40.319 STATISTICAL AND MACHINE LEARNING
SPRING 2020 HOMEWORK 2

DUE 26 FEB. TOTAL 40 POINTS.

Submit your written/typed solutions as a PDF using Gradescope.
Upload your Python files using this link.

`https://www.dropbox.com/request/I93CAZNMfLUDQssVmz0Q`

1. Entropy [5 Points]

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

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

1.1. Use Lagrange multipliers to find the distribution which maximizes entropy.

1.2. Which probability distribution minimizes entropy?

2. Schur Complement [5 Points]

Let A be an n x n matrix, B be an n x p matrix, C be a p x n matrix and D be a p x 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

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

Date: February 12, 2020.

3. Convolutional Networks [15 Points]

We will use PyTorch to train a Convolutional Neural Network (CNN) to improve classification accuracy on the Fashion MNIST dataset. This dataset comprises 60,000 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"
  }
```

3.1. 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 ...
```

3.2. Complete the main function below; the test and train functions will be defined later.

```
def main():
  N_EPOCH = # Complete here
  L_RATE = # 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=L_RATE)

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

  test(model, device, test_loader)

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

3.3. 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())
```

3.4. 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.

Append the print-outs from your program (test accuracy and plots of images with their predicted labels) to your PDF submission on Gradescope. Upload the final script as a file named `[student-id]-cnn.py` using the Dropbox link at the start of this assignment.

4. Support Vector Machines [15 Points]

In this problem, we will implement Support Vector Machines (SVMs) for classifying two datasets. We start by importing the required packages and modules.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from sklearn.svm import SVC
from sklearn.datasets.samples_generator import make_blobs, make_circles
```

The `make_blobs` and `make_circles` functions from `sklearn.datasets` can be invoked to generate data for the first and second example, respectively.

The following will be used to plot decision boundaries, margins and support vectors.

```
def plot_svc_decision(model, ax=None):
  if ax is None:
    ax = plt.gca()
  xlim = ax.get_xlim()
  ylim = ax.get_ylim()

  # create grid to evaluate model
  x = np.linspace(xlim[0], xlim[1], 30)
  y = np.linspace(ylim[0], ylim[1], 30)
  Y, X = np.meshgrid(y, x)
  xy = np.vstack([X.ravel(), Y.ravel()]).T
  P = model.decision_function(xy).reshape(X.shape)

  # plot decision boundary and margins
  ax.contour(X, Y, P, colors='k', levels=[-1, 0, 1],
             alpha=0.5, linestyles=['--', '-', '--'])

  # plot support vectors
  ax.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1],
             s=300, linewidth=1, edgecolors='black', facecolors='none')
  ax.set_xlim(xlim)
  ax.set_ylim(ylim)
```

4.1. Use the following lines of code to plot the first dataset.

```
X, y = make_circles(100, factor=.1, noise=.1)
fig1 = plt.figure()
ax1 = fig1.add_subplot(111)
ax1.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='seismic')
```

Use `SVC` to construct a support vector machine (you will have to specify a kernel and the regularization parameter C) to classify this dataset, then use `fit(X, y)` to feed in the data and labels. Show your results using the `plot_svc_decision` function. Provide one graph, labelled with your choice of kernel function and your value of C.

4.2. Now generate and plot the second dataset.

```
X, y = make_blobs(n_samples=100, centers=2,
                  random_state=0, cluster_std=1.0)
fig2 = plt.figure(figsize=(16, 6))
fig2.subplots_adjust(left=0.0625, right=0.95, wspace=0.1)

ax2 = fig2.add_subplot(121)
ax2.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='seismic')

ax3 = fig2.add_subplot(122)
ax3.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='seismic')
```

Your task here is to classify the dataset using different values of the regularization parameter C to understand soft margins in SVM. Indicate clearly what values of C you are using, and plot your results with `plot_svc_decision` using `ax2` for one model and `ax3` for the other.

Append the plots from your program to your PDF submission on Gradescope. Upload the final script as a file named `[student-id]-svm.py` using the Dropbox link at the start of this assignment.