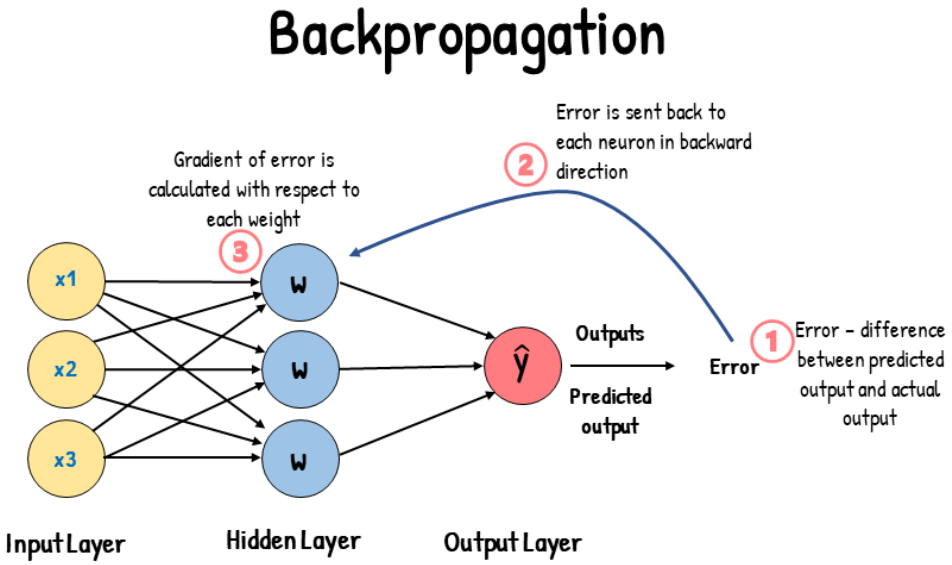
Nama : Nadhifi Qurrunul B F H

Nim : 1103204156

Catatan Materi: Backpropagation

Backpropagation adalah metode pelatihan jaringan saraf tiruan yang memanfaatkan konsep propagasi gradien untuk mengoptimalkan parameter-model. Algoritma ini bekerja dengan menganalisis kesalahan (error) antara output yang dihasilkan oleh jaringan dan target yang seharusnya. Gradien dari kesalahan ini kemudian dikembalikan melalui jaringan, dan parameter-model diperbarui berdasarkan gradien tersebut.



Prinsip kerja backpropagation dapat dijelaskan sebagai berikut:

1. Pertama, jaringan saraf dilalui dengan data pelatihan. Proses ini disebut sebagai forward pass.
2. Setelah forward pass, kesalahan jaringan dihitung. Kesalahan ini merupakan ukuran seberapa jauh output jaringan dari target yang diinginkan.
3. Kesalahan tersebut kemudian dipropagasi mundur melalui jaringan. Proses ini disebut sebagai backward pass.
4. Dalam backward pass, gradien kesalahan dihitung untuk setiap bobot dan bias jaringan.
5. Gradien tersebut kemudian digunakan untuk menyesuaikan bobot dan bias jaringan.

Aturan Rantai

Backpropagation menggunakan aturan rantai untuk menghitung gradien kesalahan. Aturan rantai adalah metode yang digunakan untuk menghitung turunan dari fungsi yang kompleks.

Secara umum, aturan rantai dapat ditulis sebagai berikut:

$\partial f(g(x))/\partial x = \partial f(g(x))/\partial g(x) * \partial g(x)/\partial x$

Dimana:

- f adalah fungsi yang ingin dihitung turunannya
- g adalah fungsi yang digunakan untuk mengkomposisi f
- x adalah argumen dari f

Dalam konteks backpropagation, fungsi f adalah fungsi kesalahan jaringan, dan fungsi g adalah fungsi aktivasi jaringan.

Learning rate adalah konstanta yang menentukan seberapa cepat bobot dan bias jaringan disesuaikan. Learning rate yang terlalu besar dapat menyebabkan jaringan tidak stabil, sedangkan learning rate yang terlalu kecil dapat menyebabkan jaringan membutuhkan waktu lama untuk belajar.

