LAPORAN TUGAS BESAR BAGIAN B
IF3270 PEMBELAJARAN MESIN

ARTIFICIAL NEURAL NETWORK

Diajukan sebagai tugas besar Mata Kuliah IF3270 Pembelajaran Mesin

Semester II Tahun Akademik 2023/2024

Anggota Kelompok:

Jason Rivalino	13521008
Muhammad Rifko Favian	13521075
Moch. Sofyan Firdaus	13521083
Fazel Ginanda	13521098

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023/2024

BAB I
DESKRIPSI TUGAS

Tugas besar kali ini adalah untuk membuat implementasi *backpropagation* dari modul FFNN yang sebelumnya telah dibuat. Pada pengerjaan kali ini, terdapat beberapa spesifikasi dan batasan yang perlu dipenuhi antara lain sebagai berikut:

Spesifikasi

1. Backpropagation dapat melakukan update weight saat training secara mini-batch, yang diatur melalui parameter **batch_size**.
2. Algoritma fungsi aktivasi yang diimplementasikan adalah ReLU, sigmoid, linear, dan Softmax.
    a. Turunan fungsi aktivasi:

Nama Fungsi Aktivasi	Turunan
ReLU	$\frac{d}{dx}ReLU(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geqslant 0 \end{cases}$
Sigmoid	$\frac{d}{dx}\sigma(x) = \sigma(x) \times (1 - \sigma(x))$
Linear	$\frac{d}{dx}x = 1$
Softmax*	$\frac{\partial E_d}{\partial net(x)} = \begin{cases} p_j, & j \neq \text{target} \\ -(1 - p_j), & j = \text{target} \end{cases}$

    b. Fungsi loss:

Fungsi Aktivasi	Loss

ReLU, sigmoid, dan linear	$E = \frac{1}{2} \sum_{k \in output} (t_k - o_k)^2$
Softmax	$E = -log(p_k),$ k=target

3. Update weight pada gradient descent dilakukan dengan aturan rantai.
    a. Persamaan untuk update weight

$$\Delta w = -\text{gradient} \times \text{learning rate}$$
$$w_{new} = w_{old} + \Delta w$$

    b. Persamaan gradient pada hidden layer berbeda dengan output layer
        1) Hidden layer

$$\frac{\mathrm{d}E}{\mathrm{d}w} = \frac{\mathrm{d}E}{\mathrm{d}net} \frac{\mathrm{d}net}{\mathrm{d}w}$$

        2) Output layer

$$\frac{\mathrm{d}E}{\mathrm{d}w} = \frac{\mathrm{d}E}{\mathrm{d}Out} \frac{\mathrm{d}Out}{\mathrm{d}net} \frac{\mathrm{d}net}{\mathrm{d}w}$$

        Pengecualian pada **softmax** karena langsung berbentuk dE/dnet sehingga pada softmax dE/dw = dE/dnet * dnet/dw

4. Kondisi berhenti training terjadi ketika **error kumulatif <= threshold** atau **iterasi maksimum** tercapai. Error threshold dan iterasi maksimum merupakan parameter yang dapat diubah.
5. Definisikanlah parameter apa saja yang bisa ditangani dalam implementasi kelompok Anda. Parameter yang wajib ada adalah: struktur jaringan (jumlah layer, jumlah neuron setiap layer, fungsi aktivasi setiap layer), learning_rate, error_threshold, max_iter, batch_size.

Batasan

1. Satu layer memiliki fungsi aktivasi yang sama; layer yang berbeda dapat memiliki fungsi aktivasi yang berbeda.

BAB II
PENJELASAN IMPLEMENTASI

Untuk implementasi dari program yang dibuat, program dibuat dengan memanfaatkan library numpy untuk melakukan operasi perhitungan dan juga json untuk memasukkan file JSON ke dalam program.

Adapun untuk keseluruhan struktur kode, program yang ada sama seperti pada pengerjaan tugas sebelumnya yaitu juga dibagi menjadi dua kelas yaitu kelas FFNN untuk melakukan proses pencarian nilai output dengan mengimplementasikan Mini-Batch Gradient Descent secara *Backward Propagation* dan juga kelas Layer untuk menentukan memproses hasil pada setiap layer berdasarkan fungsi aktivasi yang terdapat pada input file.

