Tugas Besar B

Implementasi Mini-Batch Gradient Descent

IF3270 Pembelajaran Mesin

Dibuat oleh:

Daffa Pratama Putra	13518033
Muchammad Ibnu Sidqi	13518072
Muhammad Firas	13518117
Muhammad Daffa Dinaya	13518141

Teknik Informatika

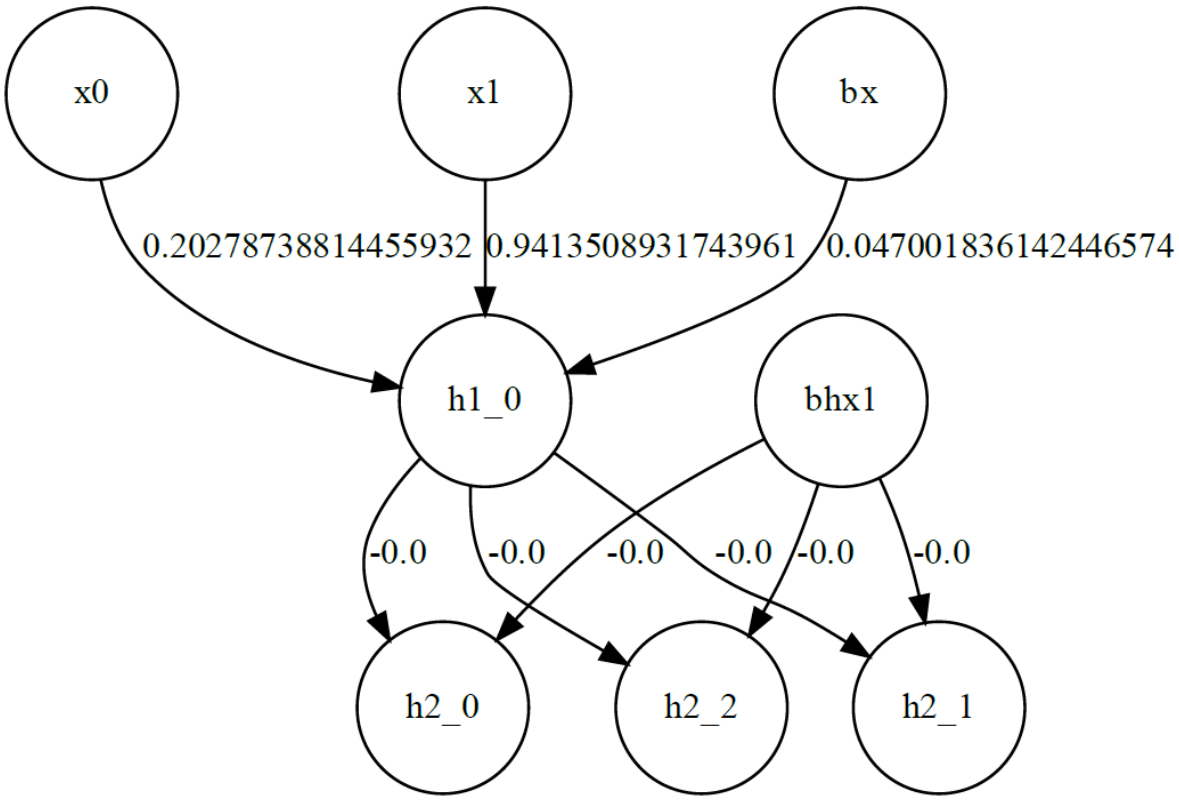
Institut Teknologi Bandung

2021

1. Penjelasan Implementasi

Pada tugas ini, kami membuat implementasi Backpropagation Learning from scratch menggunakan Jupyter Notebook. Program yang kami buat, akan menerima input berupa file eksternal csv yang berisikan data uji berupa tabel dan jumlah neuron dan fungsi aktivasi untuk setiap model. Data uji diambil dari iris dataset pada sumber berikut (https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/0e7a9b0a5d22642a06d3d5b9bcbad9890c8ee534/iris.csv)

Program dapat menampilkan visualisasi dari model yang diberikan. Visualisasi tersebut menggunakan library "graphviz" dari Python, sehingga untuk menampilkan bentuk model diwajibkan melakukan instalasi library "graphviz". Keluaran dari visualisasi model ditampilkan dalam file dengan format PDF yang dibuat secara otomatis oleh program. Berikut adalah tampilan dari model yang diberikan dari hasil training menggunakan 3 layer dengan 2 neuron sigmoid, 1 neuron linear, dan 3 neuron relu.



