بررسی کد زده شده در سوال سوم :

```
import numpy as np
import matplotlib.pyplot as plt

def true_function(x):
    """
    The function we want to approximate using a neural network.
    """
    return x**2


np.random.seed(42)
X_train = np.random.uniform(low=-3, high=3, size=(100, 1))
y_train = true_function(X_train)
```

`true_function(x)` یک تابع است که توسط یک شیب دوم (توان دوم) از ورودی x محاسبه می‌شود. به عبارت دیگر، تابع مورد نظر برابر با x به توان 2 است.

با استفاده از `np.random.seed(42)`، یک تعداد تصادفی را مشخص می‌کند تا اطمینان حاصل شود که نتایج قابل تکرار باشند.

`X_train` با استفاده از `np.random.uniform(low=-3, high=3, size=(100, 1))` به صورت تصادفی 100 نمونه از اعداد دارای توزیع یکنواخت بین -3 تا 3 تولید می‌کند. این نمونه‌ها به عنوان ورودی برای مدل مورد استفاده قرار می‌گیرند.

`y_train` برابر با مقادیر تابع `true_function` برای هر نمونه از `X_train` است. به عبارت دیگر، این بردار حاوی مقادیر متناظر با هر نمونه از `X_train` در تابع `true_function` می‌باشد.

```
input_size = 1
hidden_size = 10
output_size = 1
learning_rate = 0.01
epochs = 65000

    """
    Initialize weights and biases.

    """
weights_input_hidden = np.random.randn(input_size, hidden_size)
bias_input_hidden = np.zeros((1, hidden_size))
weights_hidden_output = np.random.randn(hidden_size, output_size)
bias_hidden_output = np.zeros((1, output_size))
```

این بخش از کد مربوط به تنظیمات مربوط به یک شبکه‌ی عصبی است. در اینجا، پارامترهایی مانند اندازه ورودی (`input_size`)، اندازه لایه مخفی (`hidden_size`)، اندازه خروجی (`output_size`)، نرخ یادگیری (`learning_rate`) و تعداد دوره‌های آموزش (`epochs`) تعیین شده‌اند.

سپس، وزن‌ها و بایاس‌ها به صورت تصادفی مقداردهی اولیه می‌شوند. به عبارت دیگر، ماتریس وزن بین لایه ورودی و لایه مخفی با ابعاد `input_size` × `hidden_size` از اعداد تصادفی با توزیع نرمال (میانگین صفر و واریانس یک) ساخته می‌شود. همچنین بایاس مربوط به لایه مخفی با ابعاد 1 × `hidden_size` با مقدار صفر مقداردهی می‌شود. همین روند برای وزن‌ها و بایاس‌های لایه مخفی به لایه خروجی نیز اعمال می‌شود.

```
 """
    Train the neural network and update weights and biases. At last epoch, print
the loss.
    """
for epoch in range(epochs):
    # Forward pass
    hidden_layer_input = np.dot(X_train, weights_input_hidden) +
bias_input_hidden
    hidden_layer_output = 1 / (1 + np.exp(-hidden_layer_input))
    predicted_output = np.dot(hidden_layer_output, weights_hidden_output) +
bias_hidden_output
```

در این بخش از کد، شبکه‌ی عصبی در حال آموزش است. این قسمت شامل یک حلقه (`for`) است که از تعداد دوره‌های آموزش (`epochs`) عبور می‌کند.

در هر دوره، مراحل زیر انجام می‌شوند:

1. **Forward Pass**:

- `hidden_layer_input` محاسبه می‌شود با ضرب داخلی بین داده‌های آموزش (`X_train`) و وزن‌های بین لایه ورودی و لایه مخفی (`weights_input_hidden`)، سپس بایاس مربوطه (`bias_input_hidden`) افزوده می‌شود.

- `hidden_layer_output` محاسبه می‌شود با اعمال تابع فعال‌سازی (sigmoid) بر روی `hidden_layer_input`.

