LAB 03:
THỰC HÀNH VỚI PYTORCH (TIẾP THEO)

Ứng dụng Xử lý ảnh số và video số – 19TGMT

Giáo viên phụ trách:

- PhD. Lý Quốc Ngọc
- MS. Phạm Minh Hoàng
- MS. Nguyễn Mạnh Hùng

Sinh viên:

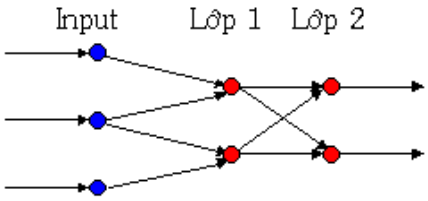
- Chung Kim Khánh (19127644)

MỤC LỤC

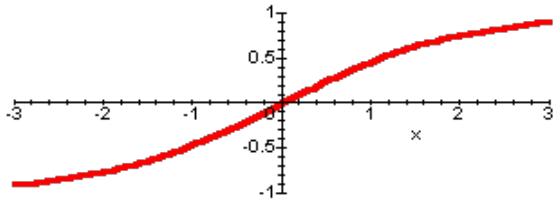
I. Giới thiệu về mạng nơ-ron Feed Forward (FF)

Mạng Perceptron không thể học được bài toán XOR, hạn chế ấy là do chính cấu trúc của mạng. Vì thế năm 1986 Rumelhart và McClelland đã cải tiến Perceptron thành mạng Perceptron nhiều lớp (MultiLayer Perceptron, MLP), hay còn gọi là mạng Feedforward. [2]

Mạng Feedforward là một mạng gồm một hay nhiều lớp nơron, trong đó các dây dẫn tín hiệu chỉ truyền theo một chiều từ input qua các lớp, cho đến output. Sau đây là một ví dụ gồm hai lớp, mỗi lớp có hai nơron: [2]



Lưu ý là lớp input không được coi là một lớp của mạng. Ngoài ra mạng Feedforward cũng khác Perceptron ở chỗ là nó không dùng hàm Heaviside làm transfer function nữa, thay vào đó là các hàm sigmoid (tansig hay logsig). Sau đây là dạng của hàm tansig (tanh(x/2)): [2]



II. Thực nghiệm

1. Thay đổi tốc độ học

- **Giá trị đầu vào và đầu ra mẫu cho training:**

x_train = torch.tensor([[[0, 0]], [[0, 1]], [[1, 0]], [[1, 1]]], dtype=torch.float)

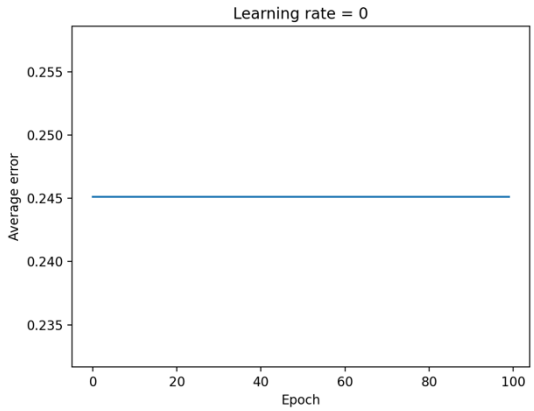
y_train = torch.tensor([[[0]], [[1]], [[1]], [[0]]], dtype=torch.float)

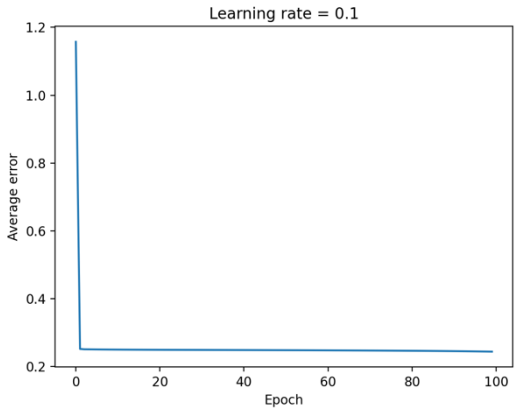
- **Epochs = 100**
- **Hàm kích hoạt: tanh**
- **Kích thước lớp ẩn = 3**

