

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

**22 - Diving Into ML**

**Soham Joshi and Anand Narasimhan**

|  |  |  |
| --- | --- | --- |
| **Team Member Name** | **Roll Number** | **Email-Id** |
| Ramswaroop | 203050117 | [203050117@iitb.ac.in](mailto:203050117@iitb.ac.in) |

**Introduction to Problem Statement**

|  |
| --- |
| The project's primary goal is to provide a comprehensive understanding of machine learning. The curriculum is designed to cover both the theoretical foundations and practical applications of ML. To achieve this, the project is divided into four weeks, each week building upon the knowledge gained from the previous one. The weeks are labeled week 0, week 1, week 2, and week 3, and each week will consist of a set of problems to be solved, gradually increasing in complexity and difficulty. By the end of the four weeks, participants will have a solid understanding of ML and its various components.  Week 0: focuses on comparing the time taken for iterative and matrix methods for performing multiplications and evaluating the pre-processing, feature extraction, model training, model evaluation, and improvement of an image classification model.  Week 1: involves computing gradients using analytic and numeric methods and training a neural network using forward and backward propagation.  Week 2: involves plotting the decision boundary of a binary classifier and evaluating its performance.  Week 3: involves loading, pre-processing, defining, compiling, training, evaluating, and fine-tuning a neural network model on the MNIST data set, and making predictions with the trained model. |

**Existing Resources**

|  |
| --- |
| **WEEK 0 -** <https://inside-machinelearning.com/en/quickly-upload-public-google-drive-files-on-notebook-and-colab/>  <https://www.analyticsvidhya.com/blog/2021/06/complete-guide-to-working-with-csv-files-in-python-with-pandas/>  <https://stackoverflow.com/questions/39662891/read-in-the-first-column-of-a-csv-in-python>  <https://colah.github.io/posts/2014-10-Visualizing-MNIST/>  <https://www.kaggle.com/code/yushg123/for-loops-vs-vectorized-who-wins-and-by-how-much/notebook>  **WEEK 1 -**  <https://explained.ai/matrix-calculus/>  [Calculus on Computational Graphs: Backpropagation -- colah's blog](https://colah.github.io/posts/2015-08-Backprop/)  [c++ - Is there any "standard" way to calculate the numerical gradient? - Stack Overflow](https://stackoverflow.com/questions/38854363/is-there-any-standard-way-to-calculate-the-numerical-gradient)  [Coding Neural Network — Forward Propagation and Backpropagtion | by Imad Dabbura | Towards Data Science](https://towardsdatascience.com/coding-neural-network-forward-propagation-and-backpropagtion-ccf8cf369f76)  [Of Gradients and Matrices. This artcle explains a line of code in… | by Paolo Perrotta | Medium](https://nusco.medium.com/of-gradients-and-matrices-1b19de65e5cd)  **WEEK 2 –**  [5.pdf (stanford.edu)](https://web.stanford.edu/~jurafsky/slp3/5.pdf)  [Logistic Regression as a Nonlinear Classifier – Data Exploration (xplordat.com)](https://xplordat.com/2019/03/13/logistic-regression-as-a-nonlinear-classifier/#:~:text=Logistic%20regression%20is%20known%20and%20used%20as%20a,to%20use%20logistic%20regression%20as%20a%20linear%20classifier.)  [numpy.random.Generator.normal — NumPy v1.25.dev0 Manual](https://numpy.org/devdocs/reference/random/generated/numpy.random.Generator.normal.html)  [Activation function - Wikipedia](https://en.wikipedia.org/wiki/Activation_function)  [Binary Classification – LearnDataSci](https://www.learndatasci.com/glossary/binary-classification/)  [matplotlib - plotting decision boundary of logistic regression - Stack Overflow](https://stackoverflow.com/questions/28256058/plotting-decision-boundary-of-logistic-regression)  [Logistic Regression and Decision Boundary | by Anuradha Wickramarachchi | Towards Data Science](https://towardsdatascience.com/logistic-regression-and-decision-boundary-eab6e00c1e8)  [Decision Boundary Visualization of Trained Logistic Regression (favtutor.com)](https://favtutor.com/blogs/decision-boundary-logistic-regression)  [F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose? - neptune.ai](https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc) **WEEK 3 -** <https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi>  [Fundamentals of Neural Networks on Weights & Biases (wandb.ai)](https://wandb.ai/site/articles/fundamentals-of-neural-networks?https://wandb.ai/site/artifacts?utm_source=bing&utm_medium=cpc&utm_campaign=Conversions%3A+Marketing+Site+-+Non-Branded+-+Dynamic+Search&utm_content=Baseline+-+1st+Ad+India&msclkid=df0b01ce732e1f0975686214ba5bf2fe&utm_term=https%3A%2F%2Fwandb.ai%2Fsite)  [Linear Regression Model Neural Networks – Surfactants](https://www.surfactants.net/linear-regression-model-neural-networks/#:~:text=A%20linear%20regression%20model%20neural%20network%20is%20a,dependent%20variable%20is%20represented%20by%20a%20scalar%20value.)  [Choosing activation functions - Neural network training | Coursera](https://www.coursera.org/lecture/advanced-learning-algorithms/choosing-activation-functions-aWivF)  [PyTorch Activation Functions - ReLU, Leaky ReLU, Sigmoid, Tanh and Softmax - MLK - Machine Learning Knowledge](https://machinelearningknowledge.ai/pytorch-activation-functions-relu-leaky-relu-sigmoid-tanh-and-softmax/)  [Loss Functions in Neural Networks (theaidream.com)](https://www.theaidream.com/post/loss-functions-in-neural-networks)  [How to Train a Model with MNIST dataset | by Abdullah Furkan Özbek | Medium](https://medium.com/@afozbek_/how-to-train-a-model-with-mnist-dataset-d79f8123ba84)  [MNIST dataset using Deep Learning algorithm (ANN) | by Prateek Goyal | Medium](https://medium.com/@prtk13061992/mnist-dataset-using-deep-learning-algorithm-ann-c6f83aa594f5) |