Training akan melakukan tiga fase sesuai dengan yang sudah dijelaskan di kuliah yaitu fase forward propagation, fase backpropagation, dan fase update weight. Fase kedua dan ketiga

akan beriringan langsung dalam sebuah proses iterasi untuk setiap neuron. Berikut ini adalah fungsi utama yang menjalankan training model. Fungsi ini merupakan fungsi helper untuk main dalam menjalankan training model

Fungsi Learn

```
def learn(self, data):
         # placeholder
        current_iter = 0
        target = []
        result = []
        print("Length curr : ", len(self.current_layer))
        for _ in range(self.max_iter):
            error = 0.0
            for index, item in data.iterrows():
                # Prepare input
                self.enqueue_layer(InputLayer([item['sepal_length'],
item['sepal_width'], item['petal_length'], item['petal_width']]))
                # Forward and result
                self.forward_propagation()

                target.append([item['Class_1'], item['Class_2'], item['Class_3']])
                print("Target : ", target)
                result.append(self.current_layer[-1].result)
                if self.current_layer[-1].activation_function_name == "relu" or
self.current_layer[-1].activation_function_name == "sigmoid":
                    error += lossFunction(target, result, 3)
                elif self.current_layer[-1].activation_function_name == "softmax":
                    for i in range(len(target)):
                        for j in range(len(target[i])):
                            # print("inside : ", result[i][j])
                            if target[i][j] != result[i][j]:
                                error += lossSoftmax(result[i][j])
                print("error : ", error)
                if error < self.error_threshold:
                    break

                # cleaning layer
                self.deque_layer()

                # Learn with bach_size
                if (index + 1) % self.batch_size == 0 or index == len(data.index):
                    # backpropagation
                    self.back_propagation(target, result)
                    # clearing list and error foreach batch_size
                    target.clear()
```

```
                    result.clear()
                    error = 0

            if current_iter < self.max_iter:
                break
```

Jalannya fungsi utama ini adalah dengan mempersiapkan input dengan memasukan input layer ke model yang sudah dibuat. Ketika sudah terpasang input layer, maka model akan menjalankan forward propagation. Kemudian, model akan mencatat hasil dan target kelas dari dataset untuk instance tersebut. Setelah itu model akan menghitung error berdasar loss function yang bersesuaian dengan fungsi aktivasinya. Setelah dilakukan forward training, input layer akan dilepas dari model untuk persiapan dari instance berikutnya. Apabila proses forward training sudah sama dengan ukuran dari batch size maka akan dilakukan backpropagation sesuai dari hasil dan error yang sudah disimpan oleh model dari instance-instance sebelumnya. Jika sudah selesai maka instance dari hasil dan target akan direset dan error akan dikembalikan menjadi bernilai 0.

Berikut ini adalah fungsi backpropagation yang sudah diimplementasikan.

Fungsi Main

```
def back_propagation(self, arr_target, arr_out):
        for i in range(len(self.current_layer) - 1, -1, -1):
            if i != len(self.current_layer) - 2 and i > 0: # Not input or output
layer
                for j in range(len(self.current_layer[i].weight)):
                    for k in range(len(self.current_layer[i].weight[j])):
                        self.current_layer[i].weight[j][k] =
self.current_layer[i].update_weight(arr_target, arr_out,
                        self.current_layer[i].weight[j],
self.current_layer[i].result[j], self.current_layer[i].input_value[j],
self.learning_rate)
                for j in range(len(self.current_layer[i].bias)):
                    self.current_layer[i].bias =
self.current_layer[i].update_bias(arr_target, arr_out,
                    self.current_layer[i].bias, self.current_layer[i].result[j],
np.array([1 for x in range(len(self.current_layer[i].bias))]), self.learning_rate)
            elif i == len(self.current_layer):
                self.current_layer[i].weight =
self.current_layer[i].update_weight_output(arr_target, arr_out,
                        self.current_layer[i].weight,
```

```
self.current_layer[i].result[j], self.current_layer[i].input_value[j],
self.learning_rate)
```