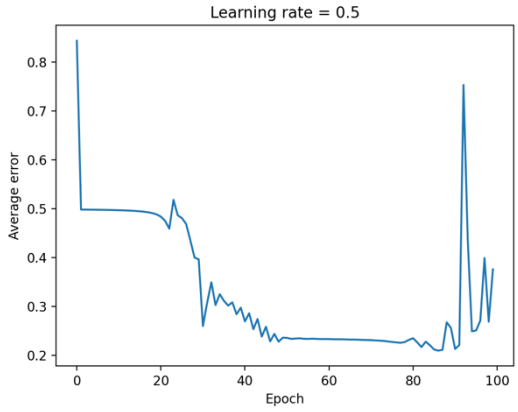
- **Số lớp ẩn = 1**

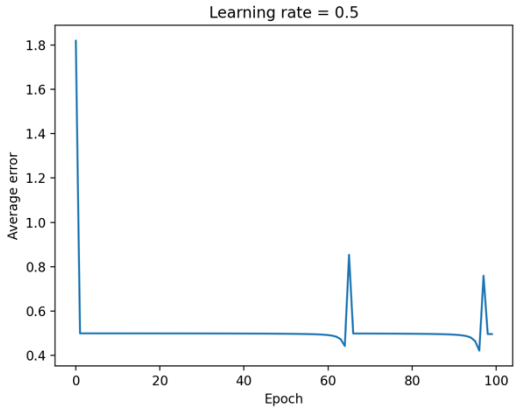
Bảng 1. Kết quả thay đổi Training Rate

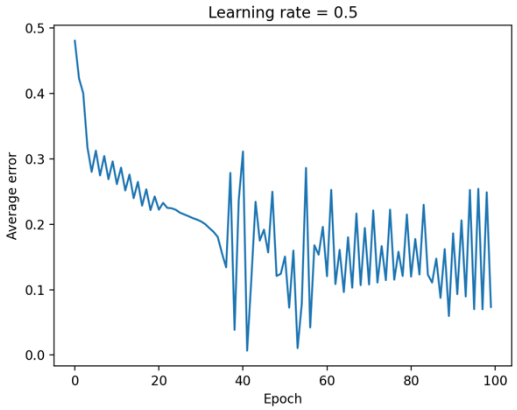
Learning rate	Số lần	Output
0	1	[tensor([[0.2054]]), tensor([[0.5943]]), tensor([[0.6639]]), tensor([[0.8129]])]
0.1	1	[tensor([[0.0108]]), tensor([[0.8679]]), tensor([[0.9019]]), tensor([[0.9698]])]
0.5	1	[tensor([[0.5167]]), tensor([[0.9992]]), tensor([[0.9975]]), tensor([[0.9991]])]
	2	[tensor([[0.5167]]), tensor([[0.9992]]), tensor([[0.9975]]), tensor([[0.9991]])]
	3	[tensor([[-0.6122]]), tensor([[0.9783]]), tensor([[0.9832]]), tensor([[0.7248]])]











Nhận xét:

Tốc độ học tập thích ứng: Điều chỉnh tham số tốc độ học tập α trong quá trình đào tạo

- α nhỏ → trọng lượng thay đổi nhỏ qua các lần lặp lại → đường cong học tập trơn tru
- α lớn → tăng tốc quá trình luyện tập với khối lượng thay đổi lớn hơn → có thể mất ổn định và dao động.

Các phương pháp tiếp cận giống như Heuristic để điều chỉnh α

- Dấu đại số của sự thay đổi SSE vẫn duy trì trong một số hệ quả của Epoch → tăng α.
- Dấu đại số của sự thay đổi SSE xen kẽ trong một số hệ quả của Epoch → giảm α.

2. Thay đổi kích thước lớp

- **Giá trị đầu vào và đầu ra mẫu cho training:**

x_train = torch.tensor([[[0, 0]], [[0, 1]], [[1, 0]], [[1, 1]]], dtype=torch.float)

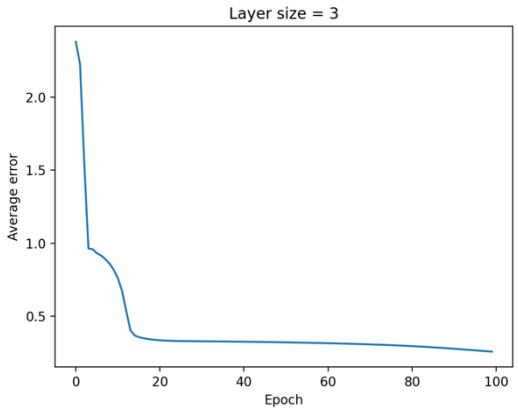
y_train = torch.tensor([[[0]], [[1]], [[1]], [[0]]], dtype=torch.float)

- **Epochs = 100**
- **Learning rate = 0.1**
- **Hàm kích hoạt: tanh**
- **Số lớp ẩn = 1**

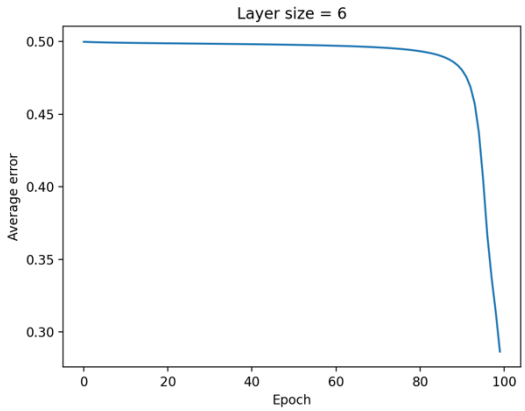
Bảng 2. Kết quả thay đổi Layer Size

Kích thước lớp	Số lần	Output
3	1	[tensor([[0.7542]]), tensor([[0.6213]]), tensor([[0.6423]]), tensor([[0.0808]])]

