Francesco Michael Kusuma

13522038

Artificial Neural Network

Cara Kerja:

1. Pilih bobot dan bias secara random terlebih dahulu.
2. Hitung *output* dari setiap neuron di setiap lapisan.
3. Hitung *loss* berdasarkan output dan target sebenarnya.
4. Hitung *gradien* terhadap setiap bobot secara *backward propagation*.
5. Update bobot menggunakan *optimizer* seperti *Gradient Descent atau Adam*.
6. Ulangi langkah 2-5 hingga konvergen atau jumlah iterasi terpenuhi.

Perbandingan:

- *Scratch*

[49]:
```
from ann import ArtificialNeuralNetwork

layers = [X_train.shape[1], 5, 3, 1]  # Input layer, 2 hidden layers, output layer
activations = ['relu', 'relu', 'sigmoid']  # Activation functions

nn_scratch = ArtificialNeuralNetwork(layers=layers,
                  activations=activations,
                  batch_size=32,
                  loss_function='binary_crossentropy',
                  learning_rate=0.01,
                  epochs=1000,
                  regularization='l2',
                  optimizer='adam')

validate_model_ann('Artificial Neural Network', 'scratch', nn_scratch)
```

```
Hold-Out Validation:
F1 Score Artificial Neural Network from scratch with Hold-Out Validation:  0.5384615384615384
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        72
           1       0.37      1.00      0.54        42

    accuracy                           0.37       114
   macro avg       0.18      0.50      0.27       114
weighted avg       0.14      0.37      0.20       114

Epoch 400/1000, Loss: 1.0126
Epoch 500/1000, Loss: 1.1142
Epoch 600/1000, Loss: 0.9413
Epoch 700/1000, Loss: 0.9867
Epoch 800/1000, Loss: 0.9503
Epoch 900/1000, Loss: 0.9658
F1 Score Artificial Neural Network from scratch with K-Fold Cross-Validation:  [0.8505747126436781, 0.8461538461538461, 0.7076923076923077, 0.9024390243902439, 0.8333333333333334]
Average F1 Score: 0.8280386448426817
```

- *Library*

```
[50]: import tensorflow as tf
      from tensorflow import keras
      from tensorflow.keras import layers

      model = keras.Sequential([
          layers.Dense(128, input_dim=X_train.shape[1], activation='relu'),
          layers.Dense(64, activation='relu'),
          layers.Dense(1, activation='sigmoid')
      ])

[51]: model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

[52]: validate_model_ann('Artificial Neural Network', 'library', model)

      Hold-Out Validation:
      4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
      F1 Score Artificial Neural Network from library with Hold-Out Validation:  0.5384615384615384
                    precision    recall  f1-score   support

                 0       0.00      0.00      0.00        72
                 1       0.37      1.00      0.54        42

          accuracy                           0.37       114
         macro avg       0.18      0.50      0.27       114
      weighted avg       0.14      0.37      0.20       114

      K-Fold Cross-Validation:
      15/15 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.5059 - loss: 17.4845
      4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
      15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6113 - loss: 3.1578
      4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
      15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8446 - loss: 1.3480
      4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
      15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8659 - loss: 0.6129
      4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
      15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8385 - loss: 0.9499
      4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
      F1 Score Artificial Neural Network from library with K-Fold Cross-Validation:  [0.8247422680412371, 0.7843137254901961, 0.8108108108108109, 0.93827160493
      82716, 0.9382716049382716]
      Average F1 Score: 0.8592820028437573
```

Dari implementasi secara *scratch* dan *library,* terlihat bahwa penggunaan *library* memiliki nilai F1 score yang lebih tinggi.

Improvement:

Improvement yang dapat dilakukan pada algoritma *Artificial Neural Network* secara *scratch* dapat dilakukan dengan memperbaiki *learning rate, regularization, batch size, optimizer,* dan jenis *activation function* yang digunakan.