# Diving into ML

Sathwika Reddy

27th January 2023

## 1 Week-0

We started with basic python programs to explore the concept of vectorising processes. We multiplied vectors by following methods

- iteratively
- matrix multiplication

and compared, plotted the time taken in 2 methods using matplotlib library and tried to reduce the number of for loops used for computing multiplication using numpy library
The following observations were made from the graph plotted

1. Order of time for iterative method is in seconds and that of matrix method is in milliseconds
2. Order of iterative method is O(mn) and that if iterative method is O(m)

Then we went into loading the MNIST dataset from CSV file where each row has a long list of 784 numbers(pixel values from 0 to 255) and its label( from 0 to 9) and found ways to store the data into a tensor of rank 3 using reshape function in numpy from the file and then plotted the images of data using matplotlib. And then tried to group the images based on their labels using minimum number of for loops using numpy library. I have computed the mean of all images of same label and plotted them.
The assignment is basically focused on acquainting us with the numpy, pandas, matplotlib libraries.

## 2 Week-1

The 2nd assignment is basically focused on concepts of analytic and numeric computation of gradients. We looked into concepts of computation graphs, forward and backward propagations.

## 2.1 Analytic Gradient

It is the normal way of calculating the gradient of a function using backpropagation. We calculated the gradient of loss function wrt to the input vector x. To compute the gradient of a scalar function wrt vector, we basically need to compute the gradient of function wrt every component of the vector.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

## 2.2 Numeric gradient

The numeric gradient of the scalar function wrt vector is calculated by changing a single component of the vector keeping others constant and finding the change in the value of the function

$$\frac{\partial f}{\partial x_i} = \frac{f(x_i(1+\sqrt{\epsilon})) - f(x_i(1-\sqrt{\epsilon}))}{2x_i\sqrt{\epsilon}}$$

## 2.3 Computational Graphs

We then went into computational graphs which is one of the important concepts in neural networks. They are a form of directed graphs that represent a mathematical expression and are used for computing the gradients of variables wrt the other variables in the graph. Every node in the graph can contain either an operation, variable, or equation itself.
We then defined classes for addition and multiplication operations for variables and the gradient of the result wrt each variable. Then the addition and multiplication with a constant and the gradient of the result and then went on to define classes for power, sine, logarithm, and exponential operations and their gradients wrt variable involved.

### 2.3.1 Forward Propagation

Forward propagation is where input data is fed through a network, in a forward direction to generate the output. We used the classes for operations defined above to calculate the output.

#### 2.3.2 Backward Propagation

Backpropagation is an algorithm that backpropagates the errors from the output nodes to the input nodes. It is often referred to as the backward propagation of errors. We used backpropagation for calculating the gradient of loss function wrt to all parameters so as to minimize it.
At the end of the assignment, we tried to test the functions on an example

## 3 Week-2

This week, we basically focussed on one of the most important algorithms in Machine Learning, i.e Gradient descent and Ml models like logistic regression and linear classifiers.

### 3.1 Supervised Learning

The basic idea of supervised learning is that we are given the inputs and the outputs associated with them. We are basically required to make use of the datasets to train algorithms to classify data or predict the outcomes accurately. Based on this we have 2 types of supervised learning,

- Classification
- Regression

#### 3.1.1 Classification

In this type of supervised learning, the target values are a discrete set of values and if a new input is given, we are required to predict the target value and classify it. The classification problem is binary or two-class problem if target value is drawn from the set of 2 possible values, otherwise a multi-class problem.

#### 3.1.2 Regression

In this type of supervised learning, the target values are a continuous set of values. The goal is to fit find the best-fit line or curve for the data.