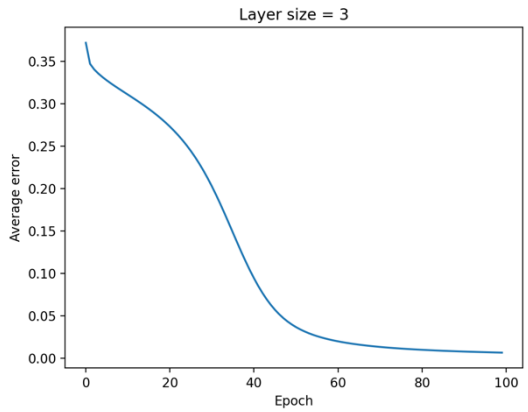
	2	[tensor([[0.0184]]), tensor([[0.8976]]), tensor([[0.8828]]), tensor([[-0.0062]])]
6	1	[tensor([[0.9878]]), tensor([[0.7641]]), tensor([[0.7707]]), tensor([[-0.1899]])]
	2	[tensor([[0.0026]]), tensor([[0.9681]]), tensor([[0.9718]]), tensor([[0.9997]])]
	3	[tensor([[-5.3048e-06]]), tensor([[0.9764]]), tensor([[0.9341]]), tensor([[0.9922]])]
	4	[tensor([[0.0651]]), tensor([[0.7875]]), tensor([[0.7706]]), tensor([[-0.0224]])]



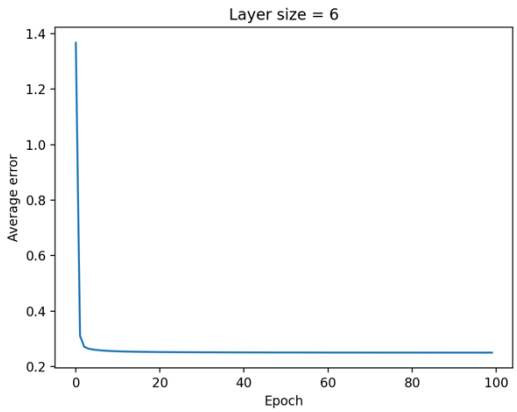
Layer size = 3 _ 1



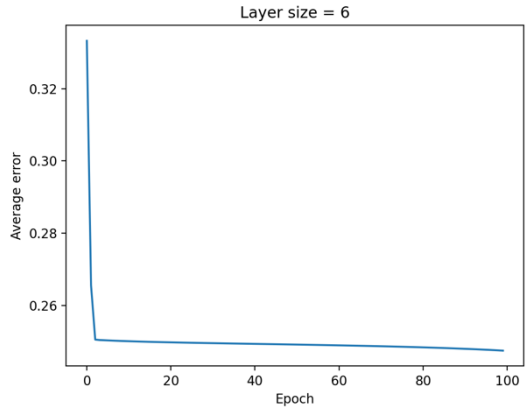
Layer size = 6 _ 1



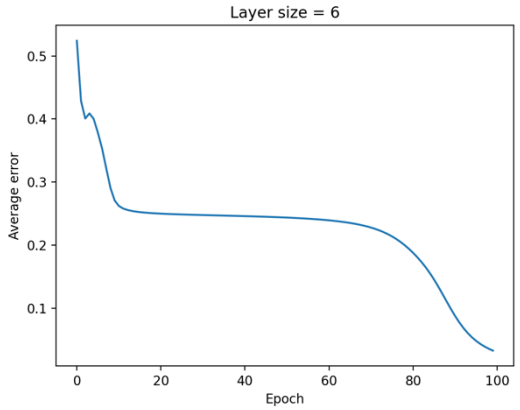
Layer size = 3 _ 2



Layer size = 6 _ 2



Layer size = 6 _ 3



Layer size = 6 _ 4

Nhận xét: Trong thực tế, ta không thể dễ dàng xác định được chính xác kích thước lớp ẩn phù hợp. Vậy nên cách tốt nhất là dùng phương pháp Thử - Sai để tìm kích thước lớp ẩn tương ứng. [4]

3. Thay đổi số lượng Hidden Layers

- **Giá trị đầu vào và đầu ra mẫu cho training:**

x_train = torch.tensor([[[0, 0]], [[0, 1]], [[1, 0]], [[1, 1]]], dtype=torch.float)

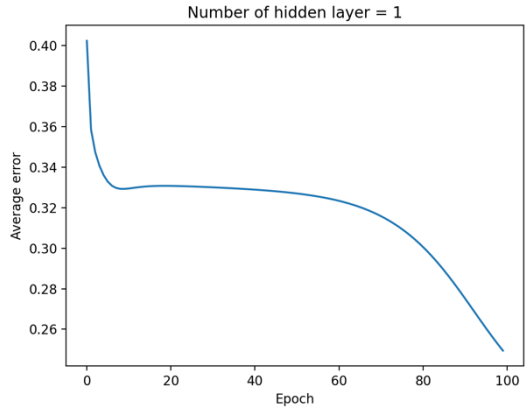
y_train = torch.tensor([[[0]], [[1]], [[1]], [[0]]], dtype=torch.float)