Backpropagation akan melakukan prosesnya dari layer terbelakang. Kemudian untuk setiap layer tersebut terbagi menjadi beberapa test case:

1. Jika layer merupakan hidden layer, maka akan menggunakan update weight untuk hidden layer dan update bias. Update bias akan menggunakan array 1-d yang semua berisi angka 1 sebagai input. Hal ini dilakukan agar implementasi menyesuaikan cara kerja dari teori yang diajarkan di perkuliahan.
2. Jika layer merupakan output layer, maka akan menggunakan update weight untuk output layer

2. Hasil Eksekusi

Dengan menggunakan implementasi Backpropagation yang sudah dibuat, juga dengan menggunakan Feed Forward Neural Network pada tugas sebelumnya, berikut adalah hasil eksekusi menggunakan dataset Iris.

Parameter:

- Struktur jaringan : 3 hidden layer ⇒ [(3, sigmoid), (2, sigmoid), (3, sigmoid)]
- Learning rate : 0.001
- Error threshold : 0.001
- Max iteration : 100
- Batch size : 5

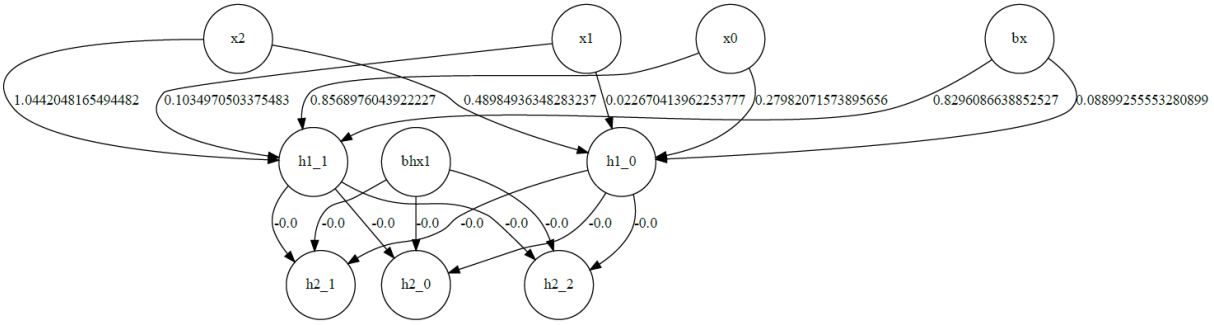
```
Learning Iris Dataset using Backpropagation + Feed Forward Neural Network

   neuron activation
0       3    sigmoid
1       2    sigmoid
2       3    sigmoid
-Learning rate : 0.001
-Error threshold : 0.001
-Maximum iteration : 100
-Batch Size : 3

Accuracy        :  0.3333333333333333
```

Berdasarkan learning tersebut didapatkan hasil prediksi akurasi sebesar 0,3333 dari 1 atau sebesar ⅓ dari data yang ada. Hasil prediksi tersebut didapatkan akibat target dataset yang

memiliki nilai hasil encoding sehingga nilai yang mungkin adalah sebesar 0,1,2 sedangkan prediksi menggunakan Neural Network lebih diuntungkan bila target dataset memiliki nilai biner. Berikut merupakan graph hasil neural network dari pembelajaran dataset iris.



Terlihat bahwa masing masing vektor h1 menuju *target class* h2 memiliki nilai 0 akibat hasil encoding pada target class sehingga nilai yang didapatkan bukan merupakan nilai biner dan tidak merepresentasikan *insight* dari target class.

