TugasBesarII_13515058_13515086_13515118

November 15, 2018

0.0.1 IF 4071 Pembelajaran Mesin

Tugas Besar 2 Feed Forward Neural Network Disusun oleh:

13515058 Afif Bambang Prasetia (K-01)
13515086 Muhammad Iqbal Al Khowarizmi (K-01)
13515118 Gianfranco Fertino Hwandiano (K-01)

0.0.2 Source Code 1.a dan Penjelasannya.

In [1]:
```
import numpy as np

class FeedForwardNN():
    """
    Implementasi algoritma backpropagation sederhana. Kelas ini didesain
    dengan antarmuka mirip dengan antarmuka kelas MLPRegressor dari sklearn.
    Kelas ini mengimplementasi mini batch gradient descent dengan ukuran
    mini batch yang bisa diatur.
    """

    def __init__(self, hidden_layer_sizes=(5,),
                 batch_size=1, learning_rate=0.1,
                 max_iter=100, momentum=0, tol=0.0001):
        """
        Parameter:
            hidden_layer_sizes - Jumlah neuron pada hidden setiap hidden layer.
                                 Panjang tuple menunjukkan jumlah layer dan elemen
                                 ke-i tuple menunjukkan jumlah neuron pada layer
                                 ke-i.

            batch_size         - Jumlah data pada setiap mini batch.

            learning_rate      - Parameter learning rate untuk update weight.

            max_iter           - Jumlah epoch maksimal yang akan dilakukan.

            momentum           - Parameter momentum untuk update weight.
        """
        assert(len(hidden_layer_sizes) <= 10), "Max layer size is 10."
```

```
        self.layer_sizes = hidden_layer_sizes
        self.batch_size = batch_size
        self.learning_rate = learning_rate
        self.max_iter = max_iter
        self.momentum = momentum
        self.tol = tol
        self.weights = []
        self.delta_weights = []
        self.epoch = 0
        self.activate = lambda x: 1 / (1 + np.exp(-x))
        self.o_err = lambda o, t: o*(1-o)*(t-o)

    def __initiate_network(self, n_input=4):
        """
        Inisialisasi fully connected network dengan n_input input neuron.
        Inisialisasi weight dan bias.

        Parameter:
            n_input - Jumlah neuron pada input layer.
        """
        network_without_bias = [n_input, *self.layer_sizes]
        self.neurons_at_layers = [n+1 for n in network_without_bias] + [1] # Add bias
        for i in range(len(self.neurons_at_layers) - 1):
            shape = (self.neurons_at_layers[i+1], self.neurons_at_layers[i])
            self.weights.append(np.random.random(shape) - 0.5)
            self.delta_weights.append(np.zeros(shape))
        self.prev_error = 0
        self.convergent = False

    def score(self, X, y):
        """
        Mengembalikan error kumulatif dari prediksi pada data X terhadap label y.

        Parameter:
            X - Data untuk diprediksi.

            y - Label untuk perbandingan dengan prediksi yang dihasilkan.
        """
        cummulative_error = 0
        for data, label in zip(X, y):
            output = self.__feed_forward(data)
            cummulative_error += self.o_err(output, label)
        return cummulative_error

    def predict(self, x, discrete=False):
        """
        Mengembalikan nilai prediksi data x. Data x akan dimasukkan ke dalam
        neural network dan menghasilkan nilai prediksi.
```

```
        Parameter:
            x         - Data yang akan diprediksi.

            discrete - Jika bernilai true, nilai prediksi akan dibulatkan.
        """
        data = np.array(x)
        predictions = []
        for x in data:
            predictions.append(self.__feed_forward(x))
        if discrete:
            return np.around(predictions)
        else:
            return predictions

    def fit(self, X, y):
        """
        Metode untuk melakukan pembelajaran terhadap data X dan label y.

        Parameter:
            X - Data pembelajaran.

            y - Label pembelajaran.
        """
        self.__initiate_network(X.shape[1])
        self.batch_size = min(X.shape[0], self.batch_size)
        iterator = 0
        while self.epoch < self.max_iter and not self.convergent:
            iterator = 0
            while iterator < X.shape[0]:
                batch_X = X[iterator:iterator + self.batch_size]
                batch_y = y[iterator:iterator + self.batch_size]
                error = self.__feed_batch(batch_X, batch_y)
                delta_error = abs(self.prev_error - error)
                if delta_error <= self.tol:
                    self.convergent = True
                else:
                    self.prev_error = error
                iterator += self.batch_size
            self.epoch += 1


    def __feed_batch(self, batch_X, batch_y):
        """
        Metode untuk memasukkan mini batch ke dalam network, melakukan propagasi
        pada error, dan melakukan update weight. Mengembalikan batch error.

        Parameter:
```

```
        batch_x - Data dalam mini batch.

        batch_y - Label untuk data dalam mini batch.
    """
    error = 0
    for x, y in zip(batch_X, batch_y):
        output = self.__feed_forward(x)
        error += self.o_err(output, y)
    self.__backpropagate(error)
    self.__update_weights()
    return error


def __feed_forward(self, x):
    """
    Metode untuk memasukkan data ke dalam network. Mengembalikan
    nilai output.

    Parameter:
        x - Data yang akan dimasukkan ke dalam network.
    """
    inputs = np.append(x, 1) # Add bias
    self.outputs = [inputs]
    for layer in self.weights:
        inputs = self.activate(np.dot(layer, inputs))
        self.outputs.append(inputs)
    return inputs[0]


def __backpropagate(self, error):
    """
    Metode untuk mempropagasikan error dari output layer hingga hidden layer.

    Parameter:
        error - Error untuk dipropagasikan ke hidden layer.
    """
    propagation = np.array([error])
    self.errors = [propagation]
    for weight, output in zip(reversed(self.weights[1:]), reversed(self.outputs[1:-
        propagation = output*(1-output) * np.dot(weight.transpose(), propagation)
        self.errors.append(propagation)
    self.errors = list(reversed(self.errors))


def __update_weights(self):
    """
    Metode untuk melakukan update weight.
    """
```

```
                for i in range(len(self.weights)):
                    shape = (len(self.errors[i]), 1)
                    weight_changes = self.learning_rate * (self.errors[i].reshape(shape) * sel
                    self.delta_weights[i] = weight_changes + (self.momentum * self.delta_weight
                    self.weights[i] = self.weights[i] + self.delta_weights[i]
```