- **Epochs = 100**
- **Learning rate = 0.1**
- **Hàm kích hoạt: tanh**
- **Kích thước lớp ẩn = 3**

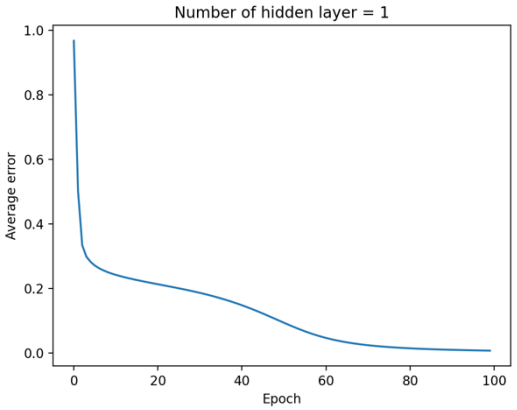
Bảng 3. Kết quả thay đổi số lượng Hidden Layers

Số lớp	Số lần	Output
1	1	[tensor([[0.2333]]), tensor([[0.6847]]), tensor([[0.5994]]), tensor([[0.7197]])]
	2	[tensor([[0.0449]]), tensor([[0.5959]]), tensor([[0.7455]]), tensor([[0.6736]])]
	3	[tensor([[0.7891]]), tensor([[0.7181]]), tensor([[0.6718]]), tensor([[0.0214]])]
	4	[tensor([[0.0250]]), tensor([[0.8865]]), tensor([[0.8752]]), tensor([[-0.0162]])]
6	1	[tensor([[0.2533]]), tensor([[0.8304]]), tensor([[0.6090]]), tensor([[0.1900]])]

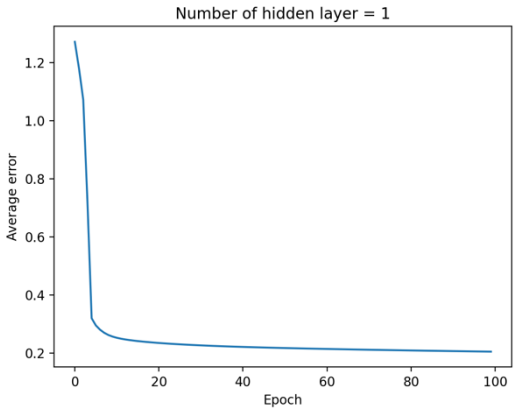
	2	[tensor([[0.4288]]), tensor([[0.7746]]), tensor([[0.4366]]), tensor([[0.4503]])]
	3	[tensor([[0.0085]]), tensor([[0.9451]]), tensor([[0.9213]]), tensor([[-0.0035]])]
	4	[tensor([[0.0018]]), tensor([[0.9482]]), tensor([[0.9586]]), tensor([[-0.0005]])]



