Machine Learning for Computer Vision

Exercise 11

Kodai Matsuoka, Yuyan Li

July 14, 2017

Our program is found in listing 1

1 Forward Pass

Our neural network is given by

$$z = sigmoid(\sin(\phi) \cdot x_0 + \cos(\phi) \cdot x_1 + r).$$

We do a binary classification with the probabilities

$$p(y = 0) = z \quad \text{and} \quad p(y = 1) = 1 - z$$

For the loss we use cross entropy, The loss of output z with ground truth y is

$$loss(z, y) = y \log(z) + (1 - y) \log(1 - z).$$

2 Fischer Matrix

The entries of the Fischer matrix are given by

$$F_{a,b} = \sum_{x \in X} \sum_{y \in \{0,1\}} \left(\frac{\partial}{\partial a} \log p(y|x) \right) \left(\frac{\partial}{\partial b} \log p(y|x) \right) p(y|x)$$

We need the derivatives of $z(x; \phi, r)$ with respect to ϕ and r.

$$\frac{\partial z}{\partial \phi} = (x_0 \cos(\phi) - x_1 \sin(\phi)) \cdot sigmoid'(\sin(\phi) \cdot x_0 + \cos(\phi) \cdot x_1 + r)$$

$$\frac{\partial z}{\partial r} = sigmoid'(\sin(\phi) \cdot x_0 + \cos(\phi) \cdot x_1 + r)$$

The derivative of the sigmoid is given by:

$$sigmoid'(x) = sigmoid(x) \cdot (1 - sigmoid(x))$$

With these we can compute the Fischer matrix entries:

$$F_{\phi\phi} = \sum_{x \in X} \frac{\frac{\partial z}{\partial \phi}^2}{z(1-z)}$$

$$F_{rr} = \sum_{x \in X} \frac{\frac{\partial z}{\partial r}^2}{z(1-z)}$$

$$F_{\phi r} = \sum_{x \in X} \frac{\frac{\partial z}{\partial \phi}\frac{\partial z}{\partial r}}{z(1-z)}$$

We plot the loss and the Fischer matrizes as heat maps for various ϕ and r in fig. 1.

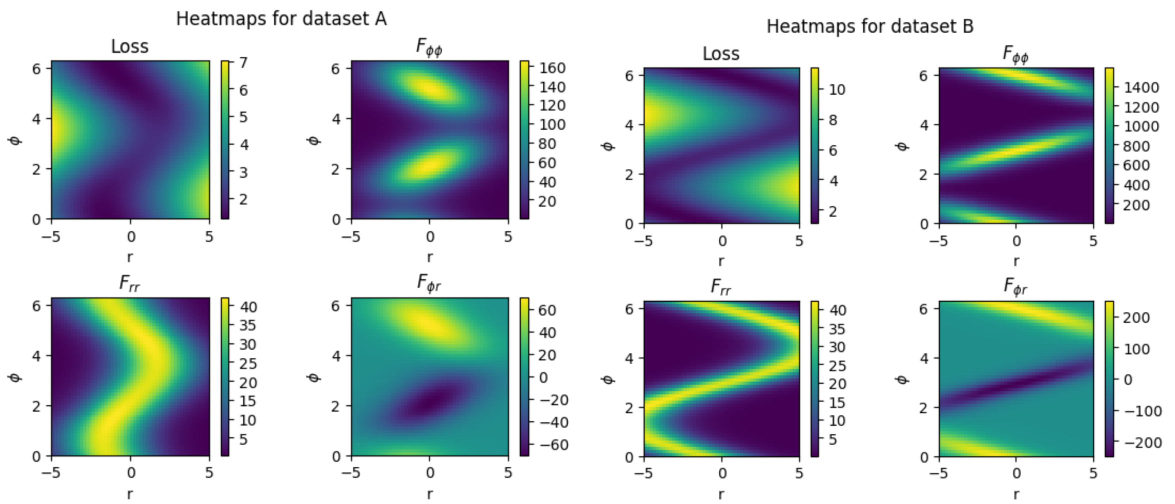

Figure 1: The heatmaps show the values of the loss and the Fischer matrix entries of the two datasets A (left) and B (right). The plots are made with N=M=50 and a sample size for each class of $N_{data} = 100$