### 3.2 Unsupervised Learning

It doesn't involve learning a function from inputs to outputs based on set of input-output pairs. Instead, we are given a dataset and are expected to find some patterns or structures inherent in it. Some examples of unsupervised learning are

- Clustering
- Dimension Reduction
- Density Estimation

### 3.3 Linear classifiers

Linear classifiers achieve the classification goal by making the decision based on the value of the combination of the characteristics( also called feature values) and are generally fed to the machine in the vector called feature vector.
If the input feature vector is x, then the output is

$$y = f(w.x) = f(\Sigma_{i=1}^{n} w_i.x_i)$$

where w is the real vector of weights and it is learned from the set of training samples. The function f is called the threshold function, which classifies the data based on the function value

$$f(x) = \begin{cases} 1, & \text{if } w.x > b \\ 0, & \text{otherwise} \end{cases}$$

b is the scalar threshold. For the two-class or binary classification problem, we can visualize the operation of a linear classifier as splitting the high-dimensional input space with a hyperplane( all points on one side of the hyperplane are classified as "yes", while the others are classified as "no")
In the assignment, we are required to find a hyperplane that separates the labeled dataset of red and blue points in $R^2$. To get the best values of w and b to fit the data, we need to use gradient descent on the loss function.

### 3.4 Gradient Descent

This algorithm is used for finding a local minimum of a differential function. The idea is to take repeated steps in the opposite direction of the gradient of the function at the current point. This will lead to the local minimum of the function.
Algorithm for the one-dimensional function,

**Algorithm 1** Gradient Descent Algorithm

```
1: procedure 1D-GRADIENT-DESCENT($\theta$,$\eta$, $f$)
2:     $\theta^0$=$\theta_{init}$
3:     t=0
4:     repeat
5:         t=t+1
6:         $\theta^t$=$\theta^{t-1}$-$\eta$.f$^1$($\theta^{t-1}$)
7:     until |$f(\theta^t)$ - f($\theta^{t-1}$) | < $\epsilon$
8: return $\theta^t$
```

So in the assignment problem, we run the gradient descent algorithm on loss function wrt both w and b so as to find the best-fit line. And then plotted the line using matplotlib library.
Then I found the accuracy of the predicted w and b values using correctly classified points and the total number of points.

Correctly classified points are the ones with the positive sign of w.x+b and y=1 and the ones with the negative sign of w.x+b and y=0. And then finally plotted the graph of the accuracy of the model versus the number of iterations in the gradient descent algorithm. As the number of iterations increases, the accuracy of the model increases.

## 3.5 Week-3

This week we focused on building neural networks with multiple layers. We have broken down each layer into 2 parts( linear module, activation module). Linear module implements linear transformation of output from previous layer. Activation module applies an activation function(ReLu or Tanh(which converts the input into range(0,1) and Softmax function( used in multi-classification model) at the output layer) on output of the linear module. These activations are given as input for the next layer.
Then we implemented the forward, backward and sgdstep in linear module. Forward function is for computing output from the input by linear transformation and backward function for computing the values of derivative of loss function wrt A, W, $W_0$ given its derivative wrt Z.

$$\frac{\partial L}{\partial W_0} = \frac{\partial L}{\partial Z}$$
$$\frac{\partial L}{\partial A} = W.\frac{\partial L}{\partial Z}$$
$$\frac{\partial L}{\partial W} = A.(\frac{\partial L}{\partial Z})^T$$

Sgdstep( Gradient Descent Step) is for estimating the values of W and $W_0$ at which the loss function is minimized
In the activation layer, we implemented the forward(computing output from input by applying the function) and backward( given the derivative wrt A, we need to compute the derivative wrt Z) member functions in Tanh, ReLU, and SoftMax modules.
Then we implemented loss functions which takes in a batch of predictions Ypred and actual labels Y. Loss function is,

$$loss = -\Sigma_{i=1}^{n} Y_i.\log(Ypred_i)$$

and backward fucntion in NLL module computes the derivative of the loss fucntion wrt preactivation to SoftMax.

$$\frac{\partial L}{\partial Z} = Ypred - Y$$

And then went into the final implementation of the neural network using class Sequential. To train the data, sgd member function is defined which randomly picks the data from provided data and calls the forward function( which calculates the final output by using for loop, output in one layer is the input for the next layer) and then calls the loss module for computing the loss for Ypred relative to Y and then calls the backward function( which computes the gradient of the loss function wrt to the A, W, $W_0$ at every layer, we use the values of gradients calculated at next layer to compute the gradients at this layer, therefore we calculate the gradients in the order from outermost layer to innermost) and then finally sgdstep which updates the values of W, $W_0$ so as to reduce the loss function. At the end of the training of the data, we finally get the values for W and $W_0$ that minimizes the loss function and thereby increasing the accuracy of our model. As the size of our training set increases, the accuracy of our mode increases.