Langkah-langkah Backpropagation:

1. Inisialisasi Bobot dan Bias
    - Awalnya, bobot dan bias diatur secara acak atau menggunakan metode inisialisasi tertentu.
2. Feedforward:
    - Data input diteruskan melalui jaringan untuk menghasilkan output.
    - Setiap lapisan melakukan transformasi pada input menggunakan bobot dan biasnya.
3. Perhitungan Kesalahan (Error):
    - Kesalahan dihitung dengan membandingkan output yang dihasilkan dengan target yang seharusnya.
    - Fungsi kerugian (loss function) digunakan untuk mengukur seberapa besar kesalahan.
4. Backward Pass:
    - Gradien kesalahan terhadap bobot dan bias dihitung menggunakan aturan rantai.
    - Gradien mengukur seberapa besar kesalahan akan berubah jika bobot dan bias diubah sedikit.
5. Update Bobot dan Bias:
    - Bobot dan bias diperbarui berdasarkan gradien yang dihitung.
    - Metode optimasi seperti gradien turun (gradient descent) digunakan untuk memperbarui parameter.
6. Iterasi:
    - Langkah 2 hingga 5 diulang untuk setiap batch data atau seluruh dataset pelatihan.
    - Proses diulang hingga model mencapai tingkat kinerja yang dapat diterima atau sejumlah iterasi tertentu.

Implementasi:

```python
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_classification

# Membuat dataset sintetis untuk contoh
X, y = make_classification(n_samples=1000, n_features=10, n_classes=2, random_state=42)

# Memisahkan dataset menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standarisasi data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Membangun model jaringan saraf tiruan dengan TensorFlow
model = models.Sequential([
    layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

# Menentukan optimizer, fungsi kerugian, dan metrik
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Melatih model dengan backpropagation
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

# Evaluasi model pada data uji
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {test_accuracy}')
```

```
Epoch 1/10
25/25 [==============================] - 3s 30ms/step - loss: 0.5875 - accuracy: 0.7175 - val_loss: 0.5130 - val_accuracy: 0.8000
Epoch 2/10
25/25 [==============================] - 0s 19ms/step - loss: 0.4437 - accuracy: 0.8500 - val_loss: 0.4233 - val_accuracy: 0.8250
Epoch 3/10
25/25 [==============================] - 1s 24ms/step - loss: 0.3693 - accuracy: 0.8650 - val_loss: 0.3874 - val_accuracy: 0.8300
Epoch 4/10
25/25 [==============================] - 0s 19ms/step - loss: 0.3301 - accuracy: 0.8788 - val_loss: 0.3700 - val_accuracy: 0.8400
Epoch 5/10
25/25 [==============================] - 1s 25ms/step - loss: 0.3135 - accuracy: 0.8825 - val_loss: 0.3636 - val_accuracy: 0.8300
Epoch 6/10
25/25 [==============================] - 1s 23ms/step - loss: 0.3053 - accuracy: 0.8800 - val_loss: 0.3686 - val_accuracy: 0.8350
Epoch 7/10
25/25 [==============================] - 1s 20ms/step - loss: 0.2973 - accuracy: 0.8775 - val_loss: 0.3620 - val_accuracy: 0.8300
Epoch 8/10
25/25 [==============================] - 1s 26ms/step - loss: 0.2898 - accuracy: 0.8775 - val_loss: 0.3685 - val_accuracy: 0.8350
Epoch 9/10
25/25 [==============================] - 0s 13ms/step - loss: 0.2853 - accuracy: 0.8800 - val_loss: 0.3636 - val_accuracy: 0.8300
Epoch 10/10
25/25 [==============================] - 0s 15ms/step - loss: 0.2830 - accuracy: 0.8813 - val_loss: 0.3693 - val_accuracy: 0.8300
7/7 [==============================] - 0s 7ms/step - loss: 0.3693 - accuracy: 0.8300
Test Accuracy: 0.8299999833106995
```

Manfaat Backpropagation:

- Memungkinkan jaringan saraf untuk mengoptimalkan parameter-modelnya berdasarkan kesalahan yang dihasilkan.
- Mampu belajar representasi yang kompleks dari data input.
- Digunakan dalam berbagai bidang seperti pengenalan gambar, pengolahan bahasa alami, dan prediksi.

Kelebihan:

- Backpropagation adalah algoritma yang sederhana dan mudah diimplementasikan.

- Backpropagation dapat digunakan untuk melatih jaringan saraf dengan berbagai jenis fungsi aktivasi.

Kekurangan:

- Backpropagation dapat terjebak pada titik lokal minimum.
- Backpropagation dapat membutuhkan waktu lama untuk belajar, terutama untuk jaringan saraf yang besar.

Backpropagation digunakan dalam berbagai aplikasi machine learning, termasuk:

- Klasifikasi
- Regresi
- Pengenalan pola
- Pembobotan
- Pengolahan bahasa alami
- Computer vision