Introduction to Machine Learning

Homework 3
Due: Tuesday, 10/29, 5pm

Instructor:

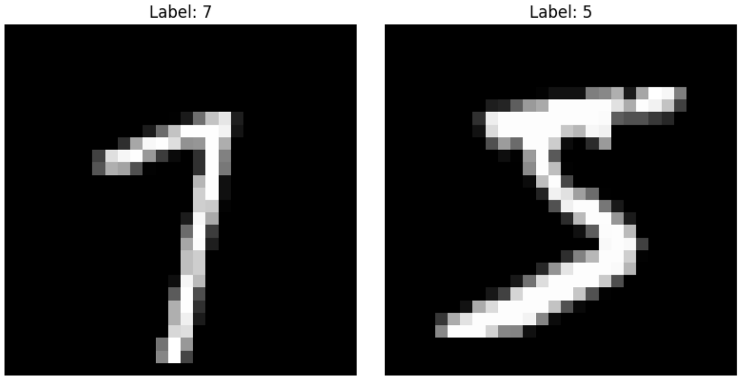
Dr. Tom Arodz

The problem

- Implement and test:
  - Linear classifier for 10 classes
  - With cross entropy loss (over softmax)
  - With L1 regularization (see below)
  - Trained using gradient descent
- Use PyTorch library

Dataset

- MNIST: a 10-class classification problem
    - 28x28 gray-scale images of handwritten digits



- Training set of 50,000 samples (5,000 per class)
- Test set of 10,000 samples (1,000 per class)

```
import torch
from torchvision import datasets, transforms
import matplotlib.pyplot as plt

def show_mnist_image(index):
    # Load MNIST dataset
    mnist_data = datasets.MNIST(
        root='./data', train=True,
        download=True,
        transform=transforms.ToTensor())

    # Get the image and label
    image, label = mnist_data[index]

    # Convert image from PyTorch tensor
    # to numpy array and reshape
    image = image.squeeze().numpy()

    # Display the image
    plt.imshow(image, cmap='gray')
    plt.title(f"Label: {label}")
    plt.axis('off')
    plt.show()
show_mnist_image(42)
```

Your task

- Write a function to train a model on the dataset, and a function to test it on another dataset

```
def my_train(train_dataset)
    #your code
    return W, b

def my_test(W,b,test_dataset)
    #your code (error rate on 0-1 scale)
    return test_error_rate
```

- Your goal for the `my_train` function is to train the model to achieve high accuracy (at least 87.5% accuracy, i.e., 0.125 (or, 12.5%) error rate), while keeping some weights for redundant features at 0
    - You will need to select appropriate hyperparameter (e.g., learning rate, number of epochs, regularization strength)
- Your goal for the `my_test` function is to correctly calculate the error rate, using pytorch functions (incl. DataLoader)

L1 regularization

- L1 regularization is an approach similar to L2 regularization (weight decay)
- It aims at reducing the number of irrelevant features, by bringing their weights to 0

- Loss_with_regularization(W) = CrossEntropy(W) + λ $||W||_1$
  - Λ controls the "strength" of regularization, with λ=0 meaning no regularization, and high λ means strong regularization (more features will become 0)

- $||W||_1$ is L_1-norm, defined as sum of absolute values of all the matrix cells

- $||W||_1 = \mathrm{sum}_{ij} |W[i,j]|$

Your task – variable details

-
  ```
  def my_train(train_dataset)
      #your code
      return W, b

  def my_test(W,b,test_dataset)
      #your code (error rate on 0-1 scale)
      return test_error_rate
  ```

- W, b are be numpy arrays,
  - shape of W: (num_classes, num_features)
  - shape of b: (num_classes,)

- test_error_rate is single number

- train_dataset, test_dataset are of type:
  torchvision.datasets.mnist.MNIST
  loaded as in the next slide

Classification model and training

- Dataset loading that will be used in grading:

```
from torchvision import datasets, transforms

transform = transforms.Compose([
   transforms.ToTensor(),
   transforms.Normalize((0.1307,), (0.3081,))
   ])

train_dataset = datasets.MNIST(root='./data’,
   train=True, download=True, transform=transform)

test_dataset = datasets.MNIST(root='./data’,
   train=False, download=True, transform=transform)
```

Model and objective function

- The classification model should be:
  - A linear model (W^Tx+b)
    - shape of W: (num_classes,num_features)
    - shape of b: (num_classes,)
    - num_features should be 784

      (28x28 pixels, each one grayscale value)
  - It is ok to use nn.Linear, but observe the shape of their internal parameter (.weight and .bias) and convert the shape if it doesn't match the above specs for input/output of your functions my_train and my_test

- The objective function should be cross-entropy loss + L1 regularization
  - note whether it should take raw predictions (a.k.a. logits, from [-infinity,+infinity] range), or probabilities ([0,1] range)

Returning the Assignment

- Solution code should be written by you and you only (no web/book/friend/etc. code)

- Upload through Canvas/Gradescope
  - Similar to Homework 1 & 2
  - A single file with your two functions
    - Do not forget to do all the necessary imports
    - If your code doesn't "compile" or throws an exception, gradescope will fail, with 0 points
    - It is advisable to either delete any of your testing code, or "guard" it with:

```
if __name__ == "__main__":
```