MultilayerPerceptron_XOR

November 28, 2024

1 Implementing a Multilayer Perceptron on XOR dataset

1.1 2) Loading the Dataset

[1]:
```
import pandas as pd
df = pd.read_csv('xor.csv')
df
```

[1]:
```
           x1        x2  class label
0    0.781306  1.062984            0
1   -1.060524 -1.095550            0
2    0.632125  0.674028            0
3   -1.424712  0.535203            1
4    1.383161  1.368510            0
..        …         …          …
745  0.792484  0.839275            0
746  0.582466 -0.749250            1
747 -1.593475  0.671721            1
748 -0.812671 -0.268542            0
749 -1.286524  0.655459            1

[750 rows x 3 columns]
```

[2]:
```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(1.2.2)
Requirement already satisfied: numpy>=1.17.3 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from scikit-learn) (1.24.2)
Requirement already satisfied: scipy>=1.3.2 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from scikit-learn) (3.1.0)

[notice] A new release of pip is
available: 24.2 -> 24.3.1
[notice] To update, run:
pip install --upgrade pip
```

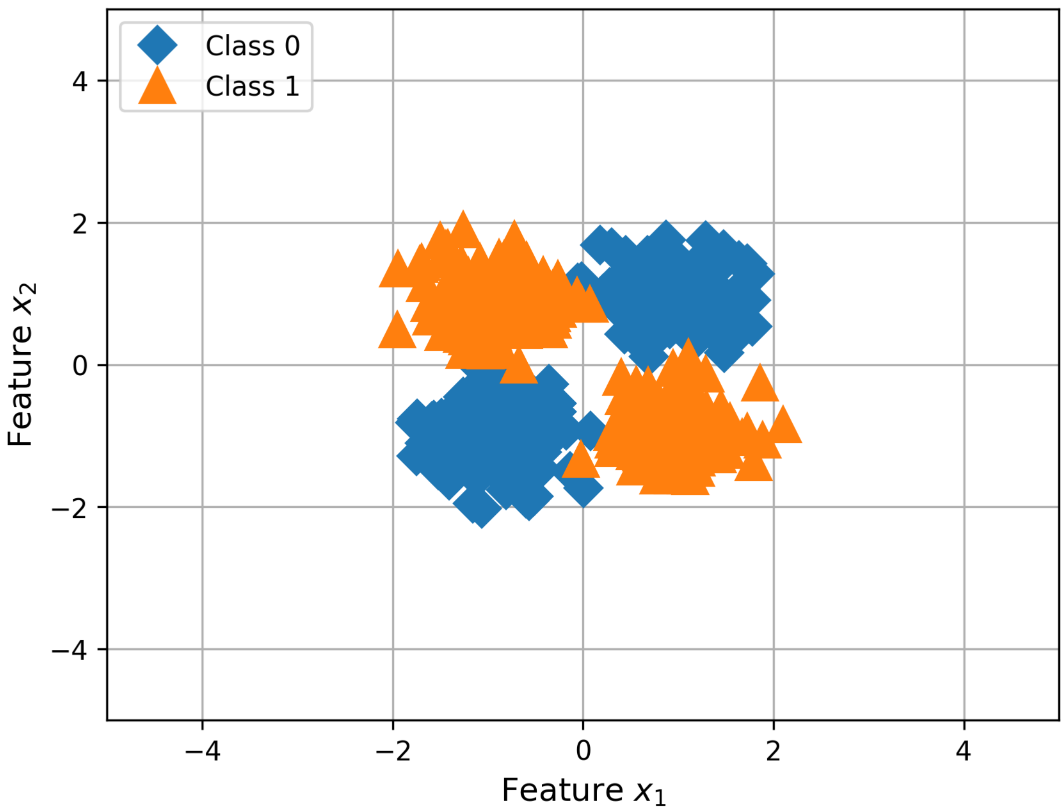
[3]:
```python
from sklearn.model_selection import train_test_split
import numpy as np

X = df[["x1", "x2"]].values
y = df["class label"].values

X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.15, random_state=1, stratify=y)

X_train, X_val, y_train, y_val = train_test_split(
        X_train, y_train, test_size=0.1, random_state=1, stratify=y_train)

print("In kích thước của các bộ dữ liệu:")
print("- Training size:", X_train.shape)
print("- Validation size :", X_val.shape)
print("- Test size: ", X_test.shape)
print("In số lượng các nhãn trong từng tập: ")
print("- Training labels:", np.bincount(y_train))
print("- Validation labels:", np.bincount(y_val))
print("- Test labels:", np.bincount(y_test))
```

```
In kích thước của các bộ dữ liệu:
- Training size: (573, 2)
- Validation size : (64, 2)
- Test size:  (113, 2)
In số lượng các nhãn trong từng tập:
- Training labels: [287 286]
- Validation labels: [32 32]
- Test labels: [57 56]
```

