Ứng dụng Trí tuệ nhân tạo trong Nuôi trồng thủy sản

NGUYỄN HẢI TRIỀU[1]

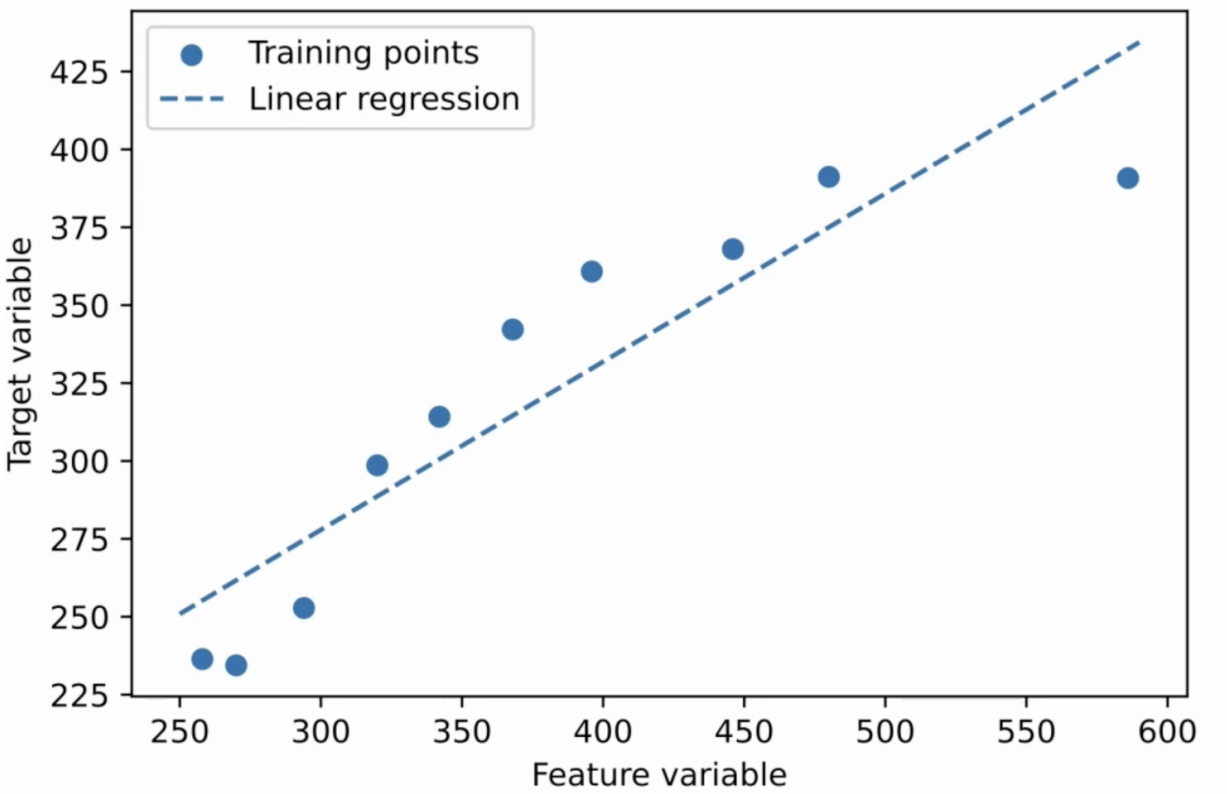
[1] Bộ môn Kỹ thuật phần mềm,

Khoa Công nghệ thông tin, Trường ĐH Nha Trang

NhaTrang, September 2024

Bài toán Regression

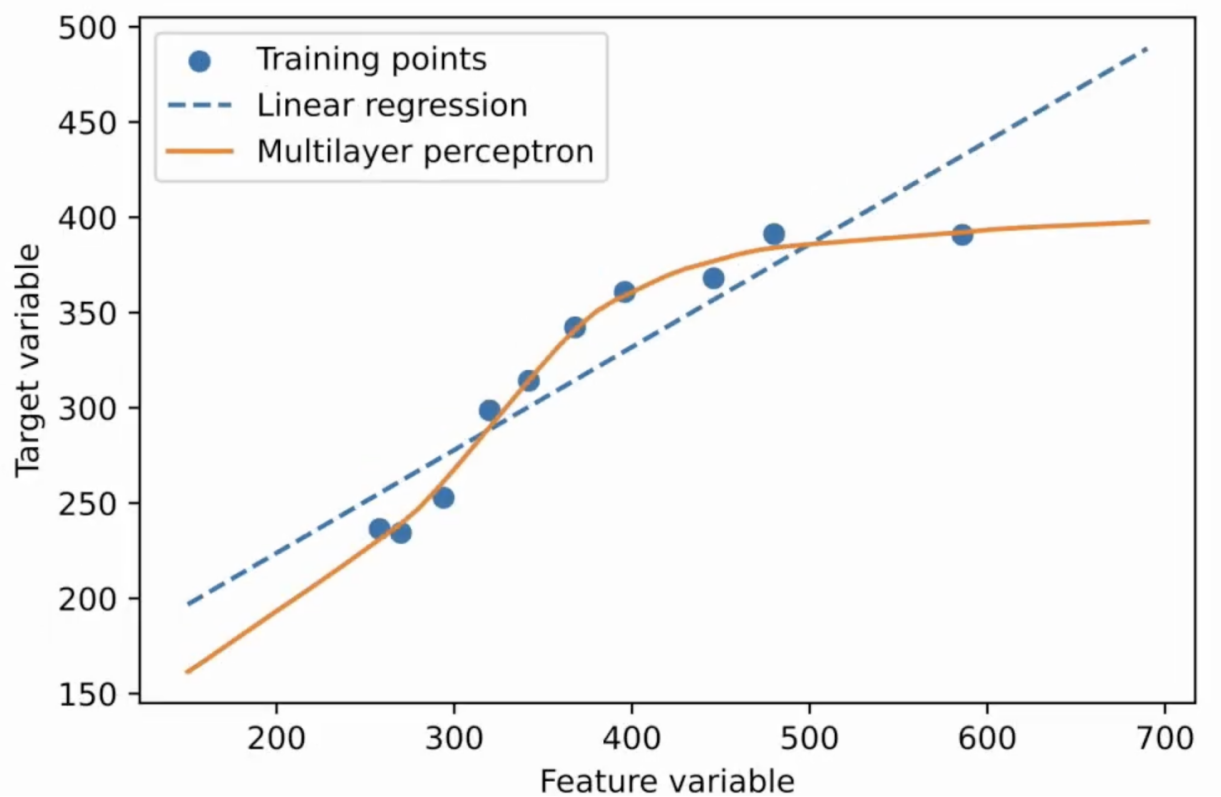
Trong các phần trước, chúng ta đã xem xét bài toán phân loại. Tiếp theo chúng ta quay lại *bài toán Regression dùng để dự đoán 1 giá trị liên tục.*



