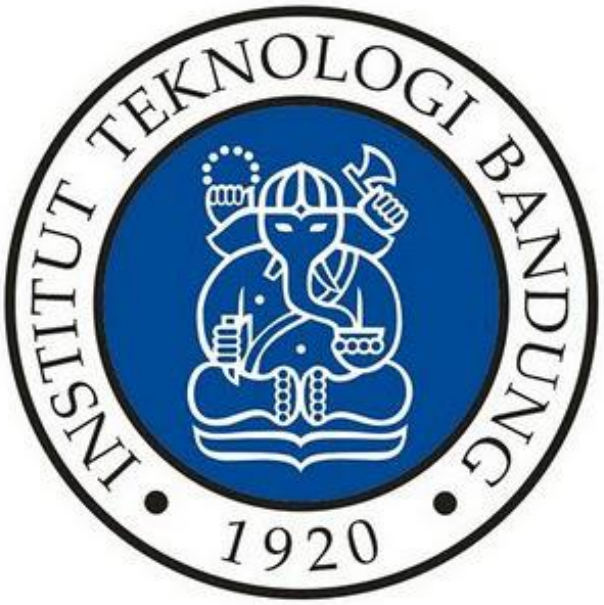
IF3270 Pembelajaran Mesin

Tugas Besar Bagian B: Implementasi Mini-batch Gradient Descent



Disusun oleh:

Daffa Ananda Pratama Resyaly	13519107
Raihan Astrada Fathurrahman	13519113
M. Ibnu Syah Hafizh	13519177
Muhammad Rayhan Ravianda	13519201

PROGRAM STUDI INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2019

I. Penjelasan Implementasi

Implementasi backpropagation dengan mini-batch gradient descent dalam tugas besar bagian B ini menggunakan program dalam bahasa python. Program yang dibangun memanfaatkan library numpy dan math. Pada program ini dibuat 2 buah kelas yaitu kelas **Layer** yang merepresentasikan sebuah Layer dalam Neural Network, serta kelas **NeuralNetwork** yang merepresentasikan sebuah Neural Network.

Kelas **Layer** merupakan sebuah layer yang memiliki atribut ***weight*** yang berarti bobot untuk setiap neuron, bias yang merupakan nilai ***bias*** setiap neuron, ***activation*** yang berarti fungsi aktivasi pada layer tersebut, **delta_weight** untuk menyimpan perubahan bobot , dan delta_bias yang digunakan untuk menyimpan perubahan bias. Pada kelas **Layer**, terdapat method atau fungsi forward propagation dengan nama fungsi ***forward_propagation(input)***. Method ini menerima input berupa data masukkan array (dapat berupa matriks, array n dimensi) dan akan mengembalikan nilai dari perhitungan fungsi aktivasi pada suatu layer yang sudah disesuaikan dengan fungsi aktivasi yang digunakan (dalam program ini yang dikenali adalah ***Linear***, ***Sigmoid***, ***ReLU***, dan ***Softmax***) dengan input fungsi aktivasi adalah hasil perhitungan perkalian dot antara weight atau matriks bobot dengan input awal pada fungsi forward propagation. Setiap fungsi aktivasi mengembalikan nilai menggunakan perhitungan pada setiap fungsi aktivasi yang valid. Method kedua yang terdapat pada kelas **Layer** adalah **net(input).** Method ketiga yang terdapat dalam kelas Layer adalah **calculate_error.** Method **calculate_error** digunakan untuk menghitung nilai error dengan fungsi loss yang tepat untuk tiap aktivasi. Method **calculate_error** memiliki dua parameter, yaitu target dan output yang digunakan untuk perhitungan dalam fungsi loss. Terdapat dua fungsi loss yang diimplementasikan yaitu *sum of squad errors* (untuk linear, sigmoid, ReLu), dan dan *cross enthropy* (untuk softmax).