**Proposed Solution**

|  |
| --- |
| The proposed solution would involve completing the tasks outlined in each of the four weeks, building up a comprehensive understanding of machine learning and its components through available resources and understanding both theoretical foundations and practical applications. |

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

|  |
| --- |
| **Week wise methodology and progress**  **Week0:**  a) To compare the performance of iterative and matrix methods for performing multiplications, the following steps need to be taken:   1. Measure the time it takes to perform a specific number of multiplications for each method. 2. Calculate the difference between the times obtained in step 1 to determine which method is faster.   b) For second problem about MNIST dataset (a large dataset of hand-written images of numbers), the following steps followed: like loading data, organizing data, plotting images, group images, computing mean images in details following steps followed:   1. Preprocessing - In this step, the images are standardized by converting them to a specific format and normalizing the pixel values. 2. Feature extraction - The next step is to extract important features from the images to be used as inputs for the model. 3. Model training - A suitable machine learning model is trained on the preprocessed data. 4. Model evaluation - To assess the performance of the model, it is evaluated on a separate test dataset. 5. Model improvement - Finally, the model is fine-tuned or alternative models are tried to improve its accuracy. It is important to continuously evaluate and improve the model to achieve the desired performance.   **Week1: Linear Algebra and Calculus**  Problem 1: Analytic and Numeric Computation of Gradients for Scalar valued functions.   1. Analytic Computation:    1. Use a symbolic library, such as SciPy, to symbolically differentiate the function with respect to each independent variable like normal variable, uniform variable and binomial variables etc    2. Combine the partial derivatives to calculate the gradient vector 2. Numeric Computation:    1. Iterate over each independent variable using a loop    2. In the loop, disrupt the variable with a pitch size, evaluate the function by calculating the change in the function value.    3. Divide the change in the function value by the step size to update the gradient approximation    4. Repeat the process for each independent variable to obtain the gradient vector   Problem 2: computational graphs, and their calculus, finding gradients of variables with respect to other variables in the graph and Forward and Backward Propagation in Neural Networks.   1. Multi-input nodes:    1. The Add class computes the output as the sum of all the variables in lst\_values and computes the gradients as a dictionary with all the variable names in lst\_names and gradient values of 1.    2. The Multiply class computes the output as the product of all the variables in lst\_values and computes the gradients as a dictionary with variable names and gradient values as the product of all lst\_values except the variable at the current iteration index. 2. Scalar multiplication/addition nodes:    1. The "Add\_Scalar" class adds a variable with a scalar and the "Multiply\_Scalar" class multiplies a variable with a scalar. Both classes have the same structure:    2. They take two inputs, a list of variable names and a list of values, in their constructor.    3. They have two methods, "compute\_output" and "compute\_gradients".    4. "compute\_output" method calculates and returns the result of the operation.    5. "compute\_gradients" method returns the gradient of the output with respect to each input. 3. Nodes for special functions:    1. This defines several mathematical operations (Add\_Scalar, Multiply\_Scalar, Power, Sine, Logarithm, and Exponential) as Python classes. Each class has two methods: compute\_output and compute\_gradients.    2. The compute\_output method computes the result of the operation applied to its input variables, while the compute\_gradients method computes the partial derivative of the result with respect to each input variable. The input variables are stored as class attributes, along with any additional parameters required for each operation. 4. Forward Propagation:    1. Pass input data through the network using matrix operations to compute activations at each layer    2. Apply activation functions to obtain the output of each layer    3. Repeat the process until reaching the final output layer 5. Backward Propagation:    1. Compute the error or loss between the predicted and true output    2. Propagate the error backwards from the final output layer through the network    3. Compute the gradients of the loss with respect to the parameters of each layer    4. Use gradient-based optimization algorithms, such as stochastic gradient descent, to update the parameters and minimize the loss.   **Week2: Linear Classifiers and Logistic Regression**   1. Data set generation: 2. This generates and plots random data points with two categories (positive and negative) that have been rotated and added a constant value. The data points are generated using a random number generator with a fixed seed and normal distribution. 