1.2 3) Visualizing the dataset

[4]:
```python
#%matplotlib notebook
import matplotlib.pyplot as plt
import matplotlib as mpl

mpl.rcParams['savefig.dpi'] = 80
mpl.rcParams['figure.dpi'] = 300
```

```
plt.plot(
    X_train[y_train == 0, 0],
    X_train[y_train == 0, 1],
    marker="D",
    markersize=10,
    linestyle="",
    label="Class 0",
)

plt.plot(
    X_train[y_train == 1, 0],
    X_train[y_train == 1, 1],
    marker="^",
    markersize=13,
    linestyle="",
    label="Class 1",
)

plt.legend(loc=2)

plt.xlim([-5, 5])
plt.ylim([-5, 5])

plt.xlabel("Feature $x_1$", fontsize=12)
plt.ylabel("Feature $x_2$", fontsize=12)

plt.grid()
plt.show()
```



1.3 4) Implementing the model

[5]:
```
import torch

class PyTorchMLP(torch.nn.Module):
    def __init__(self, num_features, num_classes):
        super().__init__()

        self.all_layers = torch.nn.Sequential(

            # 1st hidden layer
            torch.nn.Linear(num_features, 25),
            torch.nn.ReLU(),

            # 2nd hidden layer
            torch.nn.Linear(25, 15),
            torch.nn.ReLU(),

            # output layer
            torch.nn.Linear(15, num_classes),
```

```
        )

    def forward(self, x):
        logits = self.all_layers(x)
        return logits
```

1.4 5) Defining a DataLoader

- More details in Unit 4.4

[6]:
```
from torch.utils.data import Dataset, DataLoader

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

        self.features = torch.tensor(X, dtype=torch.float32)
        self.labels = torch.tensor(y, dtype=torch.int64)

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

    def __len__(self):
        return self.labels.shape[0]


train_ds = MyDataset(X_train, y_train)
val_ds = MyDataset(X_val, y_val)
test_ds = MyDataset(X_test, y_test)

train_loader = DataLoader(
    dataset=train_ds,
    batch_size=32,
    shuffle=True,
)

val_loader = DataLoader(
    dataset=val_ds,
    batch_size=32,
    shuffle=False,
)

test_loader = DataLoader(
    dataset=test_ds,
    batch_size=32,
    shuffle=False,
```

```
)
```

1.5 6) The training loop

[7]:
```python
def compute_accuracy(model, dataloader):

    model = model.eval()

    correct = 0.0
    total_examples = 0

    for idx, (features, labels) in enumerate(dataloader):

        with torch.inference_mode(): # basically the same as torch.no_grad
            logits = model(features)

        predictions = torch.argmax(logits, dim=1)

        compare = labels == predictions
        correct += torch.sum(compare)
        total_examples += len(compare)

    return correct / total_examples
```

Training loop

- Similar to Unit 3.6 – Logistic Regression in PyTorch
- Differences are
  - `PytorchMLP` instead of `LogisticRegression` model
  - `F.cross_entropy` instead of `F.binary_cross_entropy`
- Note that F.cross_entropy takes `logits` as inputs, not probabilities
  - it does the one-hot encoding and softmax internally

[8]:
```python
import torch.nn.functional as F

torch.manual_seed(1)
model = PyTorchMLP(num_features=2, num_classes=2)
optimizer = torch.optim.SGD(model.parameters(), lr=0.05) # Stochastic gradient␣
 ↪descent

num_epochs = 10

for epoch in range(num_epochs):

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

```
        logits = model(features)

        loss = F.cross_entropy(logits, labels) # Loss function

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

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

    train_acc = compute_accuracy(model, train_loader)
    val_acc = compute_accuracy(model, val_loader)
    print(f"Train Acc {train_acc*100:.2f}% | Val Acc {val_acc*100:.2f}%")
```