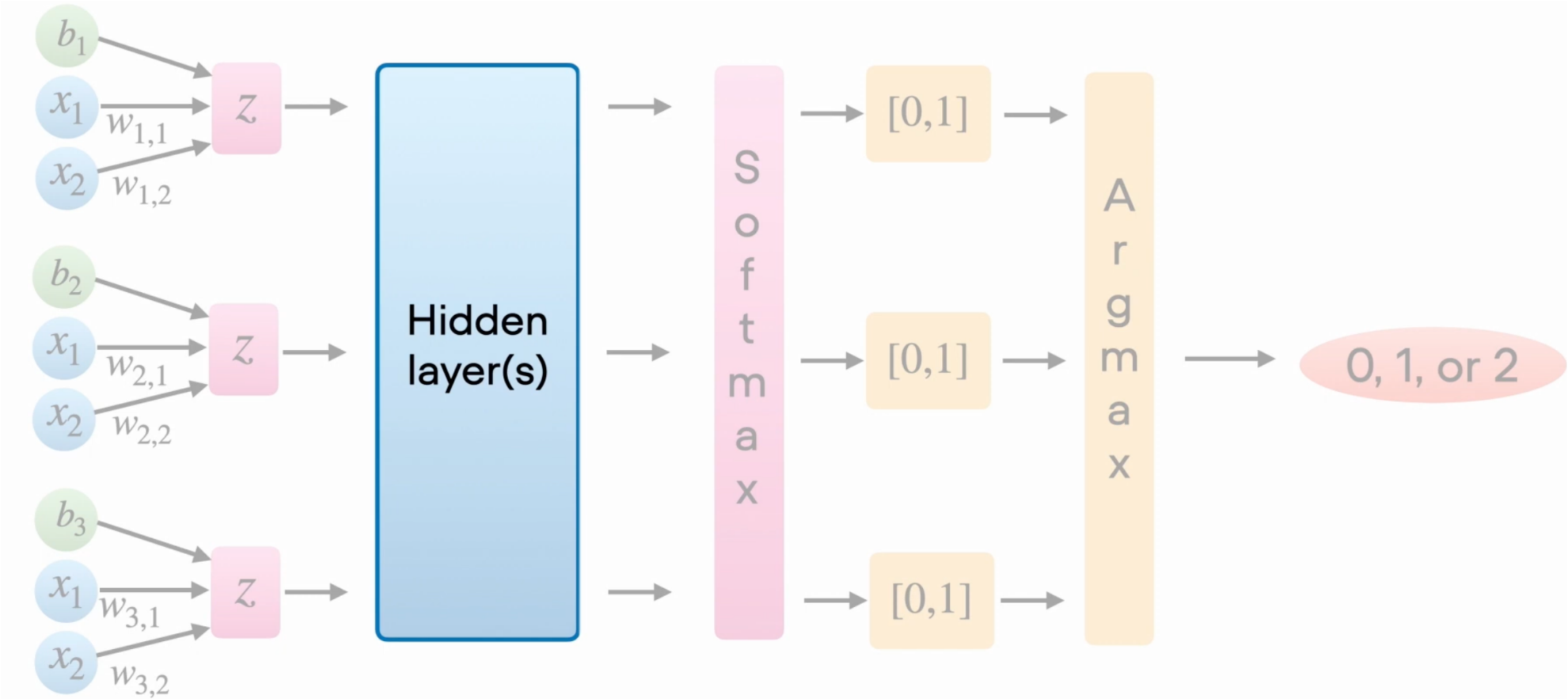
A Simple Regression Dataset with 1 feature.

Bài toán Regression

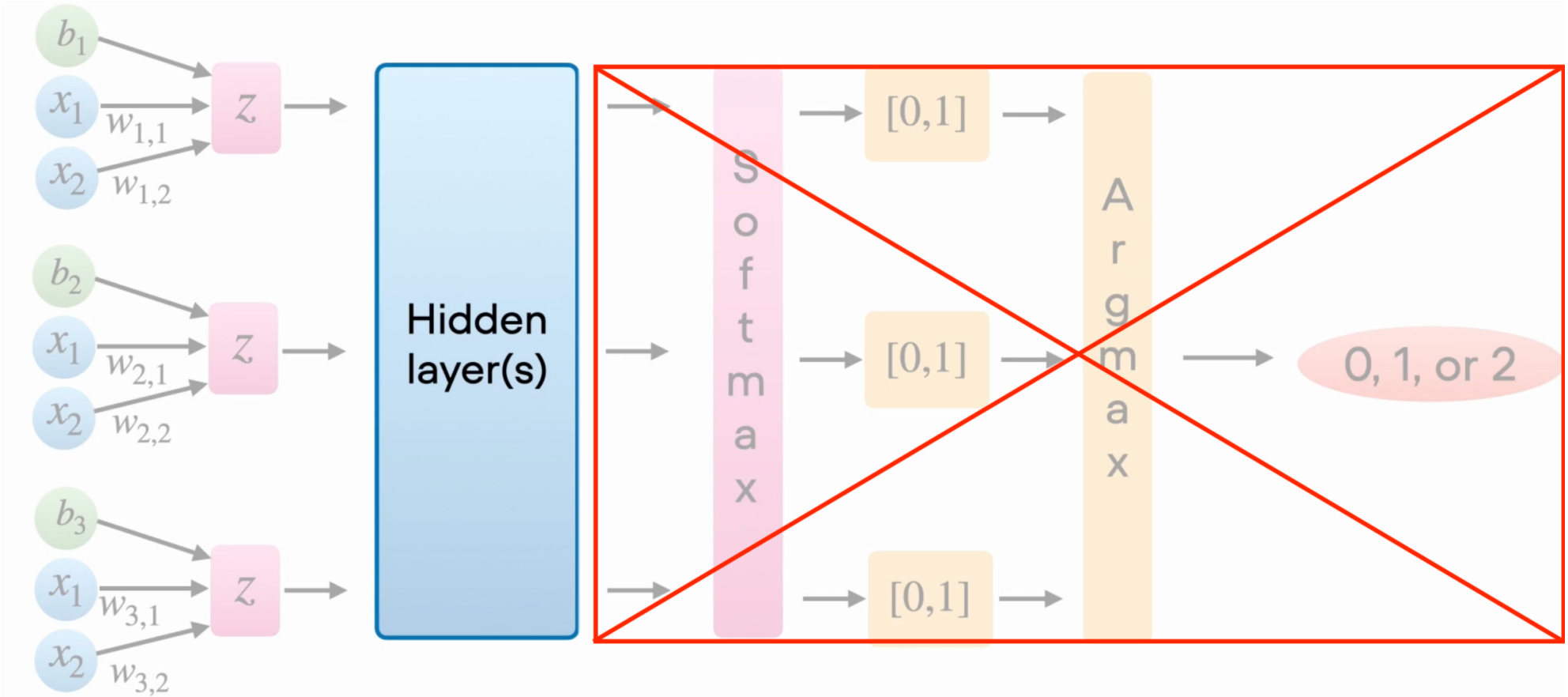
Thay vì sử dụng Linear regression, chúng ta có thể sử dụng Multilayer Perceptron model sẽ cho kết quả dự đoán tốt hơn (*pretty good at fitting data*). Tuy nhiên lưu ý vấn đề overfitting khi quá khớp dữ liệu.