3. Perbandingan Pembelajaran

3.1. Pembelajaran dengan Implementasi Backpropagation

a. Pembelajaran Uji 1

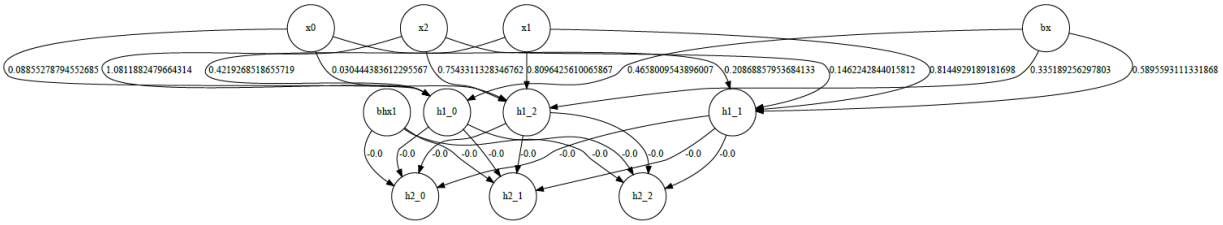
layerSize : 1; **activationFunction** : sigmoid; **solverMethod** : sgd; **batchSize** : 2; **learningRateMethod** : constant; **learningRate** : 0.001; **iterationSize** : 50; **error_threshold** : 0.001;

```
Learning Iris Dataset using Backpropagation + Feed Forward Neural Network

   neuron activation
0       3    sigmoid
1       3    sigmoid
2       3    sigmoid

-Learning rate : 0.001
-Error threshold : 0.001
-Maximum iteration : 50
-Batch Size : 2

Accuracy        :  0.3333333333333333
```



b. Pembelajaran Uji 2

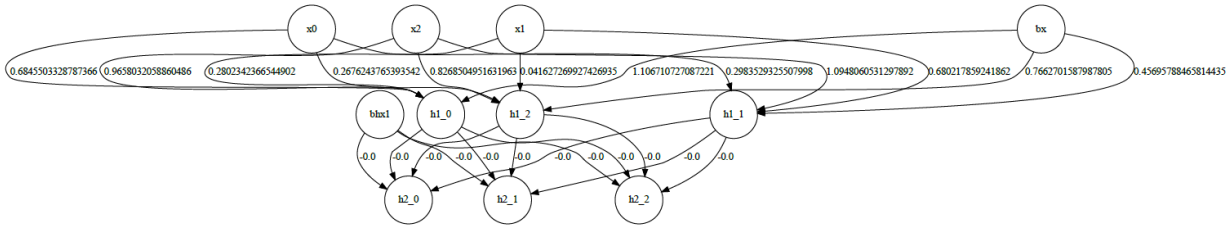
layerSize : 3; **activationFunction** : sigmoid; **solverMethod** : sgd; **batchSize** : 2; **learningRateMethod** : constant; **learningRate** : 0.05; **iterationSize** : 100; **error_threshold** : 0.05;

```
Learning Iris Dataset using Backpropagation + Feed Forward Neural Network

   neuron activation
0       3    sigmoid
1       3    sigmoid
2       3    sigmoid

-Learning rate : 0.05
-Error threshold : 0.05
-Maximum iteration : 100
-Batch Size : 2

Accuracy        :  0.3333333333333333
```



c. Pembelajaran Uji 3

layerSize : 3; **activationFunction** : relu; **solverMethod** : sgd; **batchSize** : 3; **learningRateMethod** : constant; **learningRate** : 0.001; **iterationSize** : 50; **error_threshold** : 0.05;

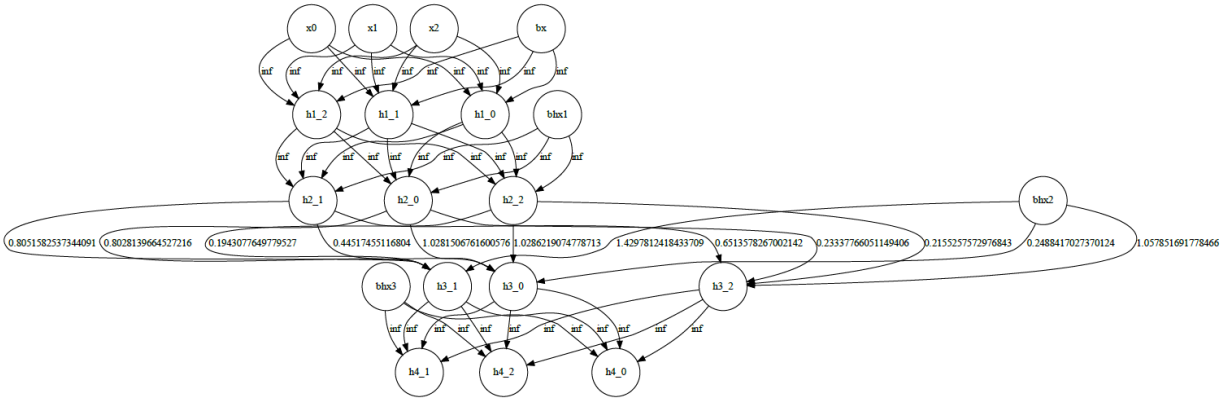
```
Learning Iris Dataset using Backpropagation + Feed Forward Neural Network

   neuron activation
0       3       relu
1       3       relu
2       3       relu
3       3       relu
4       3       relu

-Learning rate : 0.001
-Error threshold : 0.05
-Maximum iteration : 50
-Batch Size : 3
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python36\site-packages\ipykernel_launcher.py:43: RuntimeWarning: overflow encountered in d
ouble_scalars
C:\Users\ASUS\AppData\Roaming\Python\Python36\site-packages\ipykernel_launcher.py:27: RuntimeWarning: overflow encountered in d
ouble_scalars
C:\Users\ASUS\AppData\Roaming\Python\Python36\site-packages\ipykernel_launcher.py:171: RuntimeWarning: overflow encountered in
matmul
```

```
Accuracy        :  0.3333333333333333
```



