| **3** | [128, 64] | [Tanh, Tanh] | 0.84 | 0.83 | 0.84 | 0.79 | 0.88 | 0.84 |
| **4** | [64, 64, 32] | [ReLU, Tanh, ReLU] | 0.75 | 0.81 | 0.88 | 0.81 | 0.85 | 0.82 |
| **5** | [128, 128, 64] | [ReLU, ReLU, ReLU] | 0.72 | 0.83 | 0.83 | 0.90 | 0.85 | 0.83 |
| **6** | [128, 128, 64] | [Tanh, ReLU, ReLU] | 0.83 | 0.81 | 0.84 | 0.84 | 0.87 | 0.84 |

- The first configuration provides the highest F1-Score - 0.84 - for the NN, which proves that even the simplest configuration of the implemented NN provides good results.
- The rest of the configurations have similar scores, indicating that the depth and complexity of the NN do not influence the results.

**Depth vs Width**: Comparing the results, the simpler configurations offer the best results for a dataset that is balanced and does not have a high dimensionality compared to the other ones.
**Impact of activation functions**: different activation functions seem to capture different patterns, although analysing the 3rd and 6th configurations, the Tanh activation function seems to be ideal.

## Result on College US News Dataset

|   | Model | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | average | Testset |
|---|---|---|---|---|---|---|---|---|
| **0** | MLP | 0.60 | 0.65 | 0.58 | 0.73 | 0.64 | 0.64 | 0.69 |
| **1** | SimpleNN | 0.76 | 0.77 | 0.77 | 0.77 | 0.76 | 0.77 | 0.71 |
| **2** | LightGBM | 0.71 | 0.73 | 0.72 | 0.76 | 0.73 | 0.73 | 0.73 |

**Model performance**:

- The performance of all the models is similar, the LightGBM model slightly outperforming the MLP and the implemented Simple NN model.
- LightGBM is known to have better performance when dealing with categorical features and non-linear data.

**Model selection**:

- Over the cross-validation folds, the F1-Scores are similar between models, however, on unseen data, LightGBM outperformed the other two.

**Overfitting and generalization**:

- While the performance of MLP for the test set remains similar to the cross-validation results, for the Simple NN, the results for the test set decrease due to its inability to generalize pattern on unseen data.

## Result on Breast Cancer Dataset

| | Layer | Activation | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | average |
|---|---|---|---|---|---|---|---|---|
| **0** | [64] | [ReLU] | 0.95 | 0.98 | 1.00 | 1.00 | 1.00 | 0.99 |
| **1** | [64, 64] | [ReLU, ReLU] | 0.98 | 0.96 | 1.00 | 1.00 | 1.00 | 0.99 |
| **2** | [128, 64] | [Tanh, ReLU] | 0.96 | 0.93 | 0.98 | 1.00 | 1.00 | 0.98 |
| **3** | [128, 64] | [Tanh, Tanh] | 0.98 | 1.00 | 1.00 | 0.98 | 1.00 | 0.99 |
| **4** | [64, 64, 32] | [ReLU, Tanh, ReLU] | 0.98 | 0.98 | 1.00 | 1.00 | 0.96 | 0.99 |
| **5** | [128, 128, 64] | [ReLU, ReLU, ReLU] | 0.98 | 0.96 | 1.00 | 1.00 | 0.98 | 0.99 |
| **6** | [128, 128, 64] | [Tanh, ReLU, ReLU] | 0.98 | 1.00 | 1.00 | 0.98 | 0.98 | 0.99 |

- The configuration with index 3 performs slightly better than the rest.
- Overall the results show that for this dataset neural networks in general achieve high scores.

**Depth vs Width**: All scores are high, there is very little difference.
**Impact of activation functions**: Again, the differences are negligible; Tanh seems to perform slightly better.

## Result on Breast Cancer Dataset

| | Model | Fold1 | Fold2 | Fold3 | Fold4 | Fold5 | average | Testset |
|---|---|---|---|---|---|---|---|---|
| **0** | MLP | 0.89 | 0.88 | 0.88 | 0.83 | 0.96 | 0.89 | 0.88 |
| **1** | SimpleNN | 0.98 | 0.98 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| **2** | LightGBM | 1.00 | 0.88 | 0.93 | 0.91 | 0.95 | 0.94 | 0.95 |

**Model performance**:

- Interestingly, SimpleNN outperforms its competition with LightGBM a close second.
- MLP performs worst.

**Model selection**:

- SimpleNN performs well on this dataset.

**Overfitting and generalization**:

- Due to the structure of the dataset, neural networks in general perform well.

## Conclusion

- Looked at three models: **SimpleNN**, **MLPClassifier** and **LightGBM**
- Looked at three datasets: **Loans**, **College US News** and **Breast Cancer**
- Compared results:
  - LightGBM outperforms other models except for the Breast Cancer dataset
  - Tanh seems to perform slightly better