Kelas **NeuralNetwork** memiliki atribut ***layers*** yang merupakan array of layer yang digunakan untuk menyimpan layer-layer yang terdapat pada file eksternal, ***learning_rate*** yang merupakan *learning rate****, error_threshold*** yang merupakan *error threshold*, ***max_iter*** yang menyimpan nilai iterasi maksimum, dan ***batch_size*** yang merupakan ukuran *batch*. Pada kelas **NeuralNetwork**, method yang terdapat dalam kelas ini yang pertama adalah ***loadfile(filename)*** yang digunakan untuk membaca dan memproses file eksternal menjadi layer-layer dan menambahkan layer-layer tersebut ke dalam atribut layers dengan menggunakan operasi ***add(layer)***. Method kedua adalah ***summary()***, yang digunakan untuk menampilkan model struktur dan koefisien dari file eksternal (beban) yang diinput dengan format yang telah ditentukan. Method ketiga yaitu method ***predict(input)*** yang digunakan untuk memprediksi output dari nilai yang diinput berdasarkan layers yang terdapat pada kelas. Method keempat yaitu ***shuffle(x, y)*** yang digunakan untuk mengacak data. Method kelima yaitu ***create_batch(x, y)*** yang digunakan untuk membagi data ke dalam batch dengan ukuran batch yang sudah ditentukan. Method keenam yaitu ***backward_propagation(x, y, output),*** yang digunakan untuk menghitung gradien dari setiap neuron pada suatu layer. Pada method ini, dihitung nilai gradien berdasarkan informasi yang dikandung dari layer dan akan mengembalikan nilai-nilai gradient, serta menyimpan nilai *delta weight* dan d*elta bias*. Method terakhir pada kelas **NeuralNetwork** adalah **mgd(x, y, output)** yaitu method yang merupakan implementasi dari mini-batch gradient descent. Pada method ini, dilakukan

iterasi sebanyak nilai maksimum. Setiap iterasi, dilakukan pembagian ke batch dengan method ***create_batch***. Selanjutnya dilakukan *forward propagation,* dihitung nilai errornya, dan dilakukan *backward propagation* untuk menghitung gradien dan mendapatkan nilai delta weight dan delta bias. Lalu dilakukan update bobot dengan menambahkan bobot lama dengan delta bobot. Kemudian, dicek nilai error, jika nilai error masih lebih besar dari nilai *error threshold* maka dilakukan iterasi dilanjutkan, tetapi jika nilai error sudah lebih kecil dari nilai *error threshold* maka program berhenti. Program berhenti jika nilai error lebih kecil dari nilai *error threshold* atau iterasi sudah mencapai maksimum iterasi.

II. Hasil Pengujian

```python
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Python

```
(75, 4)
(75, 4)
(75, 3)
(75, 3)
```

```python
model = NeuralNetwork(learning_rate=0.001, max_iteration=2000, verbose=False)

#input of first layer must match the num of feature in the training data
model.add(Layer("relu", 4, 10))

model.add(Layer("relu", 10, 10))
model.add(Layer("linear", 10, 5))

#output of the layer must match the num of feature in the target classes
model.add(Layer("sigmoid", 5, 3))

# print(model)

model.fit(X_train, y_train)
```

Python

```
Output exceeds the size limit. Open the full output data in a text editor
Iteration 0:  0.6578218139475396
Iteration 50:  0.14084565614572248
Iteration 100:  0.050456121526701814
Iteration 150:  0.05335024052797809
Iteration 200:  0.06509370059047964
Iteration 250:  0.038995809443770076
Iteration 300:  0.041143408156487336
Iteration 350:  0.04512352965430908
Iteration 400:  0.03092923197487617
Iteration 450:  0.05232508902201129
Iteration 500:  0.042087456389641266
Iteration 550:  0.043413144113871746
Iteration 600:  0.06687252928417363
Iteration 650:  0.03058305985015891
Iteration 700:  0.0475815445226404
```