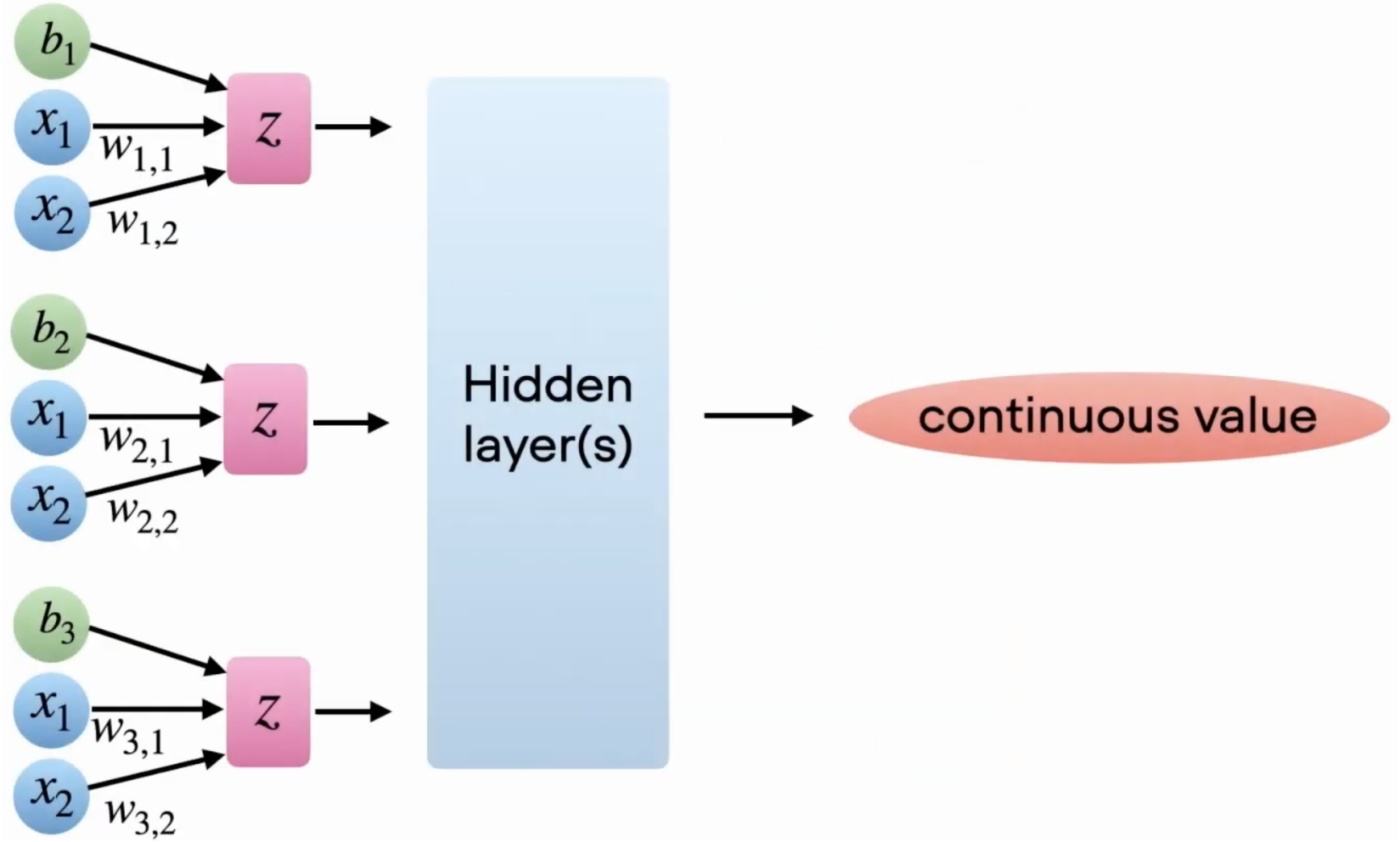
Vậy câu hỏi đặt ra, làm sao chỉnh sửa mạng MLP cho bài toán phân loại thành bài toán regression?.

