Experiment No. 4
Design a fully connected deep neural network with at least 2 hidden layers on Titanic survival classification by choosing appropriate Learning Algorithms
Date of Performance: 05/09/23
Date of Submission: 12/09/23

Aim: Design a fully connected deep neural network with at least 2 hidden layers on Titanic survival classification by choosing appropriate Learning Algorithms.

Objective: Ability to perform experimentations and deciding upon the best learning algorithm for a 3 layered neural network.

Theory:

In deep learning, we have the concept of loss, which tells us how poorly the model is performing at that current instant. Now we need to use this loss to train our network such that it performs better. Essentially what we need to do is to take the loss and try to minimize it, because a lower loss means our model is going to perform better. The process of minimizing (or maximizing) any mathematical expression is called optimization.

Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Optimizers are used to solve optimization problems by minimizing the function.

How do Optimizers work?

For a useful mental model, you can think of a hiker trying to get down a mountain with a blindfold on. It's impossible to know which direction to go in, but there's one thing she can know: if she's going down (making progress) or going up (losing progress). Eventually, if she keeps taking steps that lead her downwards, she'll reach the base.

Similarly, it's impossible to know what your model's weights should be right from the start. But with some trial and error based on the loss function (whether the hiker is descending), you can end up getting there eventually.

How you should change your weights or learning rates of your neural network to reduce the losses is defined by the optimizers you use. Optimization algorithms are responsible for reducing the losses and to provide the most accurate results possible.

Various optimizers are researched within the last few couples of years each having its advantages and disadvantages. Read the entire article to understand the working, advantages, and disadvantages of the algorithms.

Algorithms:

Gradient Descent

Stochastic Gradient Descent (SGD)

Mini Batch Stochastic Gradient Descent (MB-SGD)

SGD with momentum

Nesterov Accelerated Gradient (NAG)

Adaptive Gradient (AdaGrad)

AdaDelta

RMSprop

Adam

Code:

```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

data = pd.read_csv("titanic.csv")
```

```
data = data.drop(["Name", "Ticket", "Cabin", "Embarked", "PassengerId"], axis=1)
data = pd.get_dummies(data, columns=["Sex", "Pclass"], drop_first=True)
data["Age"].fillna(data["Age"].median(), inplace=True)
data["Fare"].fillna(data["Fare"].median(), inplace=True)

X = data.drop("Survived", axis=1)
y = data["Survived"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

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

model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2, verbose=2)

loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss:.4f}, Test Accuracy: {accuracy:.4f}")
```

Output:

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss:.4f}, Test Accuracy: {accuracy:.4f}")
```

```
3/3 [==============================] - 0s 9ms/step - loss: 0.0012 - accuracy: 1.0000
Test Loss: 0.0012, Test Accuracy: 1.0000
```

Conclusion:

Through these practical I get to know that, the code provided uses the Adam optimizer, an acronym for "Adaptive Moment Estimation." Consequently, the code employs the "Adam" optimization algorithm. Adam represents an advanced optimization technique that integrates aspects of both RMSprop and Momentum approaches. It dynamically adjusts the learning rates for each parameter by considering their historical gradients, rendering it highly effective for the training of deep neural networks.