

Codelet 3: Softmax Regression using PyTorch

The due date for this codelet is **Wednesday, Feb 12 at 11:59PM**.

Introduction

The aim of this codelet is to build on your PyTorch skills via the concrete implementation of parts of softmax regression. You should draw on Chapter 4 of the textbook and the PyTorch documentation. You may, and likely should, build on your code from codelet2.

Important: Only use PyTorch, sklearn, matplotlib, torchvision, and in-built Python. The use of any other libraries, including the books d2l library results in an automatic unsatisfactory grade for this assignment. A core goal of this class is to build your competencies as a machine learning engineer. I want to minimize abstractions from other libraries so that you build these skills.

Outline

- ☐ Grading
- ☐ Softmax Regression
 - ☐ Model
 - ☐ Validation Loss
 - ☐ Light error analysis

Your assignment

Your task is to:

1. Download `codelet3.zip` from the course website and open it. You will find these instructions and `codelet3.py` which has scaffolding for you.
2. Complete each of the 3 broad tasks below in `codelet3.py` and include a file called `codelet3.pdf` with your answers to the written questions.

Grading

When assessing your work, satisfactory achievement is demonstrated, in part, by:

- Simple and clear code
- Code utilizes PyTorch methods in a way that simplifies your code
- Code passes relevant tests
- Explanation extends beyond repeating the textbook/material online
- Response are concrete and clearly demonstrate an understanding of the concepts

Softmax Regression

You will complete an implementation of softmax regression trained using gradient descent. You should build on `codelet3.py`. You have three tasks:

1. Complete the model class
2. Implement validation loss
3. Conduct some light error analysis

Softmax Regression Model

No basic class structure is provided. You should instead adapt yours from `codelet2.py`. Carefully review the provided train function when consider your implementation. Like with `codelet2`, you must use PyTorch's Linear layer to handle your model's weights (and compute your model's output). Additionally, you must use PyTorch's Softmax in creating your output (which should be the probability of each label for each sample).

To evidence completion of this task, and to check your own progress, please train your model using the `train` method. The `train` method should work without any tweaks. Do not modify or otherwise change that function. You should add to the pdf accompanying your code a screenshot of the output of your code. Mine, for example, looks as follows (skipping parts in the middle):

```
batch 100 loss: 1.672284483909607
batch 200 loss: 1.1067351633310318
batch 300 loss: 0.9611724829673767
batch 400 loss: 0.8704170614480973
batch 500 loss: 0.8233233571052552
batch 600 loss: 0.7921148812770844
batch 700 loss: 0.7688001942634582
batch 800 loss: 0.734564779996872
batch 900 loss: 0.6996732714772225
Epoch 1 Train Avg Loss 0.929627036304077 Validation Loss -1
```

```
-----
batch 100 loss: 0.6923972260951996
batch 200 loss: 0.6773819029331207
batch 300 loss: 0.6719656646251678
batch 400 loss: 0.6548320659995079
batch 500 loss: 0.6613071784377098
batch 600 loss: 0.6577168375253677
batch 700 loss: 0.6382966065406799
batch 800 loss: 0.6172605124115944
batch 900 loss: 0.6231981575489044
Epoch 2 Train Avg Loss 0.6551079480027185 Validation Loss -1
```

```
...
...
-----
```

```
batch 100 loss: 0.49717589110136035
batch 200 loss: 0.5047877758741379
batch 300 loss: 0.48548407793045045
batch 400 loss: 0.48455618917942045
batch 500 loss: 0.4776347289979458
batch 600 loss: 0.47942177444696427
batch 700 loss: 0.4911013525724411
batch 800 loss: 0.4988816994428635
batch 900 loss: 0.48646573185920716
Epoch 10 Train Avg Loss 0.49039147889664997 Validation Loss -1
```

Implement Validation Loss

Notice that the `valid_loss` function returns -1 in the provided code. Your task is to complete this function. It should meet the following specifications:

- Return the average loss per batch for the validation data
- Not calculate a gradient

Notice, PyTorch automatically calculates the gradient when generating predictions. You should research how to disable this for a whole function or for a block of code (either way works).

To evidence completion of this task, and to check your own progress, please train your model using the `train` function. You should add to the pdf accompanying your code a screenshot of the output of your code. Mine, for example, looks as follows (ignoring the remaining epochs):

```

batch 100 loss: 1.6615215480327605
batch 200 loss: 1.099790757894516
batch 300 loss: 0.9497145491838456
batch 400 loss: 0.8745188522338867
batch 500 loss: 0.8291540437936783
batch 600 loss: 0.7907151341438293
batch 700 loss: 0.7683601039648056
batch 800 loss: 0.7313806009292603
batch 900 loss: 0.7262751519680023
Epoch 1 Train Avg Loss 0.928412394119047 Validation Loss 0.7187230858453519
-----

```

...

Light Error Analysis

To further evaluate your model's performance, you should complete two things:

1. Calculate the accuracy, precision, recall, and f1 score for you model on your validation data (both before training and after training; you should use sklearn)
2. Compare the average probability your model assigns to it's top prediction when it is correct to when it is incorrect.
3. Plot a few of your model's incorrect predictions. Do you see any pattern?

To evidence completion of this task, and to check your own progress, please provide screenshots of the output of 1 and 2 in the pdf accompanying your code. Mine, for example, look like:

Initial metrics:

Acc 0.0625 P 0.07361111111111111 R 0.04892156862745098 F1 0.038225108225108224

Final metrics:

Acc 0.78125 P 0.7961111111111111 R 0.7908730158730158 F1 0.7836813186813186

Average probability of correct predictions: 83.2%

Average probability of incorrect predictions: 57.67%

For 3, include images of your plots and a description of any patterns you see or explanation you may have for your model's mislabeling. Notice, a plotting function has been provided with the class for the data.