Proses yang terjadi yaitu dimulai dari pembacaan file test dengan format JSON pada program. Dari hasil pembacaan file tersebut kemudian untuk berbagai macam model akan didefinisikan nilai-nilai yang diperlukan untuk menjalankan FFNN antara lain:

1. Input_size: jumlah ukuran data input pada program
2. Layers: informasi untuk setiap layers (ada jumlah neuron dan juga jenis fungsi aktivasi)
3. Learning_rate: kecepatan dari pembelajaran pada model
4. Epoch: jumlah iterasi maksimum yang ada pada model
5. Batch_size: ukuran batch yang ada pada model
6. Threshold: nilai dari batas kesalahan yang digunakan untuk membatasi iterasi
7. Initial_weights: nilai bobot awal yang ada pada model

Dari berbagai variabel yang diinisiasi ini, kemudian akan dimasukkan ke dalam fungsi FFNN. Untuk setelahnya, terdapat pembagian data yang ada berdasarkan file JSON yaitu untuk data training (X_train) dan juga data test (y_train). Setelah pembagian dua jenis data ini kemudian akan dilakukan proses menjalankan fungsi train pada kelas FFNN untuk mendapatkan hasil output dengan proses sebagai berikut:

1. Inisiasi bias dan bobot (*weights*) pada bobot awal dari file JSON dengan bias merupakan data pada baris pertama dan weights merupakan data sisanya
2. Melakukan proses Mini-Batch Gradient Descent untuk setiap inisiasi epoch pada file

3. Membagi training data dan test data ke dalam bentuk mini-batch untuk dilakukan pemrosesan
4. Menjalankan *forward propagation* untuk training data dengan proses mengalikan data input dengan data weights yang kemudian hasilnya ditambahkan dengan data bias
5. Menjalankan *backward propagation* untuk test data dengan proses untuk tiap layer dari yang terakhir hingga pertama, lalu menghitung nilai delta (dibedakan berdasarkan kondisi layer dan juga fungsi aktivasi) yang kemudian hasil delta yang ada akan dipergunakan untuk memperbaharui nilai bobot (*weights*) dan bias
6. Melakukan proses penambahan terhadap nilai error secara terus menerus tiap iterasi epoch
7. Kondisi training untuk pencarian bobot akhir berhenti ketika nilai error sudah kurang dari sama dengan nilai threshold ataupun ketika iterasi maksimum sudah tercapai
8. Mengeluarkan nilai output untuk hasil bobot akhir dan mengecek apakah nilai yang ada sudah sesuai dengan hasil yang ada pada final_weights pada file JSON dengan metode SSE

BAB III
HASIL PENGUJIAN DAN PERBANDINGAN

A. Hasil Pengujian

1. Pengujian dengan file linear_small_lr.json

   Hasil harapan:

```
"expect": {
        "stopped_by": "max_iteration",
        "final_weights": [
            [
                [ 0.1008,  0.3006, 0.1991],
                [ 0.402,  0.201, -0.7019],
                [ 0.101, -0.799,  0.4987]
            ]
        ]
    }
```

   Nilai output yang didapat dari program:

```
Stopped by max_iteration
Final Weights:
[0.101200 0.300600 0.199100]
[0.402400 0.201000 -0.701900]
[0.101800 -0.799000 0.498700]

Test case failed: Final weight at layer 0 does not match. SSE = 0.00000096
```

   Note: untuk testcase ini nilai SSE masih ditoleransi (berdasarkan QnA)

2. Pengujian dengan file linear_two_iteration.json

   Hasil harapan:

```
"expect": {
        "stopped_by": "max_iteration",
        "final_weights": [
            [
              [ 0.166,  0.338, 0.153],
              [ 0.502,  0.226, -0.789],
              [ 0.214, -0.718,  0.427]
```

```
            ]
        ]
    }
```

Nilai output yang didapat dari program:

```
Stopped by max_iteration
Final Weights:
[0.166000 0.338000 0.153000]
[0.502000 0.226000 -0.789000]
[0.214000 -0.718000 0.427000]

All test case passed!
```

3. Pengujian dengan file linear.json

   Hasil harapan:

```
"expect": {
        "stopped_by": "max_iteration",
        "final_weights": [
            [
                [ 0.22,  0.36, 0.11],
                [ 0.64,  0.3, -0.89],
                [ 0.28, -0.7,  0.37]
            ]
        ]
    }
```

   Nilai output yang didapat dari program:

```
Stopped by max_iteration
Final Weights:
[0.220000 0.360000 0.110000]
[0.640000 0.300000 -0.890000]
[0.280000 -0.700000 0.370000]

All test case passed!
```

4. Pengujian dengan file mlp.json

   Hasil harapan:

```
"expect": {
        "stopped_by": "max_iteration",
        "final_weights": [
```

```
            [
                [0.08592, 0.32276],
                [-0.33872, 0.46172],
                [0.449984, 0.440072]
            ],
            [
                [0.2748, 0.188],
                [0.435904, -0.53168],
                [0.68504, 0.7824]
            ]
        ]
    }
```

