Tugas Besar B
IF3270 Pembelajaran Mesin

Implementasi *Backpropagation* dengan *Mini-batch Gradient Descent*

Disusun oleh:

13519096 Girvin Junod

13519116 Jeane Mikha Erwansyah

13519131 Hera Shafira

13519188 Jeremia Axel Bachtera

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Semester 2 Tahun 2021/2022

1. Penjelasan Program

Program ini adalah implementasi *backpropagation* untuk *Feedforward Neural Network* (FFNN). Dalam program ini, digunakan *compact model* untuk merepresentasikan FFNN di dalam program. Ini berarti *layer* yang berisi neuron direpresentasikan dengan matriks.

Dalam implementasi program di bahasa Python ini, dibuat dua *class* yaitu *class* `Layer` dan `Graph` sebagai model dari FFNN di dalam program. *Class* Layer adalah *class* representasi dari *layer* pada FFNN. *Class* ini memiliki atribut berupa `weights` yaitu *weight dari* neuron-neuron yang ada di *layer*, `num_nodes` yang merupakan banyaknya node dalam suatu layer, `activation_func` yang menandakan fungsi aktivasi dari *layer*, `inputs` yang merupakan *value* input yang ada pada seluruh neuron dalam suatu *layer*, `net_value` yang merupakan array berisi net value dari seluruh node pada suatu *layer*, dan `outputs` yang merupakan array berisi hasil fungsi aktivasi dari net value seluruh node pada sebuah layer.

Class `Graph` merupakan representasi dari kumpulan-kumpulan *layer* yang membentuk FFNN. *Class* ini mempunyai atribut yaitu `layers` yang merupakan *list* berisi layer-layer yang ada di FFNN, `n_layer` yang merupakan jumlah layer pada FFNN, dan `output_activation` yang merupakan fungsi aktivasi layer output dari FFNN . Pada *class* ini, dibuat beberapa *method* terkait dengan FFNN seperti *method* untuk menghitung *output* dari suatu *layer*, menghitung *predicted value* dari suatu *input*, menghitung derivasi pada suatu layer dengan menggunakan *chain rule*, melakukan *backpropagation*, dan untuk melakukan training dari FFNN.

Terdapat fungsi `backpropagate` pada kelas `Graph` untuk melakukan *backpropagation* pada model *Neural Network* yang sudah dibuat. *Backpropagation* dilakukan dengan tujuan untuk meng-*update* nilai bobot yang ada pada tiap *node* di seluruh layer agar prediksi model semakin akurat. Oleh karena itu, penggunaan fungsi `backpropagate` harus dikombinasikan dengan fungsi *mini-batch gradient descent* agar kondisi berhentinya *backpropagation* dapat terdefinisikan dengan jelas. Fungsi *mini-batch gradient descent* pada program ini diimplementasikan pada fungsi `train` di kelas `Graph`. Dengan menggunakan *mini-batch gradient descent* maka, *backpropagation* akan dilakukan terus di tiap iterasi hingga *error* sudah lebih kecil dari *error threshold* atau ketika iterasi maksimum sudah tercapai.

Berikut adalah penjelasan parameter yang terdapat dalam implementasi *neural network* dan *backpropagation* kami.

No.	Parameter	Objek/Fungsi	Penjelasan
1.	prev_node	Layer	Jumlah node pada layer sebelumnya, digunakan untuk menginisialisasi matriks bobot.
2.	num_nodes	Layer	Jumlah node pada suatu layer
3.	activation_func	Layer	Fungsi aktivasi yang digunakan pada suatu layer
4.	input_count	Graph	Banyaknya input yang dimasukkan ke dalam model neural network.
5.	n_layers	Graph	Banyaknya layer pada suatu model neural network.
6.	n_neurons	Graph	Array berisi banyaknya neuron di tiap layer
7.	activation_funcs	Graph	Array berisi fungsi aktivasi untuk tiap layer
8.	x_train	Graph.train	Input pada model neural network
9.	y_train	Graph.train	Target pada model neural network
10.	lr	Graph.train	Learning rate
11.	err_thresh	Graph.train	Error threshold
12.	batch_size	Graph.train	Ukuran batch
13.	max_iter	Graph.train	Jumlah iterasi maksimum dari mini-batch gradient descent
14.	print_per_iter	Graph.train	Print error per berapa iterasi, agar fungsi menjadi verbose.

2. Hasil Eksekusi

Untuk melakukan *predict* output dari suatu input, maka pertama-tama model neural network harus diinisialisasi terlebih dahulu dengan memanggil konstruktor `Graph` dengan parameter yang sesuai. Pada implementasi tugas B ini, nilai bobot untuk tiap node pada suatu layer akan diinisialisasi secara acak dengan nilai yang kecil. Kemudian, setelah graf diinisialisasi, maka dilakukan *training* dari model dengan method `train` dengan parameter yang sesuai. Setelah *training* model, dapat dilakukan prediksi output dari suatu nilai dengan dipanggil method *predict* dengan parameter `x_train`.