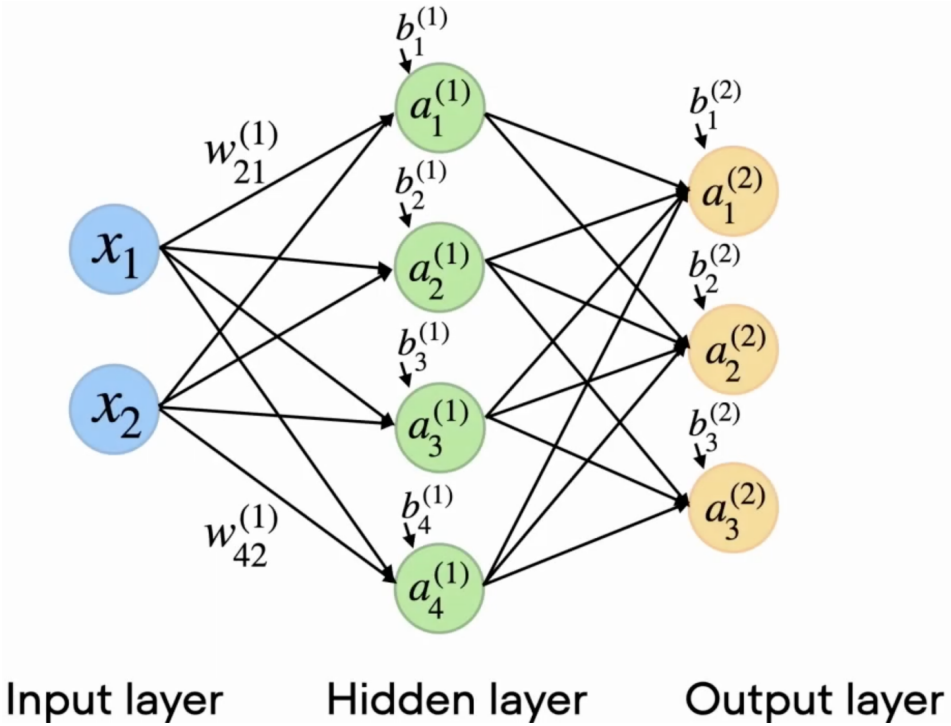


Vậy câu hỏi đặt ra, làm sao chỉnh sửa mạng MLP cho bài toán phân loại thành bài toán regression?.
Không cần quan tâm đến việc phân loại nhãn, bỏ đi Softmax và Argmax. *Chỉ nhận giá trị liên tục sau lớp ẩn.*