3.2. Pembelajaran dengan MLP Sklearn

Untuk melakukan pengujian pembelajaran dengan MLP Sklearn, berikut adalah kode program yang digunakan.

MLP Sklearn Testing

```
from sklearn.datasets import load_iris
from sklearn.neural_network import MLPClassifier

data = load_iris(as_frame = True)
full_data_X, full_data_Y = load_iris(return_X_y = True, as_frame=True)

clf = MLPClassifier(hidden_layer_sizes = layerSize, activation =
activationFunction, solver = 'solverMethod, batch_size = batchSize,
learning_rate = learningRateMethod, learning_rate_init = learningRate,
max_iter = iterationSize, tol = error_threshold)
```

```
clf.fit(full_data_X, full_data_Y)
clf.predict(full_data_X)
clf.score(full_data_X, full_data_Y)
```

Keterangan :

- **layerSize** : ukuran banyaknya hidden layer pada model
- **activationFunction** : pilihan fungsi aktivasi yang digunakan ('identity' = linear, 'logistic' = sigmoid, 'relu', 'tanh')
- **solverMethod** : metode solver yang digunakan. Dalam pengujian digunakan 'sgd'
- **batchSize** : banyaknya iterasi dalam satu batch
- **learningRateMethod** : metode *updating* learning rate. Dalam pengujian digunakan 'constant'
- **learningRate** : besar learning rate yang digunakan
- **iterationSize** : banyaknya iterasi
- **error_threshold** : batasan error yang digunakan dalam pembelajaran. Nilai default adalah 0.0001

Dengan menggunakan kode yang telah dibuat, berikut adalah hasil pembelajaran Backpropagation menggunakan MLP Sklearn menggunakan dataset Iris.

a. Pembelajaran Uji 1

layerSize : 1; **activationFunction** : logistic; **solverMethod** : sgd; **batchSize** : 2; **learningRateMethod** : constant; **learningRate** : 0.001; **iterationSize** : 50; **error_threshold** : 0.001;

In [3]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier

data = load_iris(as_frame = True)
full_data_X, full_data_Y = load_iris(return_X_y = True, as_frame=True)

clf = MLPClassifier(hidden_layer_sizes = 1, activation = 'logistic', solver = 'sgd', batch_size = 2,
                    learning_rate = 'constant', learning_rate_init = 0.001, max_iter = 50, tol = 0.001)

clf.fit(full_data_X, full_data_Y)
clf.predict(full_data_X)
clf.score(full_data_X, full_data_Y)
```

```
C:\Python 36-64\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:585: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (50) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

