

# Statistical and Machine Learning (01.113)

## Homework 2

Due on 7 MAR, 3 PM

### Problem 1 (2 points)

The entropy of a discrete probability distribution, which is always greater than or equal to zero, is given by

$$\text{Ent}(p) = - \sum_{i=1}^n p_i \log p_i, \quad \sum_{i=1}^n p_i = 1.$$

- (a) Use Lagrange multipliers to find the probability distribution which maximizes entropy.
- (b) Which probability distribution minimizes entropy?

### Problem 2 (2 points)

Let  $A$  be an  $n \times n$  matrix,  $B$  be an  $n \times p$  matrix,  $C$  be a  $p \times n$  matrix and  $D$  be a  $p \times p$  matrix. Show that

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{bmatrix},$$

where

$$M = (A - BD^{-1}C)^{-1}.$$

### Problem 3 (1 point)

Derive a formula relating  $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$  and the sigmoid function  $\sigma(x) = \frac{1}{1 + e^{-x}}$ .

### Problem 4 (5 points)

We will use PyTorch to train a Convolutional Neural Network (CNN) to improve classification accuracy on the Fashion MNIST dataset. This dataset comprises 50,000 training examples and 10,000 test examples of 28x28-pixel monochrome images of various clothing items. Let us begin by importing the libraries:

```
import numpy
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
import matplotlib.pyplot as plt
```

There are a total of 10 classes enumerated in the following way:

```
labels = {
    0 : "T-shirt",
    1 : "Trouser",
    2 : "Pullover",
    3 : "Dress",
    4 : "Coat",
    5 : "Sandal",
    6 : "Shirt",
    7 : "Sneaker",
    8 : "Bag",
    9 : "Ankle_boot"
}
```

(a) Define your model by inheriting from the "nn.Module" using the following format:

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        # initialize layers here

    def forward(self, x):
        # invoke the layers here

    return ...
```

(b) Complete the main function below; the test and train functions will be defined in later parts.

```
def main():
    NUMEPOCHS = # Complete here
    LRATE = # Complete here

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

    train_dataset = datasets.FashionMNIST('../data', train=True,
        download=True, transform=transforms.ToTensor())
    test_dataset = datasets.FashionMNIST('../data', train=False,
        download=True, transform=transforms.ToTensor())

    ##### Use dataloader to load the datasets
    train_loader = # Complete here
    test_loader = # Complete here

    model = CNN().to(device)
    optimizer = optim.SGD(model.parameters(), lr=LRATE)

    for epoch in range(1, NUMEPOCHS + 1):
        test(model, device, test_loader)
        train(model, device, train_loader, optimizer, epoch)

    test(model, device, test_loader)

if __name__ == '__main__':
    main()
```

- (c) Complete the training function by defining the model output and the loss function. Use the optimizer's `step()` function to update the weights after backpropagating the gradients. (Remember to clear the gradients with each iteration.)

```
def train(model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)

        # Fill in here

        if batch_idx % 100 == 0:
            print('Epoch:', epoch, ', loss:', loss.item())
```

- (d) In the test function, define the variable "pred" which predicts the output, and update the variable "correct" to keep track of the number of correctly classified objects so as to compute the accuracy of the CNN.

```
def test(model, device, test_loader):
    model.eval()
    correct = 0
    exampleSet = False
    example_data = numpy.zeros([10, 28, 28])
    example_pred = numpy.zeros(10)

    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)

            # fill in here

            if not exampleSet:
                for i in range(10):
                    example_data[i] = data[i][0].to("cpu").numpy()
                    example_pred[i] = pred[i].to("cpu").numpy()
                exampleSet = True

    print('Test set accuracy: ', 100. * correct / len(test_loader.dataset), '%')

    for i in range(10):
        plt.subplot(2,5,i+1)
        plt.imshow(example_data[i], cmap='gray', interpolation='none')
        plt.title(labels[example_pred[i]])
        plt.xticks([])
        plt.yticks([])
    plt.show()
```

You must achieve more than 80% accuracy to get full credit. Upload your final script in your Dropbox folder and name it as "HW2.py".