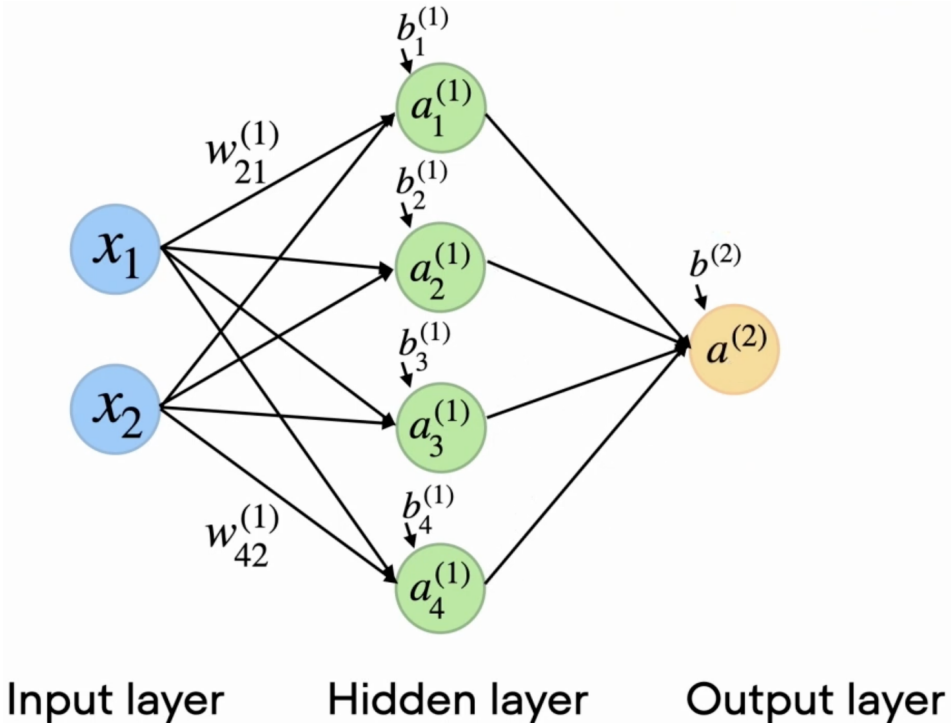


Không cần quan tâm đến việc phân loại nhãn, bỏ đi Softmax và Argmax. *Chỉ nhận giá trị liên tục sau lớp ẩn.*





(a) Multilayer Perceptron Model For Classification, 3 output nodes = 3 classes



(b) Multilayer Perceptron Model For Regression

Hình 1: The multilayer perceptron here takes two input features, has one hidden layer consisting of four units.

Loss function

Với bài toán phân loại, ta thường sử dụng **Cross Entropy**, còn trong bài toán regression, chúng ta sử dụng hàm **Mean squared Error** để tính *sự sai khác giữa giá trị đúng và giá trị dự đoán từ mô hình.*

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \hat{y}^i\right)^2.$$

Trong đó, N là số lượng mẫu huấn luyện, $\hat{y}^i$ giá trị dự đó thứ i của mô hình.

Chúng ta sẽ tiến hành cài đặt mô hình MLPs cho bài toán hồi quy Regression sử dụng PyTorch dựa trên chỉnh sửa mô hình MLPs cho bài toán phân loại.

Ví dụ dữ liệu mẫu có 1 đặc trưng

```
1  import torch
2
3  X_train = torch.tensor(
4    [258.0, 270.0, 294.0, 320.0, 342.0, 368.0, 396.0, 446.0, 480.0,
         586.0]).view(-1, 1)
5
6  y_train = torch.tensor(
7    [236.4, 234.4, 252.8, 298.6, 314.2, 342.2, 360.8, 368.0, 391.2, 390.8])
```

Chuẩn hoá dữ liệu bằng Z-Score Standardization, sau chuẩn hoá giá trị trung bình bằng 0, độ lệch chuẩn bằng 1.

```
x_mean, x_std = X_train.mean(), X_train.std()
y_mean, y_std = y_train.mean(), y_train.std()

X_train_norm = (X_train - x_mean) / x_std
y_train_norm = (y_train - y_mean) / y_std
```

