Exercício 1:

np.random.seed(0)
torch.manual_seed(0)
random.seed(0)

	Original Prof	Tentativa 1	Tentativa 2
epochs	50	100	100
batch size	64	64	64
learning rate	0.001	0.001	0.001
size splits	test: 116 train: 235	test: 116 train: 235	test: 116 train: 235
layers + activation functions	MLP((hidden1): Linear(in_features=34, out_features=10, bias=True) (act1): ReLU() (hidden2): Linear(in_features=10, out_features=8, bias=True) (act2): ReLU() (hidden3): Linear(in_features=8, out_features=1, bias=True) (act3): Sigmoid())	MLP((hidden1): Linear(in_features=34, out_features=10, bias=True) (act1): ReLU() (hidden2): Linear(in_features=10, out_features=8, bias=True) (act2): ReLU() (hidden3): Linear(in_features=8, out_features=1, bias=True) (act3): Sigmoid())	MLP((hidden1): Linear(in_features=34, out_features=20, bias=True) (act1): ReLU() (hidden2): Linear(in_features=20, out_features=10, bias=True) (act2): ReLU() (hidden3): Linear(in_features=10, out_features=8, bias=True) (act3): ReLU() (hidden4): Linear(in_features=8, out_features=1, bias=True) (act4): Sigmoid())
loss function	BCEWithLogitsLoss()	BCEWithLogitsLoss()	BCEWithLogitsLoss()
optimization function	Adam(model.parameters(), lr=LEARNING_RATE)	Adam(model.parameters(), lr=LEARNING_RATE)	Adam(model.parameters(), lr=LEARNING_RATE)
accuracy	s/ strat: 0.716 c/ strat: 0.784	s/ strat: 0.793 c/ strat: 0.862	s/ strat: 0.888 c/ strat: 0.862

Exercício 2:

np.random.seed(0)
torch.manual_seed(0)
random.seed(0)

	Tentativa 1 - Original	Tentativa 2	Tentativa 3
epochs	50	100	100
batch size	32	32	32
learning rate	0.01	0.01	0.01
size splits	Test: 50 train: 100	Test: 50 train: 100	Test: 50 train: 100
layers + activation functions	MLP((hidden1): Linear(in_features=4, out_features=10, bias=True) (act1): ReLU() (hidden2): Linear(in_features=10, out_features=8, bias=True) (act2): ReLU() (hidden3): Linear(in_features=8, out_features=3, bias=True) (act3): Softmax(dim=1))	MLP((hidden1): Linear(in_features=4, out_features=10, bias=True) (act1): ReLU() (hidden2): Linear(in_features=10, out_features=8, bias=True) (act2): ReLU() (hidden3): Linear(in_features=8, out_features=3, bias=True) (act3): Softmax(dim=1))	MLP((hidden1): Linear(in_features=4, out_features=10, bias=True) (act1): ReLU() (hidden2): Linear(in_features=10, out_features=8, bias=True) (act2): ReLU() (hidden3): Linear(in_features=8, out_features=3, bias=True) (act3): Softmax(dim=1))
loss function	CrossEntropyLoss()	CrossEntropyLoss()	CrossEntropyLoss()
optimization function	Adam(model.parameters(), lr=LEARNING_RATE)	Adam(model.parameters(), lr=LEARNING_RATE)	SGD(model.parameters(), lr=LEARNING_RATE, momentum=0.5)
accuracy	0.940	0.880	0.700

Exercício 3:

np.random.seed(0)
torch.manual_seed(0)
random.seed(0)

	Tentativa 1 - Original	Tentativa 2	Tentativa 3
epochs	100	200	
batch size	32	32	
learning rate	0.001	0.001	
size splits	Test: 10000 train: 60000	Test: 10000 train: 60000	
layers + activation functions	MLP((hidden1): Linear(in_features=784, out_features=20, bias=True) (act1): ReLU() (hidden2): Linear(in_features=20, out_features=20, bias=True) (act2): ReLU() (hidden3): Linear(in_features=20, out_features=10, bias=True) (act3): Softmax(dim=1))	MLP((hidden1): Linear(in_features=784, out_features=40, bias=True) (act1): ReLU() (hidden2): Linear(in_features=40, out_features=20, bias=True) (act2): ReLU() (hidden3): Linear(in_features=20, out_features=10, bias=True) (act3): ReLU() (hidden4): Linear(in_features=10, out_features=10, bias=True) (act4): Softmax(dim=1))	
loss function	CrossEntropyLoss()	CrossEntropyLoss()	
optimization function	Adam(model.parameters(), lr=LEARNING_RATE)	Adam(model.parameters(), lr=LEARNING_RATE)	
accuracy	0.380	0.629	