Listing 1: fischer.py

```
import numpy as np
import matplotlib.pyplot as plt


# Sigmoid function
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# Derivative of the sigmoid
def sig_deriv(x):
    return sigmoid(x) * (1 - sigmoid(x))

# Our neural network
def nn(x, phi, r):
    return sigmoid(np.sin(phi) * x[:,0] + np.cos(phi) * x[:,1] + r)

# Derivative of nn
def derivative(x, phi, r):
    return sig_deriv(np.sin(phi) * x[:,0] + np.cos(phi) * x[:,1] + r)

# Derivative dz/dphi
def dzdp(x, phi, r):
    return (np.cos(phi) * x[:,0] - np.sin(phi) * x[:,1]) * derivative(x,
        phi, r)

# Derivative dz/dr
def dzdr(x, phi, r):
    return derivative(x, phi, r)

def fischerPP(x, phi, r):
    z = nn(x, phi, r)
    f = dzdp(x, phi, r)**2 / (z * (1 - z))
    return sum(f)

def fischerRR(x, phi, r):
    z = nn(x, phi, r)
    f = dzdr(x, phi, r)**2 / (z * (1 - z))
    return sum(f)

def fischerPR(x, phi, r):
    z = nn(x, phi, r)
    f = dzdp(x, phi, r) * dzdr(x, phi, r) / (z * (1 - z))
    return sum(f)


# Cross entropy loss function, prediction z and ground truth y
def loss(z, y):
    l = y * np.log(z) + (1 - y) * np.log(1 - z)
    return - sum(l) / l.size

# ==========================================================
```

```
# === Make heat maps of Loss and Fischer Matrix Entries ===
# ==========================================================
def makefig(NData, mean, N, M, dataset=""):
    Cov = [[1,0],
           [0,1]]
    # Make sample data
    class1 = np.random.multivariate_normal(mean[0], Cov, NData)
    class2 = np.random.multivariate_normal(mean[1], Cov, NData)


    phi_list = np.linspace(0, 2*np.pi, N)
    r_list = np.linspace(-5.0, 5.0, M)


    # Losses
    l1 = np.zeros((N,M))
    l2 = np.zeros((N,M))
    # Fischer matrix entries
    fpp = np.zeros((N,M))
    frr = np.zeros((N,M))
    fpr = np.zeros((N,M))

    for ip, phi in enumerate(phi_list):
        for ir, r in enumerate(r_list):
            z1 = nn(class1[:,], phi, r)
            z2 = nn(class2[:,], phi, r)
            l1[ip, ir] = loss(z1, 0.0)
            l2[ip, ir] = loss(z2, 1.0)

            fpp[ip, ir] = fischerPP(class1[:,], phi, r) + fischerPP(class2
                [:,], phi, r)
            frr[ip, ir] = fischerRR(class1[:,], phi, r) + fischerRR(class2
                [:,], phi, r)
            fpr[ip, ir] = fischerPR(class1[:,], phi, r) + fischerPR(class2
                [:,], phi, r)
    # Total loss
    l = l1 + l2


    # Make heatmaps
    results = [l, fpp, frr, fpr]
    titles = ["Loss", "$F_{\phi \phi}$", "$F_{rr}$", "$F_{\phi r}$"]

    fig = plt.figure()
    for i, res in enumerate(results):
        plt.subplot(2, 2, i+1)
        plt.imshow(res, origin='lower', extent=[-5, 5, 0, 2*np.pi], aspect
            =5/np.pi, interpolation='nearest')
        plt.colorbar()
        plt.title(titles[i])
        plt.xlabel("r")
        plt.ylabel("$\phi$")
```

```
    fig.suptitle(f"Heatmaps for dataset {dataset}")
    fig.subplots_adjust(wspace=0.3, hspace=0.5)

    return fig

if __name__=="__main__":
    N = 50
    M = 50
    NData = 100
    meanA = [[1, 1], [1, 2]]
    meanB = [[6, 1], [6, 2]]

    figA = makefig(NData, meanA, N, M, "A")
    figA.savefig("heatmaps_a.png", bbox_inches='tight')
    figB = makefig(NData, meanB, N, M, "B")
    figB.savefig("heatmaps_b.png", bbox_inches='tight')
```