```
Epoch: 001/010 | Batch 000/018 | Train/Val Loss: 0.68
Epoch: 001/010 | Batch 001/018 | Train/Val Loss: 0.69
Epoch: 001/010 | Batch 002/018 | Train/Val Loss: 0.68
Epoch: 001/010 | Batch 003/018 | Train/Val Loss: 0.67
Epoch: 001/010 | Batch 004/018 | Train/Val Loss: 0.69
Epoch: 001/010 | Batch 005/018 | Train/Val Loss: 0.68
Epoch: 001/010 | Batch 006/018 | Train/Val Loss: 0.66
Epoch: 001/010 | Batch 007/018 | Train/Val Loss: 0.68
Epoch: 001/010 | Batch 008/018 | Train/Val Loss: 0.67
Epoch: 001/010 | Batch 009/018 | Train/Val Loss: 0.68
Epoch: 001/010 | Batch 010/018 | Train/Val Loss: 0.66
Epoch: 001/010 | Batch 011/018 | Train/Val Loss: 0.65
Epoch: 001/010 | Batch 012/018 | Train/Val Loss: 0.67
Epoch: 001/010 | Batch 013/018 | Train/Val Loss: 0.67
Epoch: 001/010 | Batch 014/018 | Train/Val Loss: 0.65
Epoch: 001/010 | Batch 015/018 | Train/Val Loss: 0.67
Epoch: 001/010 | Batch 016/018 | Train/Val Loss: 0.67
Epoch: 001/010 | Batch 017/018 | Train/Val Loss: 0.65
Train Acc 68.41% | Val Acc 65.62%
Epoch: 002/010 | Batch 000/018 | Train/Val Loss: 0.66
Epoch: 002/010 | Batch 001/018 | Train/Val Loss: 0.66
Epoch: 002/010 | Batch 002/018 | Train/Val Loss: 0.66
Epoch: 002/010 | Batch 003/018 | Train/Val Loss: 0.65
Epoch: 002/010 | Batch 004/018 | Train/Val Loss: 0.66
Epoch: 002/010 | Batch 005/018 | Train/Val Loss: 0.65
Epoch: 002/010 | Batch 006/018 | Train/Val Loss: 0.63
Epoch: 002/010 | Batch 007/018 | Train/Val Loss: 0.65
Epoch: 002/010 | Batch 008/018 | Train/Val Loss: 0.66
Epoch: 002/010 | Batch 009/018 | Train/Val Loss: 0.66
Epoch: 002/010 | Batch 010/018 | Train/Val Loss: 0.66
```