0.0.3 Source Code 1.b dan Penjelasannya.

```
In [2]: import numpy as np
        import pandas as pd
        from sklearn import preprocessing
        from sklearn.model_selection import train_test_split
        from keras.models import Sequential
        from keras.layers import Dense
```

```
Using TensorFlow backend.
```

```
In [3]: class MultiColumnLabelEncoder:
            '''
            Kelas ini berguna untuk melakukan label encoding pada setiap kolom dari
            data tabel 2 dimensi. Labelencoding akan mengubah data bernilai string
            menjadi angka secara acak (0,1,2, dst...)
            '''
            def __init__(self,columns = None):
                self.columns = columns # array berisi nama kolom yang ingin diencode
                self.le = preprocessing.LabelEncoder()

            def fit_transform(self,X):
                '''
                Mengencode dataframe X sesuai dengan kolom yang telah ditetapkan.
                '''
                output = X.copy()
                for col in self.columns:
                    output[col] = self.le.fit_transform(output[col])
                return output
```

```
In [4]: td = pd.read_csv("tennis.csv")
        print ("===== Dataset =====\n")
        print(td)
```

```
===== Dataset =====

     outlook temperature humidity   wind playtennis
0      sunny         hot     high  False         no
1      sunny         hot     high   True         no
2   overcast         hot     high  False        yes
3       rain        mild     high  False        yes
4       rain        cool   normal  False        yes
```

```
5       rain        cool   normal   True         no
6   overcast        cool   normal   True        yes
7      sunny        mild     high  False         no
8      sunny        cool   normal  False        yes
9       rain        mild   normal  False        yes
10     sunny        mild   normal   True        yes
11  overcast        mild     high   True        yes
12  overcast         hot   normal  False        yes
13      rain        mild     high   True         no
```

```
In [23]: # encode dataset tenis dengan label encoder
         td = MultiColumnLabelEncoder(columns = ['outlook','temperature','humidity','wind', 'pl
         # split dataset secara acak jadi data training dan data tes dengan holdout split 10%
         td_train, td_test = train_test_split(td, test_size=0.1)
         td_train_labels = td_train['playtennis'].values
         td_train_data = td_train.drop(columns=['playtennis']).values
         td_test_labels = td_test['playtennis'].values
         td_test_data = td_test.drop(columns=['playtennis']).values
```

```
In [37]: # Implementasi keras dengan menggunakan model sequential dan layer dense
         model = Sequential()
         model.add(Dense(2, input_dim=4, activation='sigmoid'))
         model.add(Dense(3, activation='sigmoid'))
         model.add(Dense(1, activation='sigmoid'))
         # Compile model
         # loss menggunakan binary_crossentropy karena loss function ini cocok diterapkan
         #   pada dataset dengan label biner, 0 dan 1.
         # Optimizer menggunakan stochastic gradient descent SGD
         model.compile(loss='binary_crossentropy', optimizer='SGD', metrics=['accuracy'])
         # Fit model dengan jumlah epoch 150 dan batch size 1 atau stochastic
         model.fit(td_train_data, td_train_labels, epochs=150, batch_size=1)
```

```
Epoch 1/150
12/12 [==============================] - 1s 45ms/step - loss: 0.7026 - acc: 0.2500
Epoch 2/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6958 - acc: 0.5833
Epoch 3/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6898 - acc: 0.6667
Epoch 4/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6845 - acc: 0.6667
Epoch 5/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6799 - acc: 0.6667
Epoch 6/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6758 - acc: 0.6667
Epoch 7/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6722 - acc: 0.6667
Epoch 8/150
```

```
12/12 [==============================] - 0s 4ms/step - loss: 0.6690 - acc: 0.6667
Epoch 9/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6663 - acc: 0.6667
Epoch 10/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6638 - acc: 0.6667
Epoch 11/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6614 - acc: 0.6667
Epoch 12/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6595 - acc: 0.6667
Epoch 13/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6578 - acc: 0.6667
Epoch 14/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6563 - acc: 0.6667
Epoch 15/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6549 - acc: 0.6667
Epoch 16/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6536 - acc: 0.6667
Epoch 17/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6526 - acc: 0.6667
Epoch 18/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6516 - acc: 0.6667
Epoch 19/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6508 - acc: 0.6667
Epoch 20/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6500 - acc: 0.6667
Epoch 21/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6493 - acc: 0.6667
Epoch 22/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6486 - acc: 0.6667
Epoch 23/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6481 - acc: 0.6667
Epoch 24/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6476 - acc: 0.6667
Epoch 25/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6472 - acc: 0.6667
