192.151 Introduction to Deep Learning

2025S

Problem Sheet 1

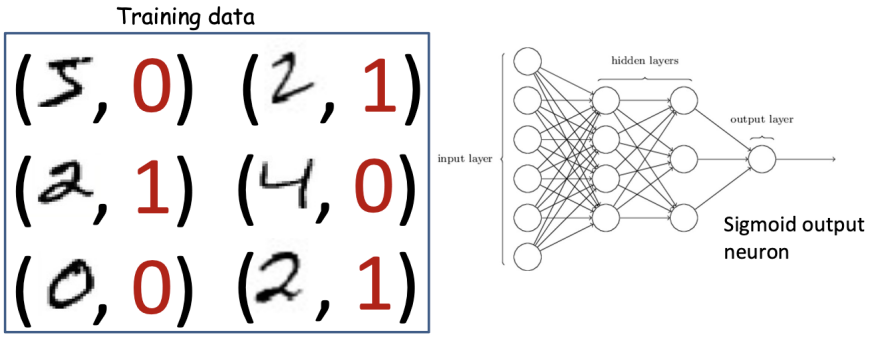
Start with producing the PyTorch code for Problem 1.1, and then reuse most of your code for Problem 1.2.

How to succeed even if you have no coding experience with PyTorch whatsoever:

- Use Google Colab (https://colab.google/) to run your PyTorch code.
- Use the PyTorch official tutorials (https://pytorch.org/tutorials/) for support.
- If you have no coding experience in PyTorch at all, then ask Google Gemini (directly in Google Colab), DeepSeek (https://www.deepseek.com/en), or ChatGPT (https://chatgpt.com/) to do the complete PyTorch coding for you, and to explain to you in detail every line of code. Ask Google Gemini, DeepSeek, or ChatGPT to explain to you anything that you do not understand in the code.
- Otherwise, try to follow the instructions below step by step, using the help of Google Gemini, DeepSeek, or ChatGPT whenever needed, e.g., by asking questions such as "How do I use nn.Sigmoid in PyTorch?" or "How do I evaluate a binary classifier?"

Problem 1.1

Using the MNIST handwritten digits dataset, build a neural network that answers: "Is this digit a 2 or not?":



Instructions:

a) Load the MNIST training and test datasets using torchvision.datasets.MNIST.

b) Preprocess the labels so that:
   - $2 \rightarrow 1$ (positive class),
   - all other digits $\rightarrow 0$ (negative class).

c) Build a simple neural network with:
   - Input size: 784 (flattened 28x28 image),
   - 1 hidden layer with ReLU,
   - 1 output neuron with sigmoid activation.

d) Use Binary Cross Entropy Loss (nn.BCELoss()).

e) Train the model for 10 epochs on the training dataset.

f) Plot the training loss for all epochs.

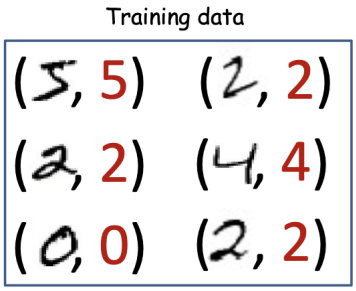
g) Evaluate using accuracy, precision, recall, and F1-score on the testing dataset.

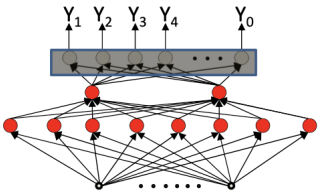
h) Visualize some predictions with actual digits.

Please find also some draft PyTorch code for this exercise in Appendix A.

Problem 1.2

Now upgrade your code from the solution of Problem 1.1 to recognize all digits (0 through 9):





Instructions:

a) Use the same MNIST dataset without modifying the labels.

b) Build a neural network with:

- Input size: 784,
- 2 hidden layers,
- output size: 10 neurons (one for each digit),
- use softmax activation on the output layer.

c) Use CrossEntropyLoss (or KL divergence with log-softmax).

d) Train the model for 20 epochs on the training dataset.

e) Plot the training loss for all epochs.

f) Evaluate using accuracy and confusion matrix on the test dataset.

g) Visualize the predictions on some sample digits.

h) Try some other learning rates and numbers of epochs to see how this changes the results.

Appendix A: PyTorch Code

```
# Digit Recognition with PyTorch (Google Colab Template)
# Task 1: Binary Classification - "Is it a 2?"

import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report

# Set device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Hyperparameters
input_size = 784
hidden_size = 128
num_epochs = 10
batch_size = 64
learning_rate = 0.001

# Preprocess: Flatten + Normalize
transform = transforms.Compose([
  transforms.ToTensor(),
  transforms.Normalize((0.1307,), (0.3081,)),
])

# Custom dataset wrapper for binary classification (2 vs not 2)
def binary_label(label):
  return 1 if label == 2 else 0

class BinaryMNIST(datasets.MNIST):
  def __getitem__(self, index):
     image, label = super().__getitem__(index)
     return image.view(-1), torch.tensor(binary_label(label), dtype=torch.float32)

train_data = BinaryMNIST(root='./data', train=True, download=True,
  transform=transform)
test_data = BinaryMNIST(root='./data', train=False, download=True,
  transform=transform)

train_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_data, batch_size=batch_size)

# Define simple binary classification model
class BinaryClassifier(nn.Module):
  def __init__(self):
    super().__init__()
    self.fc1 = nn.Linear(input_size, hidden_size)
    self.relu = nn.ReLU()
    self.fc2 = nn.Linear(hidden_size, 1)
    self.sigmoid = nn.Sigmoid()

  def forward(self, x):
    x = self.fc1(x)
    x = self.relu(x)
    x = self.fc2(x)
```

```
    return self.sigmoid(x)

model = BinaryClassifier().to(device)
criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

# Train loop
losses = []
for epoch in range(num_epochs):
  total_loss = 0
  for images, labels in train_loader:
     images, labels = images.to(device), labels.to(device).unsqueeze(1)
     outputs = model(images)
     loss = criterion(outputs, labels)

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

     total_loss += loss.item()

   avg_loss = total_loss / len(train_loader)
   losses.append(avg_loss)
   print(f"Epoch {epoch+1}, Loss: {avg_loss:.4f}")

# Plot training loss
plt.plot(losses)
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Training Loss")
plt.show()

# Evaluate binary classifier
model.eval()
all_preds = []
all_labels = []
with torch.no_grad():
  for images, labels in test_loader:
     images = images.to(device)
     outputs = model(images).cpu().numpy()
     preds = (outputs > 0.5).astype(int)
     all_preds.extend(preds.flatten())
     all_labels.extend([binary_label(lbl) for lbl in labels.numpy()])

print(classification_report(all_labels, all_preds, target_names=["Not 2", "2"]))

# Task 2: Multiclass Classification (0{9)
# TODO: You will implement this next!
# Use nn.CrossEntropyLoss and change output layer to nn.Linear(hidden_size, 10)
# Replace sigmoid with softmax in evaluation
# Ask ChatGPT for help when you're stuck!
```