```
Epoch: 002/010 | Batch 011/018 | Train/Val Loss: 0.63
Epoch: 002/010 | Batch 012/018 | Train/Val Loss: 0.64
Epoch: 002/010 | Batch 013/018 | Train/Val Loss: 0.64
Epoch: 002/010 | Batch 014/018 | Train/Val Loss: 0.65
Epoch: 002/010 | Batch 015/018 | Train/Val Loss: 0.63
Epoch: 002/010 | Batch 016/018 | Train/Val Loss: 0.63
Epoch: 002/010 | Batch 017/018 | Train/Val Loss: 0.62
Train Acc 81.50% | Val Acc 73.44%
Epoch: 003/010 | Batch 000/018 | Train/Val Loss: 0.64
Epoch: 003/010 | Batch 001/018 | Train/Val Loss: 0.63
Epoch: 003/010 | Batch 002/018 | Train/Val Loss: 0.63
Epoch: 003/010 | Batch 003/018 | Train/Val Loss: 0.64
Epoch: 003/010 | Batch 004/018 | Train/Val Loss: 0.62
Epoch: 003/010 | Batch 005/018 | Train/Val Loss: 0.61
Epoch: 003/010 | Batch 006/018 | Train/Val Loss: 0.61
Epoch: 003/010 | Batch 007/018 | Train/Val Loss: 0.64
Epoch: 003/010 | Batch 008/018 | Train/Val Loss: 0.62
Epoch: 003/010 | Batch 009/018 | Train/Val Loss: 0.61
Epoch: 003/010 | Batch 010/018 | Train/Val Loss: 0.61
Epoch: 003/010 | Batch 011/018 | Train/Val Loss: 0.61
Epoch: 003/010 | Batch 012/018 | Train/Val Loss: 0.62
Epoch: 003/010 | Batch 013/018 | Train/Val Loss: 0.60
Epoch: 003/010 | Batch 014/018 | Train/Val Loss: 0.62
Epoch: 003/010 | Batch 015/018 | Train/Val Loss: 0.61
Epoch: 003/010 | Batch 016/018 | Train/Val Loss: 0.60
Epoch: 003/010 | Batch 017/018 | Train/Val Loss: 0.59
Train Acc 89.88% | Val Acc 90.62%
Epoch: 004/010 | Batch 000/018 | Train/Val Loss: 0.59
Epoch: 004/010 | Batch 001/018 | Train/Val Loss: 0.59
Epoch: 004/010 | Batch 002/018 | Train/Val Loss: 0.58
Epoch: 004/010 | Batch 003/018 | Train/Val Loss: 0.59
Epoch: 004/010 | Batch 004/018 | Train/Val Loss: 0.59
Epoch: 004/010 | Batch 005/018 | Train/Val Loss: 0.57
Epoch: 004/010 | Batch 006/018 | Train/Val Loss: 0.55
Epoch: 004/010 | Batch 007/018 | Train/Val Loss: 0.60
Epoch: 004/010 | Batch 008/018 | Train/Val Loss: 0.59
Epoch: 004/010 | Batch 009/018 | Train/Val Loss: 0.57
Epoch: 004/010 | Batch 010/018 | Train/Val Loss: 0.57
Epoch: 004/010 | Batch 011/018 | Train/Val Loss: 0.56
Epoch: 004/010 | Batch 012/018 | Train/Val Loss: 0.58
Epoch: 004/010 | Batch 013/018 | Train/Val Loss: 0.58
Epoch: 004/010 | Batch 014/018 | Train/Val Loss: 0.56
Epoch: 004/010 | Batch 015/018 | Train/Val Loss: 0.57
Epoch: 004/010 | Batch 016/018 | Train/Val Loss: 0.54
Epoch: 004/010 | Batch 017/018 | Train/Val Loss: 0.57
Train Acc 93.19% | Val Acc 90.62%
Epoch: 005/010 | Batch 000/018 | Train/Val Loss: 0.56
Epoch: 005/010 | Batch 001/018 | Train/Val Loss: 0.56
```

```
Epoch: 005/010 | Batch 002/018 | Train/Val Loss: 0.54
Epoch: 005/010 | Batch 003/018 | Train/Val Loss: 0.54
Epoch: 005/010 | Batch 004/018 | Train/Val Loss: 0.52
Epoch: 005/010 | Batch 005/018 | Train/Val Loss: 0.54
Epoch: 005/010 | Batch 006/018 | Train/Val Loss: 0.53
Epoch: 005/010 | Batch 007/018 | Train/Val Loss: 0.50
Epoch: 005/010 | Batch 008/018 | Train/Val Loss: 0.54
Epoch: 005/010 | Batch 009/018 | Train/Val Loss: 0.52
Epoch: 005/010 | Batch 010/018 | Train/Val Loss: 0.50
Epoch: 005/010 | Batch 011/018 | Train/Val Loss: 0.54
Epoch: 005/010 | Batch 012/018 | Train/Val Loss: 0.48
Epoch: 005/010 | Batch 013/018 | Train/Val Loss: 0.53
Epoch: 005/010 | Batch 014/018 | Train/Val Loss: 0.50
Epoch: 005/010 | Batch 015/018 | Train/Val Loss: 0.48
Epoch: 005/010 | Batch 016/018 | Train/Val Loss: 0.49
Epoch: 005/010 | Batch 017/018 | Train/Val Loss: 0.51
Train Acc 95.29% | Val Acc 92.19%
Epoch: 006/010 | Batch 000/018 | Train/Val Loss: 0.47
Epoch: 006/010 | Batch 001/018 | Train/Val Loss: 0.50
Epoch: 006/010 | Batch 002/018 | Train/Val Loss: 0.51
Epoch: 006/010 | Batch 003/018 | Train/Val Loss: 0.48
Epoch: 006/010 | Batch 004/018 | Train/Val Loss: 0.43
Epoch: 006/010 | Batch 005/018 | Train/Val Loss: 0.47
Epoch: 006/010 | Batch 006/018 | Train/Val Loss: 0.44
Epoch: 006/010 | Batch 007/018 | Train/Val Loss: 0.52
Epoch: 006/010 | Batch 008/018 | Train/Val Loss: 0.45
Epoch: 006/010 | Batch 009/018 | Train/Val Loss: 0.47
Epoch: 006/010 | Batch 010/018 | Train/Val Loss: 0.44
Epoch: 006/010 | Batch 011/018 | Train/Val Loss: 0.41
Epoch: 006/010 | Batch 012/018 | Train/Val Loss: 0.50
Epoch: 006/010 | Batch 013/018 | Train/Val Loss: 0.44
Epoch: 006/010 | Batch 014/018 | Train/Val Loss: 0.45
Epoch: 006/010 | Batch 015/018 | Train/Val Loss: 0.41
Epoch: 006/010 | Batch 016/018 | Train/Val Loss: 0.47
Epoch: 006/010 | Batch 017/018 | Train/Val Loss: 0.41
Train Acc 94.76% | Val Acc 92.19%
Epoch: 007/010 | Batch 000/018 | Train/Val Loss: 0.43
Epoch: 007/010 | Batch 001/018 | Train/Val Loss: 0.42
Epoch: 007/010 | Batch 002/018 | Train/Val Loss: 0.42
Epoch: 007/010 | Batch 003/018 | Train/Val Loss: 0.41
Epoch: 007/010 | Batch 004/018 | Train/Val Loss: 0.38
Epoch: 007/010 | Batch 005/018 | Train/Val Loss: 0.41
Epoch: 007/010 | Batch 006/018 | Train/Val Loss: 0.38
Epoch: 007/010 | Batch 007/018 | Train/Val Loss: 0.41
Epoch: 007/010 | Batch 008/018 | Train/Val Loss: 0.42
Epoch: 007/010 | Batch 009/018 | Train/Val Loss: 0.39
Epoch: 007/010 | Batch 010/018 | Train/Val Loss: 0.39
Epoch: 007/010 | Batch 011/018 | Train/Val Loss: 0.38
```