- `predicted_output` محاسبه می‌شود با ضرب داخلی بین خروجی لایه مخفی (`hidden_layer_output`) و وزن‌های بین لایه مخفی و لایه خروجی (`weights_hidden_output`)، سپس بایاس مربوطه (`bias_hidden_output`) افزوده می‌شود.

```
 loss = np.mean((predicted_output - y_train)**2)

    # Backward pass
    output_error = predicted_output - y_train
    hidden_layer_error = np.dot(output_error, weights_hidden_output.T) *
hidden_layer_output * (1 - hidden_layer_output)

    # Update weights and biases
```

```
    weights_hidden_output -= learning_rate * np.dot(hidden_layer_output.T,
output_error)
    bias_hidden_output -= learning_rate * np.sum(output_error, axis=0,
keepdims=True)
    weights_input_hidden -= learning_rate * np.dot(X_train.T, hidden_layer_error)
    bias_input_hidden -= learning_rate * np.sum(hidden_layer_error, axis=0,
keepdims=True)
```

در این بخش از کد، مقدار خطا (Loss) برای هر دوره محاسبه می‌شود و سپس فرآیند پس‌انتشار انجام می‌شود تا وزن‌ها و بایاس‌ها به‌روزرسانی شوند.

1. **محاسبه خطا (Loss):**

   - `loss` مقدار خطا متوسط میان مقادیر پیش‌بینی شده (`predicted_output`) و مقادیر واقعی (`y_train`) با استفاده از مربعات اختلاف محاسبه می‌شود.

2. **پس‌انتشار (Backward Pass):**

   - `output_error` محاسبه می‌شود که تفاوت بین خروجی پیش‌بینی شده و مقدار واقعی است.

   - `hidden_layer_error` با استفاده از `output_error`، وزن‌های بین لایه مخفی و لایه خروجی (`weights_hidden_output`) و خروجی لایه مخفی محاسبه می‌شود. این مقدار به عنوان خطا مرتبط با لایه مخفی برای به‌روزرسانی وزن‌ها در لایه ورودی استفاده می‌شود.

3. **به‌روزرسانی وزن‌ها و بایاس‌ها (Update Weights and Biases):**

   - `weights_hidden_output` و `bias_hidden_output` با استفاده از گرادیان خطا نسبت به وزن‌ها و بایاس‌های لایه خروجی به‌روزرسانی می‌شوند.

   - همچنین، `weights_input_hidden` و `bias_input_hidden` با استفاده از گرادیان خطا نسبت به وزن‌ها و بایاس‌های لایه مخفی به‌روزرسانی می‌شوند.

```
# Test
X_test = np.linspace(-3, 3, 100).reshape(-1, 1)
hidden_layer_input = np.dot(X_test, weights_input_hidden) + bias_input_hidden
hidden_layer_output = 1 / (1 + np.exp(-hidden_layer_input))
predicted_output = np.dot(hidden_layer_output, weights_hidden_output) +
bias_hidden_output
```

1. `X_test` تولید می‌شود که یک دنباله از اعداد از -3 تا 3 است و برای تست شبکه استفاده می‌شود. این داده‌ها به صورت یک آرایه دو بعدی با ابعاد (100، 1) تعریف می‌شوند.

2. `hidden_layer_input` محاسبه می‌شود با ضرب داخلی بین داده‌های تست (`X_test`) و وزن‌های بین لایه ورودی و لایه مخفی (`weights_input_hidden`)، سپس بایاس مربوطه (`bias_input_hidden`) افزوده می‌شود.

3. `hidden_layer_output` محاسبه می‌شود با اعمال تابع فعال‌سازی (sigmoid) بر روی `hidden_layer_input`.

4. `predicted_output` محاسبه می‌شود با ضرب داخلی بین خروجی لایه مخفی (`hidden_layer_output`) و وزن‌های بین لایه مخفی و لایه خروجی (`weights_hidden_output`)، سپس بایاس مربوطه (`bias_hidden_output`) افزوده می‌شود.