Thiết lập Dataset, DataLoader

```
from torch.utils.data import DataLoader, Dataset


class MyDataset(Dataset):
  def __init__(self, X, y):

    self.features = X
    self.targets = y

  def __getitem__(self, index):
    x = self.features[index]
    y = self.targets[index]
    return x, y

  def __len__(self):
```

```
16      return self.targets.shape[0]
17
18
19  train_ds = MyDataset(X_train_norm, y_train_norm)
20
21  train_loader = DataLoader(
22    dataset=train_ds,
23    batch_size=20,
24    shuffle=True,)
```

Mạng MLP cho Regression. *Lưu ý*: đầu ra là **lớp Linear** chỉ có duy nhất 1 node là giá trị liên tục và khi khởi tạo model, không có tham số **num_classes**.

```
1  class PyTorchMLP(torch.nn.Module):
2  def __init__(self, num_features):
3    super().__init__()
4
5    self.all_layers = torch.nn.Sequential(
6      # 1st hidden layer
7      torch.nn.Linear(num_features, 50),
8      torch.nn.ReLU(),
```

```
9       # 2nd hidden layer
10      torch.nn.Linear(50, 25),
11      torch.nn.ReLU(),
12      # output layer
13      torch.nn.Linear(25, 1), ## Only 1 output unit
14    )
15
16  def forward(self, x):
17    logits = self.all_layers(x).flatten()
18    return logits
```

Huấn luyện mô hình MLP cho bài toán Regression. *Lưu ý*: Ví dụ mẫu chỉ có duy nhất 1 đặc trưng nên **num_features=1**, hàm Loss sử dụng Mean Squared Loss:
torch.nn.functional.mse_loss

```
import torch.nn.functional as F

torch.manual_seed(1)
model = PyTorchMLP(num_features=1)

optimizer = torch.optim.SGD(model.parameters(), lr=0.1)

num_epochs = 30

loss_list = []
train_acc_list, val_acc_list = [], []
for epoch in range(num_epochs):

  model = model.train()
  for batch_idx, (features, targets) in enumerate(train_loader):

    logits = model(features)
    loss = F.mse_loss(logits, targets)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if not batch_idx % 250:
```

```
        ### LOGGING
        print(
          f"Epoch: {epoch+1:03d}/{num_epochs:03d}"
          f" | Batch {batch_idx:03d}/{len(train_loader):03d}"
          f" | Train Loss: {loss:.2f}"
        )
      loss_list.append(loss.item())
```

Dự đoán trên bộ dữ liệu mới. *Lưu ý:* phải chuẩn hoá bộ dữ liệu mới **dựa trên trung bình và độ lệch chuẩn của bộ dữ liệu đã huấn luyện**. *Kết quả thu được là các* ***giá trị liên tục đã được chuẩn hoá.***

```
model.eval()

X_range = torch.arange(150, 800, 0.1).view(-1, 1)
X_range_norm = (X_range - x_mean) / x_std

# predict
with torch.no_grad():
  y_mlp_norm = model(X_range_norm)
```

```
9
10  # MLP returns normalized predictions
```

Hiển thị dựa trên **kết quả dự đoán đã chuẩn hoá**. *Lưu ý*: để hiện thị kết quả thực tế, chúng ta phải chuyển kết quả dự đoán đã chuẩn hoá về dữ liệu chưa chuẩn hoá và vẽ đồ thị scatter như thường.

```
1   # undo normalization of preditions for plotting
2   y_mlp = y_mlp_norm * y_std + y_mean
3
4   # plot results
5   plt.scatter(X_train, y_train, label="Training points")
6   plt.plot(X_range, y_mlp, color="C1", label="MLP fit", linestyle="-")
7
8   plt.xlabel("Feature variable")
9   plt.ylabel("Target variable")
10  plt.legend()
11  # plt.savefig("mlp.pdf")
12  plt.show()
```

Chi tiết code xem ở file "*Chuong_4.3_mlp-regression*"

Tài liệu tham khảo


Sebastian Raschka, Yuxi (Hayden) Liu, Vahid Mirjalili
Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python (2022). Published by Packt Publishing Ltd, ISBN 978-1-80181-931-2.


Sebastian Raschka
MACHINE LEARNING Q AND AI: 30 Essential Questions and Answers on Machine Learning and AI (2024). ISBN-13: 978-1-7185-0377-9 (ebook).


LightningAI
LightningAI: PyTorch Lightning (2024) .