

**WiDS ‘22 - ‘23 Final Documentation**

**22 - Diving Into ML**

**Soham Joshi and Anand Narasimhan**

|  |  |  |
| --- | --- | --- |
| **Team Member Name** | **Roll Number** | **Email-Id** |
| Dnyaneshwar Khaire | 22M0206 | [22m0206@iitb.ac.in](mailto:22m0206@iitb.ac.in) |

**Introduction to Problem Statement**

|  |
| --- |
| The project's purpose is to establish a solid foundation in ML (its theory and basic applications) on which to build in the future. The project is divided into four weeks: week 0, week 1, week 2, and week 3. Each week included a new problem set.  Week0:  a) To calculate the time difference between iterative multiplication and matrix multiplication |

**Existing Resources**

|  |
| --- |
| **WEEK 0 -** <https://www.w3schools.com/python/>  <https://stackoverflow.com/questions/49372918/group-numpy-into-multiple-sub-arrays-using-an-array-of-values>  <https://stackoverflow.com/questions/49141969/vectorized-groupby-with-numpy?noredirect=1&lq=1>  **WEEK 1 -** Forward and Backward Propagation (C1W4L06)  <https://explained.ai/matrix-calculus/>  **WEEK 2 –**  <https://www.cs.cornell.edu/courses/cs5740/2017sp/lectures/04-nn-compgraph.pdf> <https://github.com/Ihsoj-Mahos/WiDS-Week2/tree/master/resources>  <https://openlearninglibrary.mit.edu/assets/courseware/v1/2481f8f2964716032b134db99e369b81/asset-v1:MITx+6.036+1T2019+type@asset+block/notes_chapter_Introduction.pdf>  **WEEK 3 -** <https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi> <https://github.com/A9B8C7D6/WiDS-week3/tree/main/resources> |

**Proposed Solution**

|  |
| --- |
| Learning from available resources and understanding fundamental concepts try to solve each week problem. |

**Methodology & Progress (Mention the work done week-wise)**

|  |
| --- |
| **Week wise methodology and progress**  **Week0:**  a)   1. Measure the time it takes to perform a certain number of multiplications for iterative and matrix methods. 2. Compare the times obtained in above step and calculate the difference.   b)   1. Pre-processing - Convert the images to a standardized format and normalize the pixel values 2. Feature extraction - Extract important features from the images to be used as input to the model 3. Model training - Train a suitable machine learning model on the preprocessed data 4. Model evaluation - Evaluate the performance of the model on a separate test dataset 5. Model improvement - Fine-tune the model or try different models to improve its accuracy.   **Week1:**  **Problem 1)**   1. Analytic computation:    * If the function is defined in a symbolic form, use a library such as SymPy to symbolically differentiate the function with respect to each independent variable    * Calculate the gradient vector by combining the symbolic partial derivatives 2. Numeric computation:    * Create a loop to iterate over each independent variable    * Within the loop, perturb the variable by the step size, evaluate the function, and calculate the change in the function value    * Update the gradient approximation by dividing the change in the function value by the step size    * Repeat the process for each independent variable to obtain the gradient vector   **Problem 2)**   1. Forward Propagation:    * Pass the input data through the network, using matrix operations to compute the activations at each layer    * Apply activation functions at each layer to obtain the output of each layer    * Repeat the process until the final output layer is reached 2. Backward Propagation:    * Compute the error or loss between the predicted output and the true output    * Propagate the error backwards through the network, starting from the final output layer    * Compute the gradients of the loss with respect to the parameters of each layer    * Update the parameters to minimize the loss, using gradient-based optimization algorithms such as stochastic gradient descent   **Week2:**  Problem: Plotting the decision boundary   1. Prepare your data: Organize your data into input features and target labels, split into training and testing sets. 2. Train a classifier: Train a binary classifier such as a linear model, SVM, or decision tree on the training data. 3. Plot the decision boundary: To plot the decision boundary, you can use a mesh grid to generate a set of points that cover the space of the input features. You can then evaluate the classifier on these points to obtain a prediction for each point, which can be used to color the points based on their predicted label. Finally, you can plot the resulting-colored points on a graph to visualize the decision boundary. 4. Evaluate the classifier: Evaluate the performance of the classifier on the testing data to determine how well it generalizes to new data.   **Week3**   1. Load the MNIST data: Use a library such as TensorFlow or PyTorch to load the MNIST data set into memory. 2. Preprocess the data: Normalize the pixel values to be between 0 and 1, and one-hot encode the target labels. 3. Define the neural network architecture: Choose the number of hidden layers, the number of neurons in each layer, and the activation functions to use. 4. Compile the model: Define the loss function, optimizer, and metrics to use for training. 5. Train the model: Fit the model to the training data by specifying the number of epochs and batch size. Monitor the training loss and accuracy to ensure the model is converging. 6. Evaluate the model: Evaluate the model on the testing data to determine its generalization performance. 7. Fine-tune the model: Based on the evaluation results, adjust the architecture, optimizer settings, or other parameters to improve performance. 8. Use the model: Use the trained model to make predictions on new data. |

**Results**

|  |
| --- |
| https://github.com/dnyaneshwar9595/DK-Diving-into-ML.git |

**Learning Value**

|  |
| --- |
| **Got to know the fundamental concepts of ML as follows:**  ● Python (Numpy, Matplotlib, pandas, etc.) ● GitHub and google collaboratory ● Forward and backward propagation ● Computational graph ● Linear regression and binary classification ● Introductions Neural networks |

**Tech-stack Used**

|  |
| --- |
| Python (TensorFlow, PyTorch, scikit-learn) |

**Suggestions for others**

|  |
| --- |
| A really interesting and learning-oriented journey that provides hands-on Python programming as well as a basic introduction to Machine Learning and Neural Network ideas. The resources supplied were both instructive and practical. |

**Contribution by each Team Member**

|  |
| --- |
|  |

**References and Citations**

|  |
| --- |
| *1. https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.Perc eptron.html 2. https://www.w3schools.com/python/ 3. https://towardsdatascience.com/stochastic-gradient-descent-clearly-explain ed-53d239905d31 4. https://docs.python.org/3/library/timeit.html* |