```
Epoch: 007/010 | Batch 012/018 | Train/Val Loss: 0.35
Epoch: 007/010 | Batch 013/018 | Train/Val Loss: 0.37
Epoch: 007/010 | Batch 014/018 | Train/Val Loss: 0.36
Epoch: 007/010 | Batch 015/018 | Train/Val Loss: 0.38
Epoch: 007/010 | Batch 016/018 | Train/Val Loss: 0.38
Epoch: 007/010 | Batch 017/018 | Train/Val Loss: 0.40
Train Acc 96.86% | Val Acc 98.44%
Epoch: 008/010 | Batch 000/018 | Train/Val Loss: 0.39
Epoch: 008/010 | Batch 001/018 | Train/Val Loss: 0.34
Epoch: 008/010 | Batch 002/018 | Train/Val Loss: 0.33
Epoch: 008/010 | Batch 003/018 | Train/Val Loss: 0.37
Epoch: 008/010 | Batch 004/018 | Train/Val Loss: 0.35
Epoch: 008/010 | Batch 005/018 | Train/Val Loss: 0.35
Epoch: 008/010 | Batch 006/018 | Train/Val Loss: 0.37
Epoch: 008/010 | Batch 007/018 | Train/Val Loss: 0.36
Epoch: 008/010 | Batch 008/018 | Train/Val Loss: 0.30
Epoch: 008/010 | Batch 009/018 | Train/Val Loss: 0.32
Epoch: 008/010 | Batch 010/018 | Train/Val Loss: 0.28
Epoch: 008/010 | Batch 011/018 | Train/Val Loss: 0.35
Epoch: 008/010 | Batch 012/018 | Train/Val Loss: 0.32
Epoch: 008/010 | Batch 013/018 | Train/Val Loss: 0.32
Epoch: 008/010 | Batch 014/018 | Train/Val Loss: 0.32
Epoch: 008/010 | Batch 015/018 | Train/Val Loss: 0.30
Epoch: 008/010 | Batch 016/018 | Train/Val Loss: 0.31
Epoch: 008/010 | Batch 017/018 | Train/Val Loss: 0.31
Train Acc 97.03% | Val Acc 100.00%
Epoch: 009/010 | Batch 000/018 | Train/Val Loss: 0.29
Epoch: 009/010 | Batch 001/018 | Train/Val Loss: 0.28
Epoch: 009/010 | Batch 002/018 | Train/Val Loss: 0.24
Epoch: 009/010 | Batch 003/018 | Train/Val Loss: 0.26
Epoch: 009/010 | Batch 004/018 | Train/Val Loss: 0.25
Epoch: 009/010 | Batch 005/018 | Train/Val Loss: 0.28
Epoch: 009/010 | Batch 006/018 | Train/Val Loss: 0.26
Epoch: 009/010 | Batch 007/018 | Train/Val Loss: 0.32
Epoch: 009/010 | Batch 008/018 | Train/Val Loss: 0.30
Epoch: 009/010 | Batch 009/018 | Train/Val Loss: 0.30
Epoch: 009/010 | Batch 010/018 | Train/Val Loss: 0.28
Epoch: 009/010 | Batch 011/018 | Train/Val Loss: 0.28
Epoch: 009/010 | Batch 012/018 | Train/Val Loss: 0.22
Epoch: 009/010 | Batch 013/018 | Train/Val Loss: 0.33
Epoch: 009/010 | Batch 014/018 | Train/Val Loss: 0.33
Epoch: 009/010 | Batch 015/018 | Train/Val Loss: 0.25
Epoch: 009/010 | Batch 016/018 | Train/Val Loss: 0.27
Epoch: 009/010 | Batch 017/018 | Train/Val Loss: 0.27
Train Acc 98.08% | Val Acc 100.00%
Epoch: 010/010 | Batch 000/018 | Train/Val Loss: 0.26
Epoch: 010/010 | Batch 001/018 | Train/Val Loss: 0.23
Epoch: 010/010 | Batch 002/018 | Train/Val Loss: 0.25
```