Epoch 26/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6467 - acc: 0.6667
Epoch 27/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6464 - acc: 0.6667
Epoch 28/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6460 - acc: 0.6667
Epoch 29/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6457 - acc: 0.6667
Epoch 30/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6454 - acc: 0.6667
Epoch 31/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6451 - acc: 0.6667
Epoch 32/150
```

```
12/12 [==============================] - 0s 4ms/step - loss: 0.6448 - acc: 0.6667
Epoch 33/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6446 - acc: 0.6667
Epoch 34/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6444 - acc: 0.6667
Epoch 35/150
12/12 [==============================] - 0s 5ms/step - loss: 0.6441 - acc: 0.6667
Epoch 36/150
12/12 [==============================] - 0s 7ms/step - loss: 0.6439 - acc: 0.6667
Epoch 37/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6438 - acc: 0.6667
Epoch 38/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6436 - acc: 0.6667
Epoch 39/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6434 - acc: 0.6667
Epoch 40/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6432 - acc: 0.6667
Epoch 41/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6430 - acc: 0.6667
Epoch 42/150
12/12 [==============================] - 0s 5ms/step - loss: 0.6429 - acc: 0.6667
Epoch 43/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6427 - acc: 0.6667
Epoch 44/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6426 - acc: 0.6667
Epoch 45/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6424 - acc: 0.6667
Epoch 46/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6423 - acc: 0.6667
Epoch 47/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6422 - acc: 0.6667
Epoch 48/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6420 - acc: 0.6667
Epoch 49/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6419 - acc: 0.6667
Epoch 50/150
12/12 [==============================] - 0s 5ms/step - loss: 0.6417 - acc: 0.6667
Epoch 51/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6416 - acc: 0.6667
Epoch 52/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6415 - acc: 0.6667
Epoch 53/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6414 - acc: 0.6667
Epoch 54/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6412 - acc: 0.6667
Epoch 55/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6411 - acc: 0.6667
Epoch 56/150
```

```
12/12 [==============================] - ETA: 0s - loss: 0.4135 - acc: 1.000 - 0s 2ms/step - lo
Epoch 57/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6408 - acc: 0.6667
Epoch 58/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6407 - acc: 0.6667
Epoch 59/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6406 - acc: 0.6667
Epoch 60/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6405 - acc: 0.6667
Epoch 61/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6403 - acc: 0.6667
Epoch 62/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6402 - acc: 0.6667
Epoch 63/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6401 - acc: 0.6667
Epoch 64/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6400 - acc: 0.6667
Epoch 65/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6399 - acc: 0.6667
Epoch 66/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6398 - acc: 0.6667
Epoch 67/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6397 - acc: 0.6667
Epoch 68/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6396 - acc: 0.6667
Epoch 69/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6394 - acc: 0.6667
Epoch 70/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6393 - acc: 0.6667
Epoch 71/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6392 - acc: 0.6667
Epoch 72/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6391 - acc: 0.6667
Epoch 73/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6390 - acc: 0.6667
Epoch 74/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6389 - acc: 0.6667
Epoch 75/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6387 - acc: 0.6667
Epoch 76/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6386 - acc: 0.6667
Epoch 77/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6385 - acc: 0.6667
Epoch 78/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6384 - acc: 0.6667
Epoch 79/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6383 - acc: 0.6667
Epoch 80/150
```

```
12/12 [==============================] - 0s 2ms/step - loss: 0.6382 - acc: 0.6667
Epoch 81/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6381 - acc: 0.6667
Epoch 82/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6379 - acc: 0.6667
Epoch 83/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6379 - acc: 0.6667
Epoch 84/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6378 - acc: 0.6667
Epoch 85/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6376 - acc: 0.6667
Epoch 86/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6375 - acc: 0.6667
Epoch 87/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6373 - acc: 0.6667
Epoch 88/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6373 - acc: 0.6667
Epoch 89/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6372 - acc: 0.6667
Epoch 90/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6371 - acc: 0.6667
Epoch 91/150
12/12 [==============================] - 0s 6ms/step - loss: 0.6370 - acc: 0.6667