```
Iteration 700:  0.0475815445226404
Iteration 750:  0.04518812200160402
Iteration 800:  0.03103301965648578
Iteration 850:  0.039894861913848806
Iteration 900:  0.036040152908312695
Iteration 950:  0.04547905479936319
Iteration 1000:  0.041150196117991596
Iteration 1050:  0.041444789189172215
Iteration 1100:  0.10389650862409752
Iteration 1150:  0.019049489409866952
Iteration 1200:  0.04603328763385964
...
Iteration 1400:  0.033776849069734244
Iteration 1450:  0.04583432853728112
Iteration 1500:  0.03664553830072761
Iteration 1550:  0.03449236485345259
```

```python
label_pred = []
for i in range(prediction.shape[1]):
    label_pred.append(np.argmax(prediction[:, i]))

y_test_label = []
for i in range(y_test.shape[0]):
    y_test_label.append(np.argmax(y_test[i, :]))
```

Python

```python
print(label_pred)
print(y_test_label)
```

Python

```
[1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0, 0, 1, 2, 2, 1, 2, 1, 2, 1, 0, 2, 1, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 1, 2, 0, 1, 2, 0, 2, 2, 1]
[1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0, 0, 1, 2, 2, 1, 2, 1, 2, 1, 0, 2, 1, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 1, 2, 0, 1, 2, 0, 2, 2, 1]
```

```python
print(accuracy_score(label_pred_sklearn, y_test_label))
```

Python

```
0.96
```

```python
print(accuracy_score(label_pred, label_pred_sklearn))
```

Python

```
0.96
```

```python
#input of first layer must match the num of feature in the training data
model2.add(Layer("relu", 4, 10))

model2.add(Layer("relu", 10, 10))
model2.add(Layer("linear", 10, 5))

#output of the layer must match the num of feature in the target classes
model2.add(Layer("sigmoid", 5, 3))

