DLSS - Assignment 01

April 16, 2024

1 Deep Learning for Social Sciences

1.1 Assignment 01

To be handed in by 7.5.2024 23:59 at https://github.com/orgs/DLSS-24/.

These materials were developed by Max Pellert building in parts on work by Simon J.D. Prince.

1.1.1 Task 01.01 (14 points)

Do a literature research to find a research paper that fulfills the following criteria:

- There is an application in the social sciences (broadly)
- The authors use neural nets for a substantial part of their analysis, in particular they rely on a method from the domain of deep learning
- The work is at maximum three years old (has been published after the 15th of April 2021)
- It was published in a journal, as a conference publication or as pre-print (for example on arXiv)

Provide a few paragraphs (1/2 page each, ~250 words) of text covering those three points (1 1/2 pages in total, ~750 words):

- In one sentence, state your intial motivation to choose the paper. Then provide a review of the work, starting with a short summary, then identfying key strengths as well as weaknesses and finally provide an overall assessment of the quality of the work.
- Do background research on the specific method from the domain of neural nets and deep learning specifically that the authors used. Identify and read at least one of the key references connected to the specific method and cite it. Give a short characterisation of the connection of that method to the current state of the art of deep learning and try to place that method historically in the field. Mention if the method is in any way specific to the problem that it was used for by the authors and reconstruct the rationale behind the choice of exactly that method.
- Review the publication outlet of your work. If it was published in a journal, describe the focus of that journal and do a quick search in the last published editions of that journal to assess if they regularly publish works with a connection to deep learning there. If it was published in conference proceedings, do a background research on the conference and situate it in the broader field it belongs too. If the work was published as pre-print, try to find if it was published in other form in another outlet. If not, use the metadata on the pre-print platform (as for example the category the work was placed in) to characterise it further. In any case,

for all different forms of publication outlets, do a background research on the authors and provide a short assessment of their fields, research groups and typical research topics.

1.1.2 Task 01.02 (6 points)

In this task, you will explore the linear regression model discussed in Chapter 2 of the book "Understanding Deep Learning". Equation and Figure numbers refer to this book. Work through the cells below, running each cell in turn. Wherever you see the words "TO DO", follow the instructions at these places and write code to complete the functions. These code completions in those places will be the basis for our evaluation of your performance on this task.

[ ]:
```
# Math library
import numpy as np
# Plotting library
import matplotlib.pyplot as plt
```

[ ]:
```
# Create some example input / output data
x = np.array([0.03, 0.19, 0.34, 0.46, 0.78, 0.81, 1.08, 1.18, 1.39, 1.60, 1.65,␣
 ↪1.90])
y = np.array([0.67, 0.85, 1.05, 1.0, 1.40, 1.5, 1.3, 1.54, 1.55, 1.68, 1.73, 1.
 ↪6 ])

print(x)
print(y)
```

[ ]:
```
# Define 1D linear regression model
def f(x, phi0, phi1):
  # TODO :  Replace this line with the linear regression model (eq 2.4)
  y = x

  return y
```

[ ]:
```
# Function to help plot the data
def plot(x, y, phi0, phi1):
    fig,ax = plt.subplots()
    ax.scatter(x,y)
    plt.xlim([0,2.0])
    plt.ylim([0,2.0])
    ax.set_xlabel('Input, $x$')
    ax.set_ylabel('Output, $y$')
    # Draw line
    x_line = np.arange(0,2,0.01)
    y_line = f(x_line, phi0, phi1)
    plt.plot(x_line, y_line,'b-',lw=2)

    plt.show()
```

[ ]:
```
# Set the intercept and slope as in figure 2.2b
phi0 = 0.4 ; phi1 = 0.2
# Plot the data and the model
plot(x,y,phi0,phi1)
```

[ ]:
```
# Function to calculate the loss
def compute_loss(x,y,phi0,phi1):

  # TODO Replace this line with the loss calculation (equation 2.5)
  loss = 0

  return loss
```

[ ]:
```
# Compute the loss for our current model
loss = compute_loss(x,y,phi0,phi1)
print(f'Your Loss = {loss:3.2f}, Ground truth = 7.07')
```

[ ]:
```
# Set the intercept and slope as in figure 2.2c
phi0 = 1.60 ; phi1 =-0.8
# Plot the data and the model
plot(x,y,phi0,phi1)
loss = compute_loss(x,y,phi0,phi1)
print(f'Your Loss = {loss:3.2f}, Ground truth = 10.28')
```

[ ]:
```
# TO DO -- Change the parameters manually to fit the model
# First fix phi1 and try changing phi0 until you can't make the loss go down␣
 ↪any more
# Then fix phi0 and try changing phi1 until you can't make the loss go down any␣
 ↪more
# Repeat this process until you find a set of parameters that fit the model as␣
 ↪in figure 2.2d
# You can either do this by hand, or if you want to get fancy, write code to␣
 ↪descent automatically in this way
phi0 = 1.60 ; phi1 =-0.8

plot(x,y,phi0,phi1)
print(f'Your Loss = {compute_loss(x,y,phi0,phi1):3.2f}')
```

Extra: Visualizing the loss function

If you did everything correctly until here, we can now plot a part of the full space that you were just manually traversing on:

[ ]:
```
# Make a 2D grid of possible phi0 and phi1 values
phi0_mesh, phi1_mesh = np.meshgrid(np.arange(0.0,2.0,0.02), np.arange(-1.0,1.
 ↪0,0.02))

# Make a 2D array for the losses
```

```
all_losses = np.zeros_like(phi1_mesh)
# Run throught each 2D combination of phi0, phi1 and compute loss
for indices,temp in np.ndenumerate(phi1_mesh):
    all_losses[indices] = compute_loss(x,y, phi0_mesh[indices],␣
 ↪phi1_mesh[indices])
```

[ ]:
```
# Plot the loss function as a heatmap
fig = plt.figure()
ax = plt.axes()
fig.set_size_inches(7,7)
levels = 256
ax.contourf(phi0_mesh, phi1_mesh, all_losses ,levels)
levels = 40
ax.contour(phi0_mesh, phi1_mesh, all_losses ,levels, colors=['#80808080'])
ax.set_ylim([1,-1])
ax.set_xlabel('Intercept, $\phi_0$')
ax.set_ylabel('Slope, $\phi_1$')

# Plot the position of your best fitting line on the loss function
# It should be close to the minimum
ax.plot(phi0,phi1,'ro')
plt.show()
```