Epoch 92/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6369 - acc: 0.6667
Epoch 93/150
12/12 [==============================] - 0s 6ms/step - loss: 0.6367 - acc: 0.6667
Epoch 94/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6366 - acc: 0.6667
Epoch 95/150
12/12 [==============================] - 0s 6ms/step - loss: 0.6365 - acc: 0.6667
Epoch 96/150
12/12 [==============================] - 0s 6ms/step - loss: 0.6364 - acc: 0.6667
Epoch 97/150
12/12 [==============================] - 0s 5ms/step - loss: 0.6363 - acc: 0.6667
Epoch 98/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6362 - acc: 0.6667
Epoch 99/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6361 - acc: 0.6667
Epoch 100/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6360 - acc: 0.6667
Epoch 101/150
12/12 [==============================] - 0s 5ms/step - loss: 0.6358 - acc: 0.6667
Epoch 102/150
12/12 [==============================] - 0s 5ms/step - loss: 0.6357 - acc: 0.6667
Epoch 103/150
12/12 [==============================] - 0s 6ms/step - loss: 0.6357 - acc: 0.6667A: 0s - loss:
Epoch 104/150
```

```
12/12 [==============================] - 0s 4ms/step - loss: 0.6355 - acc: 0.6667
Epoch 105/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6354 - acc: 0.6667
Epoch 106/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6353 - acc: 0.6667
Epoch 107/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6352 - acc: 0.6667
Epoch 108/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6351 - acc: 0.6667
Epoch 109/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6350 - acc: 0.6667
Epoch 110/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6349 - acc: 0.6667
Epoch 111/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6348 - acc: 0.6667
Epoch 112/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6347 - acc: 0.6667
Epoch 113/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6346 - acc: 0.6667
Epoch 114/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6344 - acc: 0.6667
Epoch 115/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6343 - acc: 0.6667
Epoch 116/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6342 - acc: 0.6667
Epoch 117/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6341 - acc: 0.6667
Epoch 118/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6340 - acc: 0.6667
Epoch 119/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6339 - acc: 0.6667
Epoch 120/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6338 - acc: 0.6667
Epoch 121/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6337 - acc: 0.6667
Epoch 122/150
12/12 [==============================] - 0s 5ms/step - loss: 0.6336 - acc: 0.6667
Epoch 123/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6335 - acc: 0.6667
Epoch 124/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6334 - acc: 0.6667
Epoch 125/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6333 - acc: 0.6667
Epoch 126/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6332 - acc: 0.6667
Epoch 127/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6330 - acc: 0.6667
Epoch 128/150
```

```
12/12 [==============================] - 0s 2ms/step - loss: 0.6329 - acc: 0.6667
Epoch 129/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6328 - acc: 0.6667
Epoch 130/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6327 - acc: 0.6667
Epoch 131/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6326 - acc: 0.6667
Epoch 132/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6325 - acc: 0.6667
Epoch 133/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6324 - acc: 0.6667
Epoch 134/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6323 - acc: 0.6667
Epoch 135/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6322 - acc: 0.6667
Epoch 136/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6321 - acc: 0.6667
Epoch 137/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6320 - acc: 0.6667
Epoch 138/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6319 - acc: 0.6667
Epoch 139/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6318 - acc: 0.6667
Epoch 140/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6316 - acc: 0.6667
Epoch 141/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6315 - acc: 0.6667
Epoch 142/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6314 - acc: 0.6667
Epoch 143/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6313 - acc: 0.6667
Epoch 144/150
12/12 [==============================] - 0s 4ms/step - loss: 0.6312 - acc: 0.6667
Epoch 145/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6311 - acc: 0.6667
Epoch 146/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6310 - acc: 0.6667
Epoch 147/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6309 - acc: 0.6667
Epoch 148/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6308 - acc: 0.6667
Epoch 149/150
12/12 [==============================] - 0s 2ms/step - loss: 0.6307 - acc: 0.6667
Epoch 150/150
12/12 [==============================] - 0s 3ms/step - loss: 0.6306 - acc: 0.6667
```

```
Out[37]: <keras.callbacks.History at 0x10a4f668>
```

```
In [33]: # Implementasi feed forward NN dengan backpropagation
         # hiddenlayer 2 dengan jumlah neuron 12 dan 8
         # batch size 1 atau stochastic karena data sedikit.
         ffnn = FeedForwardNN((2,3), batch_size=1, max_iter=10000, momentum=0.5, learning_rate=
         ffnn.fit(td_train_data, td_train_labels)
```