Nilai output yang didapat dari program:

```
Stopped by max_iteration
Final Weights:
[0.085818 0.320092]
[-0.341963 0.462529]
[0.453309 0.441397]

[0.274800 0.188000]
[0.435904 -0.531680]
[0.685040 0.782400]

Test case failed: Final weight does not match. SSE = 0.00003111
```

5. Pengujian dengan file relu_b.json

   Hasil harapan:

```
"expect": {
        "stopped_by": "max_iteration",
        "final_weights": [
            [
                [-0.211, 0.105, 0.885],
                [0.3033, 0.5285, 0.3005],
                [-0.489, -0.905, 0.291]
            ]
        ]
    }
```

Nilai output yang didapat dari program:

```
Stopped by max_iteration
Final Weights:
[-0.211000 0.105000 0.885000]
[0.303300 0.528500 0.300500]
[-0.489000 -0.905000 0.291000]

All test case passed!
```

6. Pengujian dengan file sigmoid.json

   Hasil harapan:

```
"expect": {
        "stopped_by": "max_iteration",
      "final_weights":[
          [
             [0.2329, 0.0601],
             [0.1288, 0.6484],
             [0.8376, 0.2315]
          ]
      ]
    }
```

   Nilai output yang didapat dari program:

```
Stopped by max_iteration
Final Weights:
[0.232912 0.060153]
[0.128841 0.648495]
[0.837615 0.231582]

All test case passed!
```

7. Pengujian dengan file softmax_two_layer.json

   Hasil harapan:

```
"expect": {
        "stopped_by": "error_threshold",
        "final_weights": [
```

```
            [
                            [-0.28730211,  -0.28822282,  -0.70597451,
  0.42094471],
                           [-0.5790794,   -1.1836444,   -1.34287961,
  0.69575311],
                            [-0.41434377,   1.51314676,  -0.97649086,
  -1.3043465 ]
            ],
            [
                [-1.72078607,  1.74078607],
                [-0.50352956,  0.48352956],
                [ 1.25764816, -1.23764816],
                [-1.16998784,  1.14998784],
                [ 1.0907634,  -1.0707634 ]
            ]
        ]
    }
```

Nilai output yang didapat dari program:

```
Stopped by error_threshold
Final Weights:
[-0.102571 0.021357 -0.437302 0.379530]
[-0.684830 -1.307549 -1.515447 0.648185]
[-0.386470 1.541891 -0.943235 -1.208983]

[-1.371843 1.391843]
[-0.434833 0.414833]
[1.130974 -1.110974]
[-1.039429 1.019429]
[0.981249 -0.961249]

Test case failed: Final weight does not match. SSE = 0.27424629
```

8. Pengujian dengan file softmax.json

   Hasil harapan:

```
"expect": {
        "stopped_by": "max_iteration",
        "final_weights": [
            [
```

```
                [ 0.12674605,  0.9149538,  -0.14169985],
                [-0.33551647,  0.67700488,  0.45851159],
                [ 0.48314436, -0.85241216,  0.2692678 ],
                [ 0.3400255,   0.57237542, -0.31240092],
                [ 0.31397716,  0.46349737,  0.72252547],
                [-0.69652442,  0.4789189,   0.61760552],
                [-0.50884515, -0.36354141,  0.57238656],
                [ 0.41891295,  0.26354517, -0.48245812],
                [ 0.90374164, -0.01759501, -0.08614663]
            ]
        ]
    }
```

Nilai output yang didapat dari program:

```
Stopped by max_iteration
Final Weights:
[0.126746 0.914954 -0.141700]
[-0.335516 0.677005 0.458512]
[0.483144 -0.852412 0.269268]
[0.340026 0.572375 -0.312401]
[0.313977 0.463497 0.722525]
[-0.696524 0.478919 0.617606]
[-0.508845 -0.363541 0.572387]
[0.418913 0.263545 -0.482458]
[0.903742 -0.017595 -0.086147]

All test case passed!
```

BAB IV
PEMBAGIAN KERJA

Nama	NIM	Pembagian Kerja
Jason Rivalino	13521008	Implementasi Backpropagation, Aktivasi ReLU, Hasil Output, Laporan
Muhammad Rifko Favian	13521075	Implementasi Backpropagation, Aktivasi Softmax Mini-Batch, Pengujian Library, Laporan
Moch. Sofyan Firdaus	13521083	Implementasi Mini-Batch, Aktivasi Sigmoid, Input File, Laporan
Fazel Ginanda	13521098	Implementasi Mini-Batch, Aktivasi Linear, Input File, Laporan