Tahap	Tangkapan Layar

Create FFNN dan Train	`grafSigmoid = Graph(len(iris.feature_names), 2, [3, len(iris.target_names)], ['sigmoid', 'softmax'])`
	`#x, y, learning rate, error threshold, batch size, max iter?, print per iter?`
	`grafSigmoid.train(x, y.reshape(-1,1), 1e-2, 2e-2,50)`
	
	`Iterasi 1000: error 0.3753010`
	`Iterasi 2000: error 0.3476454`
	`Iterasi 3000: error 0.2847098`
	`Iterasi 4000: error 0.2433883`
	`Iterasi 5000: error 0.2596169`
	`Iterasi 6000: error 0.2408851`
	`Iterasi 7000: error 0.2457306`
	`Iterasi 8000: error 0.2254196`
	`Iterasi 9000: error 0.2360975`
	`Iterasi 10000: error 0.2140632`
	`Iterasi maksimum`
	`Berakhir pada iterasi: 10000`
	`Nilai error final: 0.21406323757416684`

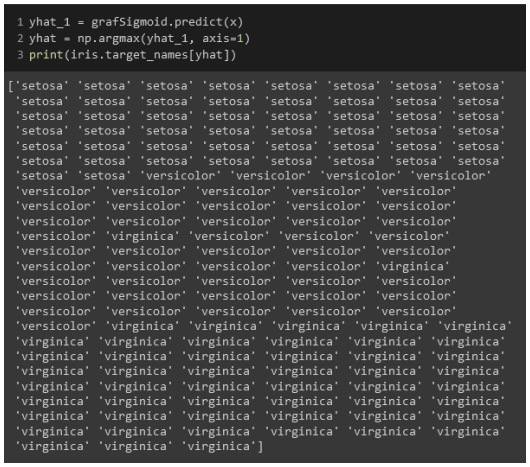
Predict

```
1 yhat_1 = grafSigmoid.predict(x)
2 yhat = np.argmax(yhat_1, axis=1)
3 print(iris.target_names[yhat])
```

```
['setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virginica' 'versicolor' 'virginica' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica']
```

3. Perbandingan Hasil Dengan MLP *sklearn*

Untuk program kami, kami memakai fungsi aktivasi *softmax* untuk layer output. Dari MLP `sklearn` sendiri tidak memiliki opsi untuk mengatur fungsi aktivasi layer output, hanya *hidden layer* saja. Oleh karena itu dilakukan perbandingan dengan fungsi aktivasi pada *hidden layer* yang sama dan *learning rate* yang sama juga. Berikut adalah perbandingan hasil program dengan hasil MLP `sklearn`.

No.	Fungsi Aktivasi Hidden Layer	Hasil Program	Hasil MLP sklearn
1.	Sigmoid (logistic di MLPClassifier)		

2. ReLU (relu di MLPClassifier)

```
1 yhat_1 = grafRelu.predict(x)
2 yhat = np.argmax(yhat_1, axis=1)
3 print(iris.target_names[yhat])
```

```
['setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virginica' 'versicolor' 'virginica' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'versicolor' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica']
```

```
iris.target_names[iris_classifier_sklearn.predict(x)]
✓ 0.3s
```

```
array(['setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'virginica', 'versicolor', 'virginica', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'virginica', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'versicolor', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica'], dtype='<U10')
```

3. Linear (identity di MLPClassifier)

```
1 yhat_1 = grafLinear.predict(x)
2 yhat = np.argmax(yhat_1, axis=1)
3 print(iris.target_names[yhat])
```

```
['setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virginica' 'versicolor' 'virginica' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
 'virginica' 'virginica' 'virginica']
```

```
iris.target_names[iris_classifier_sklearn.predict(x)]
✓ 0.3s
```

```
array(['setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'virginica', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'versicolor', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica'], dtype='<U10')
```

Dapat dilihat bahwa hasil program kami kurang lebih sama dengan hasil dari sklearn. Hasil ini membuktikan bahwa program kami dapat menghasilkan hasil prediksi yang kurang lebih sama dengan MLPClassifier sklearn dengan fungsi aktivasi layer dan learning rate yang sama.

4. Pembagian Tugas

No.	NIM	Nama	Kontribusi
1	13519096	Girvin Junod	Fungsi aktivasi turunan, fungsi backpropagation, fungsi train, implementasi program
2	13519116	Jeane Mikha Erwansyah	Fungsi aktivasi turunan, fungsi backpropagation, fungsi train, implementasi program
3	13519131	Hera Shafira	Fungsi aktivasi turunan, fungsi backpropagation, fungsi train, implementasi program
4	13519188	Jeremia Axel	Fungsi aktivasi turunan, fungsi backpropagation, fungsi train, implementasi program