3. The positive and negative categories are created by labelling data points based on a binomial distribution with a probability of 0.95. The data points are then combined and shuffled randomly. The data points are then rotated using a rotation matrix and added a constant value before being plotted. The plotted data points are shown as scatter plots with different colours representing the two categories. 4. Binary classifier: gradient descent 5. The aim is to estimate the parameters W and b by minimizing the loss function L(a, y) using gradient descent. The input to the function is the feature data X, label data y, the number of iterations (default is 1000), and the learning rate (default is 0.01). 6. The function should return the estimated parameters W and b. There is an optional bonus to plot the loss function as a function of the number of iterations. The code has been provided with a placeholder to insert the gradient descent algorithm implementation. 7. Plotting the decision boundary:    * 1. Initialize the parameters W and b with random values.   Implement the logistic regression function that takes the input X and calculates the output a using the equation z=Wx+b and a=σ(z).  Calculate the cost function L(a, y) using the equation L(a, y)=−(y.log(a)+(1−y).log(1−a)).  Implement the Gradient Descent algorithm to update the parameters W and b by minimizing the cost function L.   * + 1. Implement a function to plot the decision boundary by evaluating the logistic regression function for a range of values on the x and y axis and plotting the resulting points that have a value close to 0.5. Plot the decision boundary along with the dataset, with the appropriate colouring for the two classes.  1. Accuracy: 2. calculates the accuracy of a classifier by dividing the number of correctly classified points by the total number of points in the dataset. The predict function takes as inputs a dataset of points represented by the X numpy array, the true labels of the points given by y, and the model parameters represented by W and b. 3. The function returns an array of predicted labels, preds, for the dataset using the classifier model. The assert statement at the end of the function checks that the dimensions of the true labels and the predicted labels are the same. The accuracy is then calculated by multiplying the number of correctly classified points by 100 and dividing by the total number of points.   **Week3 : Neural Networks**   1. Linear module: 2. The Linear class takes in the size of the input layer, m, and the size of the output layer, n, as arguments to initialize the layer. It then initializes the model parameters, including the biases W0 and weights W. The forward method performs a linear transformation of the input A to produce the output of the layer. 3. The backward method calculates the gradients of the loss with respect to the parameters and the inputs. The sgd\_step method updates the parameters using the gradient descent optimization algorithm. 4. Activation functions: 5. Tanh function, the forward method takes in a batch of pre-activations Z and returns a batch of activations A by computing the hyperbolic tangent of Z. The backward method takes in dLdA and returns dLdZ. 6. ReLU function, the forward method takes in a batch of pre-activations Z and returns a batch of activations A by computing the maximum between Z and 0. The backward method takes in dLdA and returns dLdZ. 7. SoftMax function, the forward method takes in a batch of pre-activations Z and returns a batch of activations Ypred by computing the softmax activation. The backward method takes in dLdZ and returns it as is. The class\_fun method takes in Ypred and returns the class indices (integers) with the highest probability for each point. 8. Loss function 9. negative log likelihood (NLL) loss function. The forward method takes in two inputs: Ypred and Y, which are the predicted and true labels respectively. It stores Ypred and Y as attributes and returns the NLL loss. 10. The backward method computes the derivative of the loss with respect to Ypred using self.Ypred and self.Y and returns it as dLdYpred. 11. Neural network 12. It defines a Sequential class for a neural network. It has a constructor that takes in a list of modules and a loss module, which are stored as attributes.The sgd method performs Stochastic Gradient Descent (SGD) for a specified number of iterations with a given learning rate. The forward method computes the predicted labels Ypred for a given input Xt by passing Xt through each of the modules in the list. 13. The backward method updates the gradients dLdW and dLdW0 by computing the derivative of the loss with respect to the output of each module. The sgd\_step method performs a gradient descent step by updating the parameters of each module. The print\_accuracy method prints the accuracy and loss on the full dataset every specified number of iterations. 14. Training MNIST dataset 15. It trains a neural network on the MNIST dataset. It uses the "Sequential" class which is a list of the different layers of the network (Linear, ReLU, SoftMax) and the loss function (NLL). The training data is loaded and used to train the network using stochastic gradient descent (SGD) with 100 iterations and a learning rate of 0.005. 16. The testing data is loaded and the accuracy of the network is determined by comparing the predicted labels from the network to the actual labels and computing the percentage of correct predictions. |