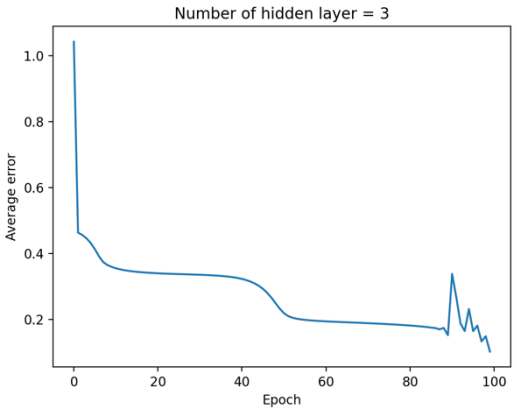
Nof Hidden Layer = 1 _ 1



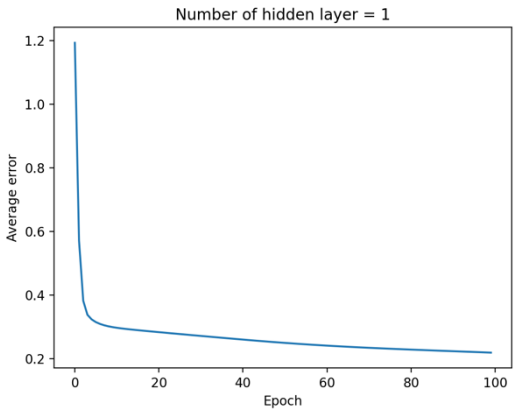
Nof Hidden Layer = 1 _ 4



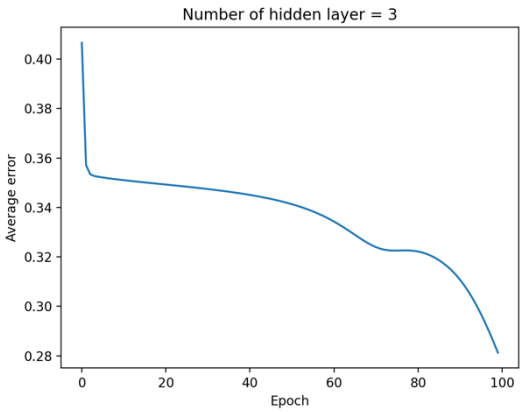
Nof Hidden Layer = 1 _ 2



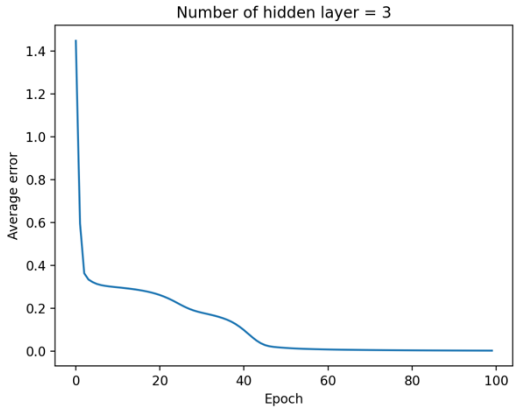
Nof Hidden Layer = 3 _ 1



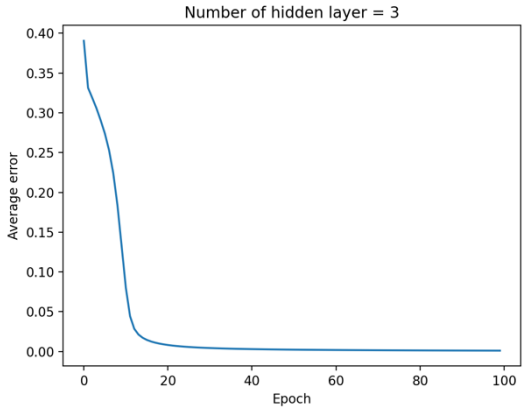
Nof Hidden Layer = 1 _ 3



Nof Hidden Layer = 3 _ 2



Nof Hidden Layer = 3 _ 3



Nof Hidden Layer = 3 _ 4

Nhận xét: Số Hidden Layer càng tăng thì kết quả càng ổn định hơn (ít thay đổi)

4. Thay đổi loại hàm kích hoạt.

- **Giá trị đầu vào và đầu ra mẫu cho training:**

x_train = torch.tensor([[[0, 0]], [[0, 1]], [[1, 0]], [[1, 1]]], dtype=torch.float)

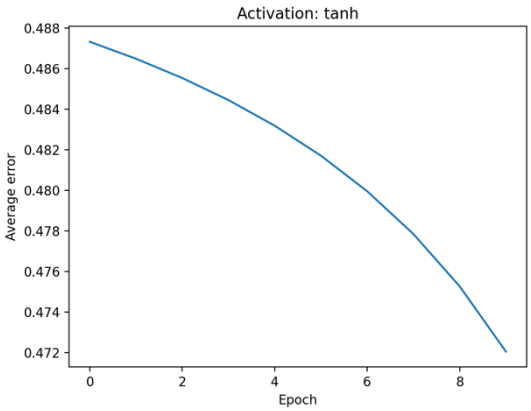
y_train = torch.tensor([[[0]], [[1]], [[1]], [[0]]], dtype=torch.float)

- **Epochs = 10 (do Epochs = 100 quá tải với sigmoid)**
- **Learning rate = 0.1**
- **Kích thước lớp ẩn = 3**
- **Số lớp ẩn = 1**

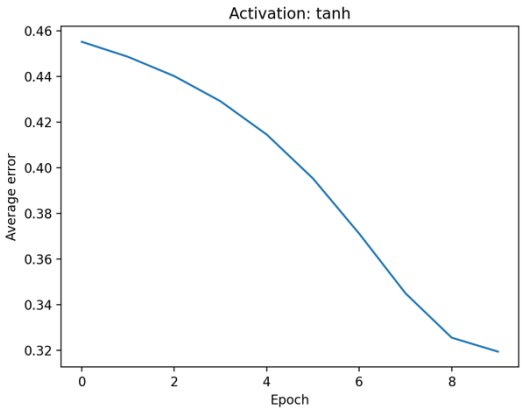
Bảng 4. Kết quả thay đổi Hàm kích hoạt

Hàm kích hoạt	Số lần	Output
Tanh	1	[tensor([[0.9937]]), tensor([[0.9839]]), tensor([[0.9576]]), tensor([[0.9396]])]
	2	[tensor([[0.4172]]), tensor([[0.6817]]), tensor([[0.5906]]), tensor([[0.6912]])]
	3	[tensor([[0.7499]]), tensor([[0.6148]]), tensor([[0.5918]]), tensor([[0.2004]])]
	4	[tensor([[0.6729]]), tensor([[0.6645]]), tensor([[0.5412]]), tensor([[0.5156]])]
Sigmoid	1	[tensor([[0.4851]]), tensor([[0.5527]]), tensor([[0.5008]]), tensor([[0.5621]])]
	2	[tensor([[nan]]), tensor([[nan]]), tensor([[nan]]), tensor([[nan]])]