```
Epoch: 010/010 | Batch 003/018 | Train/Val Loss: 0.22
Epoch: 010/010 | Batch 004/018 | Train/Val Loss: 0.25
Epoch: 010/010 | Batch 005/018 | Train/Val Loss: 0.28
Epoch: 010/010 | Batch 006/018 | Train/Val Loss: 0.22
Epoch: 010/010 | Batch 007/018 | Train/Val Loss: 0.27
Epoch: 010/010 | Batch 008/018 | Train/Val Loss: 0.23
Epoch: 010/010 | Batch 009/018 | Train/Val Loss: 0.19
Epoch: 010/010 | Batch 010/018 | Train/Val Loss: 0.22
Epoch: 010/010 | Batch 011/018 | Train/Val Loss: 0.21
Epoch: 010/010 | Batch 012/018 | Train/Val Loss: 0.21
Epoch: 010/010 | Batch 013/018 | Train/Val Loss: 0.22
Epoch: 010/010 | Batch 014/018 | Train/Val Loss: 0.25
Epoch: 010/010 | Batch 015/018 | Train/Val Loss: 0.28
Epoch: 010/010 | Batch 016/018 | Train/Val Loss: 0.16
Epoch: 010/010 | Batch 017/018 | Train/Val Loss: 0.21
Train Acc 98.25% | Val Acc 100.00%
```

1.6 7) Evaluating the results

[9]:
```python
train_acc = compute_accuracy(model, train_loader)
val_acc = compute_accuracy(model, val_loader)
test_acc = compute_accuracy(model, test_loader)

print(f"Train Acc {train_acc*100:.2f}%")
print(f"Val Acc {val_acc*100:.2f}%")
print(f"Test Acc {test_acc*100:.2f}%")
```

```
Train Acc 98.25%
Val Acc 100.00%
Test Acc 99.12%
```

1.7 8) Optional: visualizing the decision boundary

[10]:
```python
from matplotlib.colors import ListedColormap
import numpy as np


def plot_decision_regions(X, y, classifier, resolution=0.02):

    # setup marker generator and color map
    markers = ('D', '^', 'x', 's', 'v')
    colors = ('C0', 'C1', 'C2', 'C3', 'C4')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
```

```
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    tensor = torch.tensor(np.array([xx1.ravel(), xx2.ravel()]).T).float()
    logits = classifier.forward(tensor)
    Z = np.argmax(logits.detach().numpy(), axis=1)

    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot class samples
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, color=cmap(idx),
                    #edgecolor='black',
                    marker=markers[idx],
                    label=cl)

plot_decision_regions(X_train, y_train, classifier=model)
```