# print(model)

model2.fit(X, y)
```

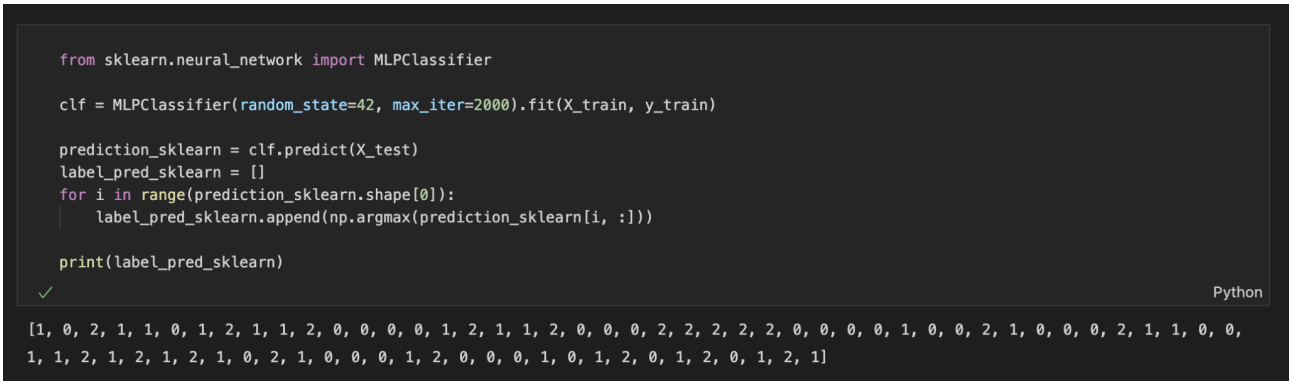
Python

```
Iteration 0:  0.5059584328878654
Iteration 50:  0.031658263747407975
Iteration 100:  0.020376098416943114
Iteration 150:  0.02395428414113574
Iteration 200:  0.02995798374933625
Iteration 250:  0.02120978283984016
Iteration 300:  0.032930467118524584
Iteration 350:  0.028317625521206587
Iteration 400:  0.024656146502651258
Iteration 450:  0.020596835693988277
Iteration 500:  0.030344388720661756
Iteration 550:  0.033443223301497904
Iteration 600:  0.022212040382965162
Iteration 650:  0.030205516944919596
Iteration 700:  0.024654119732987256
Iteration 750:  0.029273806664283145
Iteration 800:  0.015438752268958564
Iteration 850:  0.013209640114342603
```

III. Perbandingan dengan Hasil MLP sklearn

Selanjutnya adalah perbandingan nilai antara hasil perhitungan MLP sklearn dengan hasil perhitungan program.

Berikut adalah hasil perhitungan MLP sklearn,

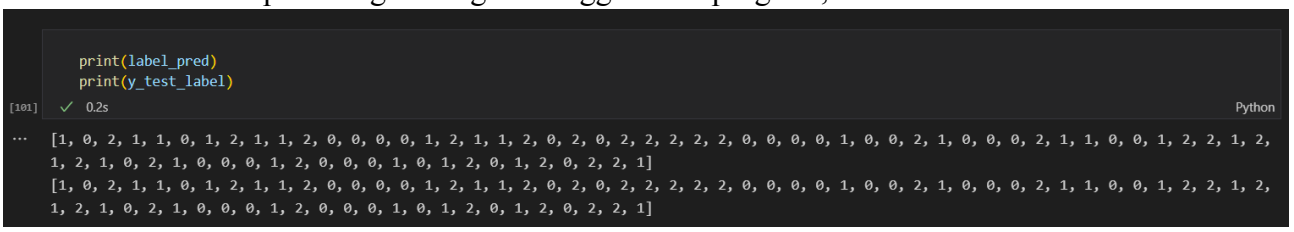


```
from sklearn.neural_network import MLPClassifier

clf = MLPClassifier(random_state=42, max_iter=2000).fit(X_train, y_train)

prediction_sklearn = clf.predict(X_test)
label_pred_sklearn = []
for i in range(prediction_sklearn.shape[0]):
    label_pred_sklearn.append(np.argmax(prediction_sklearn[i, :]))

print(label_pred_sklearn)
```

Python

```
[1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0, 0,
1, 1, 2, 1, 2, 1, 2, 1, 0, 2, 1, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 1, 2, 0, 1, 2, 0, 1, 2, 1]
```

Berikut adalah hasil perhitungan dengan menggunakan program,



```
print(label_pred)
print(y_test_label)
```

[101] ✓ 0.2s Python

```
[1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0, 0, 1, 2, 2, 1, 2,
1, 2, 1, 0, 2, 1, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 1, 2, 0, 1, 2, 0, 2, 2, 1]
[1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0, 0, 1, 2, 2, 1, 2,
1, 2, 1, 0, 2, 1, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 1, 2, 0, 1, 2, 0, 2, 2, 1]
```

Berikut adalah perbandingan akurasi antara program yang dibangun dengan MLP sklearn

```
print(accuracy_score(label_pred_sklearn, y_test_label))
```

✓ Python

```
0.96
```

```
print(accuracy_score(label_pred, label_pred_sklearn))
```

✓ Python

```
0.96
```

Berdasarkan hasil perhitungan MLP sklearn dan perhitungan menggunakan program pada kasus uji yang sama, dapat dilihat bahwa nilai perhitungan yang dihasilkan oleh program sama dengan nilai perhitungan yang dihasilkan oleh perhitungan MLP sklearn sehingga dapat disimpulkan bahwa program yang dibangun sudah tepat mengimplementasikan backward propagation dengan mini-batch gradient descent.

IV. Pembagian Tugas

Nama	**NIM**	**Tugas**
Daffa Ananda Pratama Resyaly	13519107	Membuat fungsi *sum of squared errors*; Backpropagation, Debugging; Laporan
Raihan Astrada Fathurrahman	13519113	Membuat fungsi derivative linear, Sigmoid, Relu; Fungsi; Backpropagation; Debugging; Laporan
M. Ibnu Syah Hafizh	13519177	Membuat fungsi loss *cross enthropy*; Backpropagation; Laporan
Muhammad Rayhan Ravianda	13519201	Membuat fungsi derivative softmax; Backpropagation ; Laporan

V. Lampiran

Github:
https://github.com/slarkdarr/Tubes-ML.git