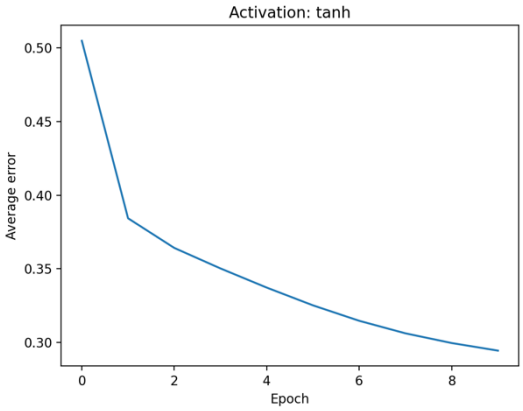
	3	[tensor([[0.6726]]), tensor([[0.7080]]), tensor([[0.6979]]), tensor([[0.7156]])]
	4	[tensor([[nan]]), tensor([[nan]]), tensor([[nan]]), tensor([[nan]])]



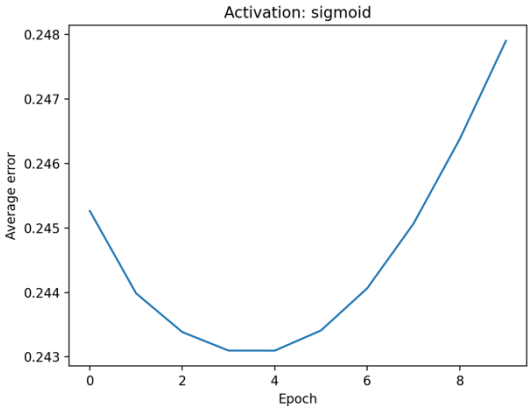
tanh _ 1



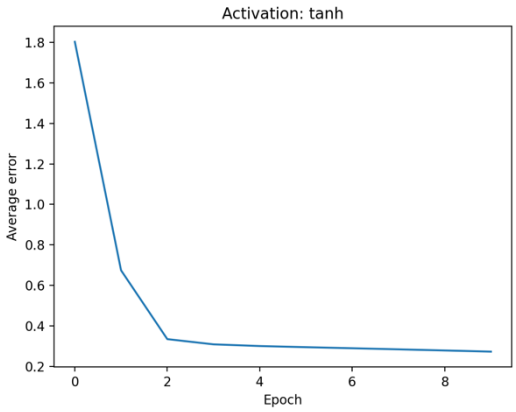
tanh _ 4



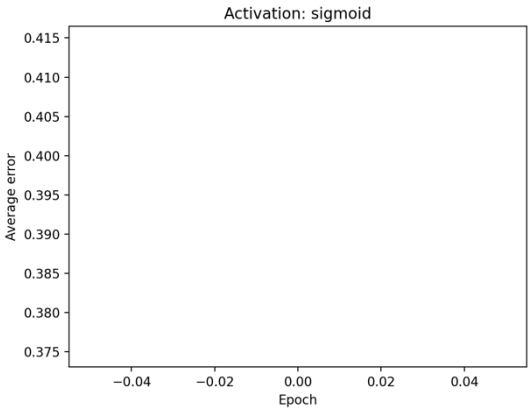
tanh _ 2



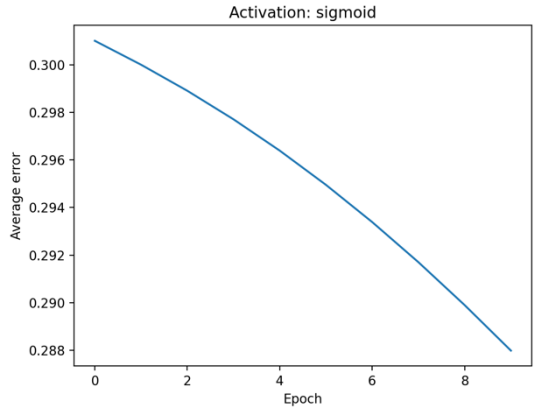
sigmoid _ 1



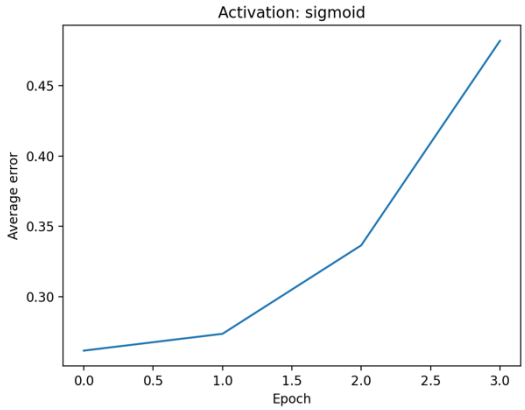
tanh _ 3



sigmoid _ 2



sigmoid _ 3



sigmoid _ 4

Nhận xét: Tùy theo kiểu dữ liệu ra mong muốn, mà bạn chọn hàm truyền ngõ ra hợp lý. Ví dụ, nếu dữ liệu ra biến thiên trong [0,1] bạn có thể chọn hàm Sigmoid, nếu dữ liệu ra biến thiên trong [-1,1], bạn có thể chọn hàm Tanh. Thông thường nên chọn hàm truyền tuyến tính, để khỏi bận tâm về 'giới hạn' dữ liệu ngõ ra. [3]