0.0.4 Hasil Esksekusi

```
In [36]: # Evaluasi model Keras Feedforward NN terhadap data test
         scores = model.evaluate(td_test_data, td_test_labels)
         print("\n%s: %.2f%%\n\n" % (model.metrics_names[1], scores[1]*100))
```

```
2/2 [==============================] - 0s 62ms/step

acc: 50.00%
```

```
In [34]: # Evaluasi model self-implemented Feedforward NN terhadap data test
         # hitung akurasi dengan membandingkan hasil prediksi dan label sesungguhnya.
         acc = 0
         correct = 0
         n = 0
         for a, b in zip(ffnn.predict(td_test_data, discrete=True), td_test_labels):
             print("Prediction", a, "Truth", b)
             if (a==b):
                 correct+=1
             n+=1
         print("acc:", correct/n * 100, "%")
```

```
Prediction 1.0 Truth 1
Prediction 1.0 Truth 0
acc: 50.0 %
```

0.0.5 Analisis Perbandingan hasil classifier A dan B

Classifier A adalah hasil implementasi sendiri dari feed-forward neural network dan classifier B adalah implementasi library Keras dari feed-forward neural network.

Susunan neuron kedua implementasi dibuat sama, yaitu 1 input dengan 4 neuron, 2 hidden layer dengan 2 neuron untuk hidden layer pertama dan 3 neuron untuk hidden layer kedua, dan 1 output dengan 1 neuron. Susunan neuron tersebut didapat berdasarkan eksperimen mandiri dan referensi dari https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw

Keduanya menghasilkan akurasi yang sama terhadap data test, yaitu sebesar 50%. Dengan holdout split 10%, dari 14 dataset, data terbagi menjadi 12 train data dan 2 test data. Akurasi 50% artinya netural network berhasil menebak dengan benar salah satu data test.

Berdasarkan hasil tersebut, dapat disimpulkan bahwa feed-forward neural network hasil dari implementasi sendiri dan library Keras dapat dibilang serupa.

0.0.6 Pembagian Tugas

13515058 Afif Bambang Prasetia (K-01): Menyusun laporan
13515086 Muhammad Iqbal Al Khowarizmi (K-01): Implementasi 1a
13515118 Gianfranco Fertino Hwandiano (K-01): Implementasi 1b