Exercise: Predicting the release year of a song

Increase accuracy of the prediction

Submission requirements:

1. source code (predict.py)
2. PDF documents
   Explaining your strategy.
   Show the outputs (before and after)
3. Upload to e-learning before 4/8 14:10

Calculate Accuracy

```
def accuracy():
    temp = 0
    for i in range(len(pred)):
        truth_value = y_test_torch[i].item()
        if truth_value - 2 <= pred[i].item() <= truth_value + 2:
            temp += 1

    print(f"Accuracy: {round(temp / 5000, 2)}")
```

The accuracy function

Use the range of [truth_value - 2, truth_value + 2] to calculate the accuracy of the predictions. The original accuracy is 0.42.

The modifies to enhance the accuracy

1. Expand the model

```
model = nn.Sequential(
    nn.Linear(x_train.shape[1], 200),
    nn.ReLU(),
    nn.Linear(200, 50),
    nn.ReLU(),
    nn.Linear(50, 1)
)
```

Before

```
model = nn.Sequential(
    nn.Linear(x_train.shape[1], 200),
    nn.ReLU(),
    nn.Linear(200, 50),
    nn.ReLU(),
    nn.Linear(50, 25),
    nn.ReLU(),
    nn.Linear(25, 1)
)
```

After

Add one linear layer with ReLU function to expand the model structure.

2. Modify the ratio of training set and testing set

```
# 拆分數據成 2 個子集，x_new: x_test = 80: 20
x_new, x_test, y_new, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)

# 再拆分數據集 x_new 成 2 個子集, x_train: x_dev = 75: 25
x_train, x_dev, y_train, y_dev = train_test_split(x_new, y_new, test_size = 0.25, random_state = 0)

print(x_train.shape, x_dev.shape, x_test.shape)
```

Before

```python
# 拆分數據成 2 個子集，x_new: x_test = 70: 30
x_new, x_test, y_new, y_test = train_test_split(x, y, test_size = 0.3, random_state = 0)

# 再拆分數據集 x_new 成 2 個子集, x_train: x_dev = 75: 25
x_train, x_dev, y_train, y_dev = train_test_split(x_new, y_new, test_size = 0.25, random_state = 0)

print(x_train.shape, x_dev.shape, x_test.shape)
```

After

Modify the ratio of training set and testing set from 4: 1 to 7: 3.

3. Modify the number of epochs

```python
epochs = 5001
for epoch in range(epochs):
    model.train()
    y_pred = model(x_train_torch)
    train_loss = myloss(y_pred, y_train_torch)
    optimizer.zero_grad()
    train_loss.backward()
    optimizer.step()

    if epoch % 400 == 0:
        with torch.no_grad():
            model.eval()
            y_pred2 = model(x_dev_torch)
            valid_loss = myloss(y_pred2, y_dev_torch)

        print(f"epoch = {epoch}, train_loss: {train_loss.item():.3f}, valid_loss: {valid_loss.item():.3f}")

        if train_loss.item() < 81:
            break
```

Before

```python
epochs = 8001
for epoch in range(epochs):
    model.train()
    y_pred = model(x_train_torch)
    train_loss = myloss(y_pred, y_train_torch)
    optimizer.zero_grad()
    train_loss.backward()
    optimizer.step()

    if epoch % 400 == 0:
        with torch.no_grad():
            model.eval()
            y_pred2 = model(x_dev_torch)
            valid_loss = myloss(y_pred2, y_dev_torch)

        print(f"epoch = {epoch}, train_loss: {train_loss.item():.3f}, valid_loss: {valid_loss.item():.3f}")

        if train_loss.item() < 81:
            break
```

After

Modify the number of epochs from 5001 to 8001. Make the process of training longer.

Result

The accuracy increases from 0.42 to 0.54.