5. Thay đổi Epochs

- **Giá trị đầu vào và đầu ra mẫu cho training:**

x_train = torch.tensor([[[0, 0]], [[0, 1]], [[1, 0]], [[1, 1]]], dtype=torch.float)

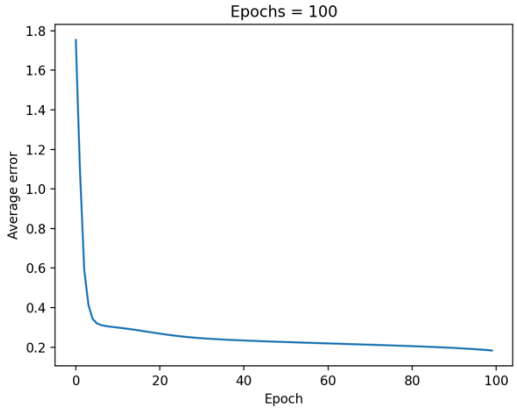
y_train = torch.tensor([[[0]], [[1]], [[1]], [[0]]], dtype=torch.float)

- **Learning rate = 0.1**
- **Hàm kích hoạt: tanh**
- **Kích thước lớp ẩn = 3**
- **Số lớp ẩn = 1**

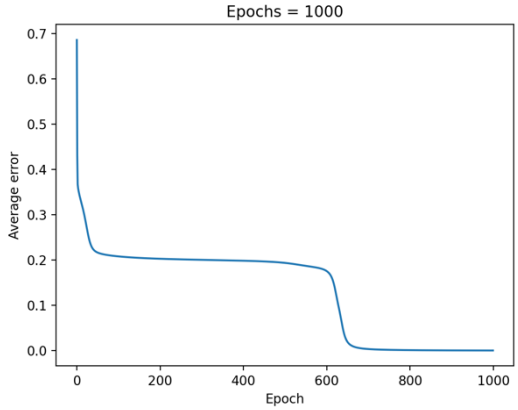
Bảng 5. Kết quả thay đổi Epochs

Epochs	Lần	Output
100	1	[tensor([[0.6525]]), tensor([[0.6776]]), tensor([[0.7054]]), tensor([[0.0201]])]
	2	[tensor([[0.4843]]), tensor([[0.5925]]), tensor([[0.5659]]), tensor([[0.5869]])]
1000	1	[tensor([[-0.0044]]), tensor([[0.9673]]), tensor([[0.9665]]), tensor([[-0.0238]])]
	2	[tensor([[0.0011]]), tensor([[0.9788]]), tensor([[0.9779]]), tensor([[0.0080]])]

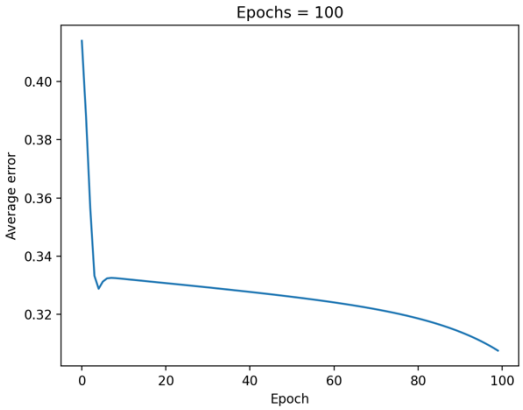
	3	[tensor([[0.0011]]), tensor([[0.9667]]), tensor([[0.9561]]), tensor([[-0.0006]])]
	4	[tensor([[0.0007]]), tensor([[0.9789]]), tensor([[0.9815]]), tensor([[-0.0008]])]
	5	[tensor([[0.0003]]), tensor([[0.9794]]), tensor([[0.9799]]), tensor([[0.0001]])]
	6	[tensor([[0.0006]]), tensor([[0.9766]]), tensor([[0.9716]]), tensor([[-0.0005]])]
	7	[tensor([[0.0007]]), tensor([[0.9809]]), tensor([[0.9799]]), tensor([[-0.0006]])]
	8	[tensor([[0.0025]]), tensor([[0.9647]]), tensor([[0.5133]]), tensor([[0.5205]])]