**Results**

|  |
| --- |
| [swaroopram117/WiDS\_Diving-into-ML (github.com)](https://github.com/swaroopram117/WiDS_Diving-into-ML) |

**Learning Value**

|  |
| --- |
| Through this journey, the basics of Machine Learning were acquired, including a solid understanding of Python programming tools such as Numpy, Matplotlib, SciPy, and pandas. The experience also involved hands-on work with GitHub and Google Colaboratory, and a comprehensive understanding of concepts such as linear algebra and calculas for machine learning, gradient descent, classifier and regression forward and backward propagation, computational graphs, and neural networks. |

**Tech-stack Used**

|  |
| --- |
| Python (NumPy, pandas, matplotlib, TensorFlow, PyTorch, scikit-learn), colab, vs code, jupyter lab |

**Suggestions for others**

|  |
| --- |
| This is a valuable and educational experience that offers practical Python programming and a basic understanding of Machine Learning and Neural Network concepts. The resources provided are not only informative but also hands-on and applied. Additionally, it gives a great opportunity to gain hands-on experience with these cutting-edge technologies and learning. |

**Contribution by each Team Member**

|  |
| --- |
|  |

**References and Citations**

|  |
| --- |
| [*https://inside-machinelearning.com/en/quickly-upload-public-google-drive-files-on-notebook-and-colab/*](https://inside-machinelearning.com/en/quickly-upload-public-google-drive-files-on-notebook-and-colab/)  [*https://www.analyticsvidhya.com/blog/2021/06/complete-guide-to-working-with-csv-files-in-python-with-pandas/*](https://www.analyticsvidhya.com/blog/2021/06/complete-guide-to-working-with-csv-files-in-python-with-pandas/)  [*https://colah.github.io/posts/2014-10-Visualizing-MNIST/*](https://colah.github.io/posts/2014-10-Visualizing-MNIST/)  [*https://www.kaggle.com/code/yushg123/for-loops-vs-vectorized-who-wins-and-by-how-much/notebook*](https://www.kaggle.com/code/yushg123/for-loops-vs-vectorized-who-wins-and-by-how-much/notebook)  [*https://explained.ai/matrix-calculus/*](https://explained.ai/matrix-calculus/)  [*Calculus on Computational Graphs: Backpropagation -- colah's blog*](https://colah.github.io/posts/2015-08-Backprop/)  [*Coding Neural Network — Forward Propagation and Backpropagtion | by Imad Dabbura | Towards Data Science*](https://towardsdatascience.com/coding-neural-network-forward-propagation-and-backpropagtion-ccf8cf369f76)  [*5.pdf (stanford.edu)*](https://web.stanford.edu/~jurafsky/slp3/5.pdf)  [*Logistic Regression as a Nonlinear Classifier – Data Exploration (xplordat.com)*](https://xplordat.com/2019/03/13/logistic-regression-as-a-nonlinear-classifier/#:~:text=Logistic%20regression%20is%20known%20and%20used%20as%20a,to%20use%20logistic%20regression%20as%20a%20linear%20classifier.)  [*Logistic Regression and Decision Boundary | by Anuradha Wickramarachchi | Towards Data Science*](https://towardsdatascience.com/logistic-regression-and-decision-boundary-eab6e00c1e8)[*https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1\_67000Dx\_ZCJB-3pi*](https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi)  [*Fundamentals of Neural Networks on Weights & Biases (wandb.ai)*](https://wandb.ai/site/articles/fundamentals-of-neural-networks?https://wandb.ai/site/artifacts?utm_source=bing&utm_medium=cpc&utm_campaign=Conversions%3A+Marketing+Site+-+Non-Branded+-+Dynamic+Search&utm_content=Baseline+-+1st+Ad+India&msclkid=df0b01ce732e1f0975686214ba5bf2fe&utm_term=https%3A%2F%2Fwandb.ai%2Fsite)  [*PyTorch Activation Functions - ReLU, Leaky ReLU, Sigmoid, Tanh and Softmax - MLK - Machine Learning Knowledge*](https://machinelearningknowledge.ai/pytorch-activation-functions-relu-leaky-relu-sigmoid-tanh-and-softmax/)  [*How to Train a Model with MNIST dataset | by Abdullah Furkan Özbek | Medium*](https://medium.com/@afozbek_/how-to-train-a-model-with-mnist-dataset-d79f8123ba84) |