Out[3]: 0.7

b. Pembelajaran Uji 2

layerSize : 3; **activationFunction** : logistic; **solverMethod** : sgd; **batchSize** : 2; **learningRateMethod** : constant; **learningRate** : 0.05; **iterationSize** : 100; **error_threshold** : 0.05;

```
In [2]: from sklearn.datasets import load_iris
        from sklearn.model_selection import train_test_split
        from sklearn.neural_network import MLPClassifier

        data = load_iris(as_frame = True)
        full_data_X, full_data_Y = load_iris(return_X_y = True, as_frame=True)

        clf = MLPClassifier(hidden_layer_sizes = 3, activation = 'logistic', solver = 'sgd', batch_size = 2,
                            learning_rate = 'constant', learning_rate_init = 0.05, max_iter = 100, tol = 0.05)

        clf.fit(full_data_X, full_data_Y)
        clf.predict(full_data_X)
        clf.score(full_data_X, full_data_Y)

Out[2]: 0.6666666666666666
```

c. Pembelajaran Uji 3

layerSize : 3; **activationFunction** : relu; **solverMethod** : sgd; **batchSize** : 3; **learningRateMethod** : constant; **learningRate** : 0.001; **iterationSize** : 50; **error_threshold** : 0.05;

```
In [19]: from sklearn.datasets import load_iris
         from sklearn.model_selection import train_test_split
         from sklearn.neural_network import MLPClassifier

         data = load_iris(as_frame = True)
         full_data_X, full_data_Y = load_iris(return_X_y = True, as_frame=True)

         clf = MLPClassifier(hidden_layer_sizes = 3, activation = 'relu', solver = 'sgd', batch_size = 3,
                             learning_rate = 'constant', learning_rate_init = 0.001, max_iter = 50, tol = 0.05)

         clf.fit(full_data_X, full_data_Y)
         clf.predict(full_data_X)
         clf.score(full_data_X, full_data_Y)

Out[19]: 0.8533333333333334
```

3.3. Kesimpulan Perbandingan

Dari pengujian pembelajaran yang telah dilakukan, terdapat perbedaan output yang signifikan dari hasil implementasi backpropagation dan MLP Sklearn. Pada hasil implementasi backpropagation, terlihat keluaran yang dihasilkan konstan pada angka 0.333333…. , yang mana hasil tersebut tidak mungkin seperti itu. Hasil tersebut merupakan *bug* yang terjadi saat mengimplementasi backpropagation, sehingga keluaran yang dihasilkan menjadi tidak seharusnya. Setelah ditelusuri, *bug* tersebut bisa terjadi akibat one-hot encoder yang digunakan saat melakukan pembelajaran.

Namun jika dilihat dari pembelajaran menggunakan MLP Sklearn, keluaran yang dihasilkan menjadi akurat. Ketika digunakan maksimum iterasi dengan nilai yang kecil, akan

menyebabkan pembelajaran berhenti pada titik tertentu sebelum mencapai konvergen, sehingga akan muncul error seperti yang terlihat pada pembelajaran uji 1. Selain berhenti karena maksimum iterasi telah dicapai, pembelajaran dapat berhenti ketika error yang dicapai mencapai nilai kurang dari threshold yang ditentukan. Selain itu, nilai learning rate juga berpengaruh dalam melakukan pembelajaran, semakin kecil nilai learning rate, prediksi yang dihasilkan makin akurat.

Saat melakukan pengujian, ternyata untuk nilai parameter yang sama, hasil yang dikeluarkan bisa berbeda. Hal ini terjadi karena inisiasi bobot dari neuron pada MLP Sklearn dilakukan secara random, sehingga bisa jadi keluaran yang dihasilkan berbeda pula, tergantung pada inisiasi bobot yang didapat pada saat itu. Dengan nilai parameter lain, terutama dengan nilai maksimum iterasi yang besar, akurasi yang dihasilkan bisa mencapai nilai 1, yang menunjukkan seluruh prediksi hasil pembelajaran memiliki nilai yang sama dengan target.

4. Pembagian Kerja

NIM	Nama	Kontribusi
13518033	Daffa Pratama Putra	- Mengerjakan laporan - Melakukan pembelajaran dengan MLP Sklearn - Membandingkan pembelajaran Sklearn dengan implementasi - Melakukan pengujian
13518072	Muchammad Ibnu Sidqi	- Mengerjakan laporan - Mengimplementasi turunan fungsi
13518117	Muhammad Firas	- Mengerjakan laporan - Mengimplementasi aturan rantai
13518141	Muhammad Daffa Dinaya	- Mengerjakan laporan - Mengimplementasi program utama backpropagation - Integrasi aturan rantai dan turunan fungsi