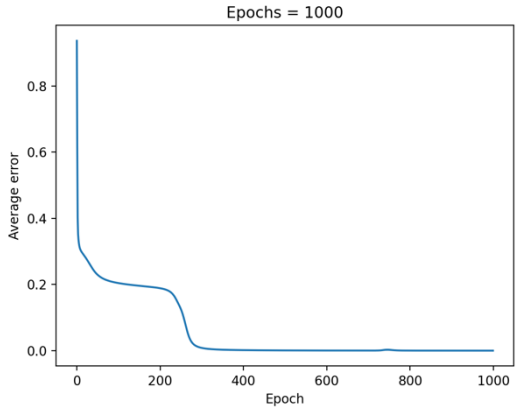
Epochs = 100 _ 1



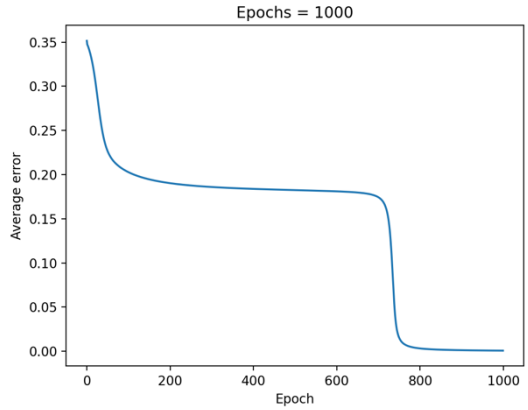
Epoch = 1000 _ 1



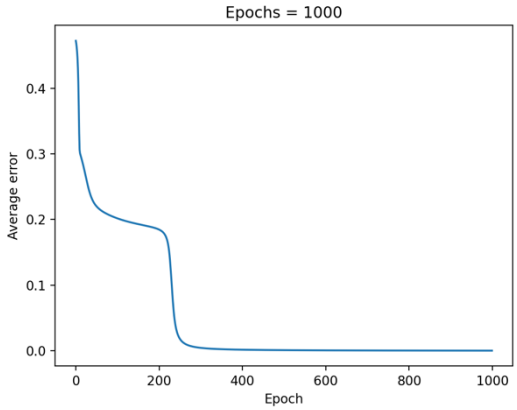
Epochs = 100 _ 2



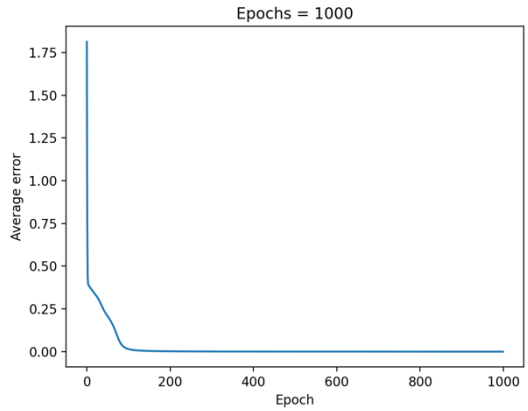
Epoch = 1000 _ 2



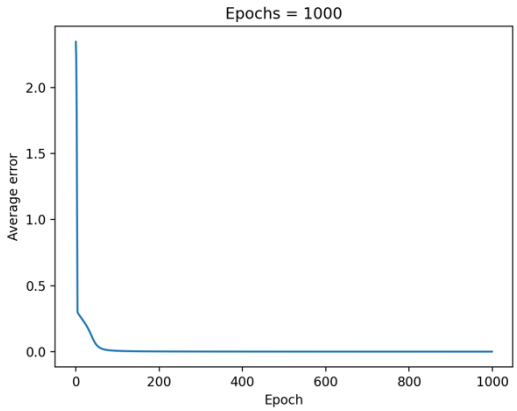
Epoch = 1000 _ 3



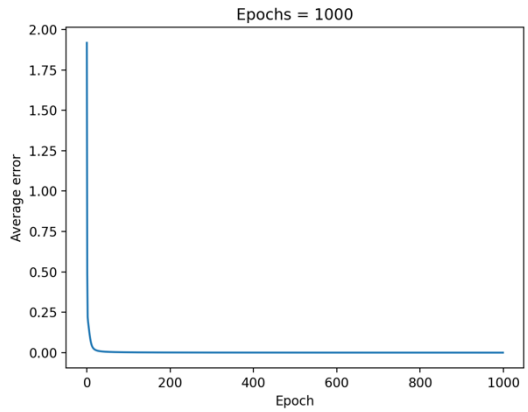
Epoch = 1000 _ 6



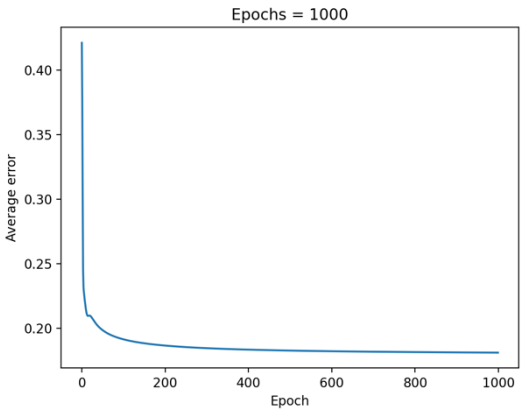
Epoch = 1000 _ 4



Epoch = 1000 _ 7



Epoch = 1000 _ 5



Epoch = 1000 _ 8

Nhận xét: Tình trạng quá khớp (over fitting) khi mạng được huấn luyện lâu. Quá trình huấn luyện sẽ kết thúc khi đạt Maximun Epochs, hoặc khi MSE đạt giá trị ngưỡng Threshold,… [3]

III. Nguồn tham khảo

[1] Feedforward neural network là gì, honamphoto
[2] http://thanhtra.nguyen.free.fr/ann/feedforward_1.html
[3] https://pdfcoffee.com/phan-mem-neurosolutions-4-pdf-free.html
[4] https://123docz.net/trich-doan/2440957-so-noron-trong-lop-an.htm