AIML - CS 335

# Lab 4: Intro to Classification with PyTorch

25 August 2022

*Lecturer: Abir De*

**Important:** Please read the instructions mentioned in the questions carefully. We have provided boilerplate code for each question. Please ensure that you make changes in the areas marked with TODO.

Please **read the comments in the code carefully** for detailed information regarding input and output format.

# 1 Classification

## 1.1 Dataset

You have been given synthetic dataset for classification. The dataset contains 2000 data points, each of which has 80 features. the shape of the dataset is 2000*81, where the last column indicates the output label (since its a binary classification problem output is 0/1).

Keep **dataset.csv** in the same folder as assignment.py

You will have to read the given **dataset.csv** file, convert it to a pandas dataframe and then split the entire dataset into training and validation data as per the split ratio for evaluation purpose.

## 1.2 Task

Complete the following functions:

- `def __init__(self, args, input_dim):`
  This is the constructor. It takes as input the args, input_dim variables. Use the args variable to specify all the hyperparameters of the model, and the neural layers as per your discretion.

- `def forward(self, data):`
  This implements the forward pass of the algorithm.
  Input data is batched data of shape BATCH_SIZE * num_features
  Returns predictions of shape BATCH_SIZE * 1

- `def loss(self, pred, label):` This implements the loss function. The inputs pred and label, are each tensors of shape BATCH_SIZE*1 . Pred contains model prediction

outputs for each of the batched inputs. Label contains 0/1 values only. Function returns a single loss value. We will be implementing different variations of the loss function as described in the next section.

- `def evaluate(loader, model):` This implements the evaluation function for binary classification. Input loader is of "torch.data_utils.DataLoader" type. It is an ordered pair where the first item is the feature matrix of shape BATCH_SIZE * num_features, and the second item is the label tensor of shape BATCH_SIZE * 1. Function returns an evaluation score based on the following definition: **it will be ratio of the number of samples which are correctly classified/total number of sample points in that batch**

We will be be implementing the three loss functions as specified next. While executions, we need to specify the type of loss function using the args variable "model_type".

### 1.2.1 Negative Log Likelihood (NLL)

The Binary Cross Entropy Function is defined as follows, for a set of N data points:

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N} t_i \log(p_i) + (1 - t_i) \log(1 - p_i)$$

Here, for the $i$th data point: $t_i$ is the true label (0 for class 0 and 1 for class 1) and $p_i$ is the predicted probability of the point belonging to class 1.

When the observation belongs to class 1 the first part of the formula becomes active and the second part vanishes, and vice versa in the case observation's actual class are 0. This is how we calculate the binary cross-entropy.

The probabiltiy scores for the forward pass can be computed using a Sigmoid function as follows:

$$S(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{e^z + 1} = 1 - S(-z)$$

**$z$ is the score of the item $x$ as given by the neural model.**

The sigmoid function outputs a $S(z) \in [0, 1]$ and indicates the probability of how close to a class the item belongs (in the case of binary classification). Therefore, having a threshold 0.5, the binary classification output $Class(x)$ can be formulated as

$$Class(x) = \begin{cases} 1 & \text{if } S(z) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Your model will output a sigmoid score for each input. Subsequently, these real valued predictions will be converted to binary labels using $Class(x)$ function. Finally, the **accuracy** is computed using the binary predictions and binary labels, and is defined as $\frac{\text{no. of label matches}}{\text{total no. of items in both labels}}$

### 1.2.2 SVM Loss

For $i$th data point, the Hinge Loss Function is given by:

$$L_{HL}(x_i, y_i) = \begin{cases} 1 - y_i(w^T x_i + b) & \text{if } 1 - y_i(w^T x_i + b)) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where, $y_i$ is the truth label for the corresponding input instance $x_i$ and parameter $w$.
Here: $w, x_i \in \mathbb{R}^n$, $b \in \mathbb{R}$, $y_i \in \{-1, 1\}$

**Note: you will have to change the output labels as (1,-1) rather than (1,0) as given in the dataset before calculating the hinge loss**.

Here, $L_{HL}$ can take any float value depending on the sign of $y_i$ and $w^T x_i$. If both signs are same (indicating correct class prediction), $L_{HL} = 0$. And if the signs are opposite (indicating misclassification), value of $L_{HL}$ increases. In other words, it finds the classification boundary that guarantees the maximum margin between the data points of the different classes.

Hence during evaluation, you need to calculate the number of items on each side of the classification boundary w.r.t to the actual classes they belong to. The formula is given as follows:

$$Class(x_i) = \begin{cases} 1 & \text{if } y_i(w^T x_i + b) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

As before, the final **accuracy** is computed using the binary predictions of $Class(x)$ and binary labels, and is defined as $\frac{\text{no. of label matches}}{\text{total no. of items in both labels}}$

### 1.2.3 Ranking Loss

The Ranking Loss is defined as follows (recap from earlier lab):

$$L(P, N) = \sum_{p \in P, n \in N} max(0, n - p)$$

Here, P denotes the model predictions for the set of positive items labelled 1. N denotes the model predictions for the set of negative items labelled 0. We want to impose the constraint that the positive scores should be greater than the negative scores. This is enforced by the above functions.

Your model will output a score for each input. During Evaluation, we will count the number of times positive items are scored higher than the negative ones. Higher the count, better is the model. The counting formula is specified as below:

$$L(P, N) = \sum_{p \in P, n \in N} \mathbb{1}[p > n]$$

The final **evaluation** score is the output of the counting formula.

# 2 Other Resources:

We have provided the following resources to help you train your models:

## 2.1 Training

```
def train(args, Xtrain, Ytrain, Xval, Yval, model)
```

Use this as a black box function to train your code. The inputs are the args variable, training labels and features, validation features and labels, and the model object. **Do not change any part of this function**.

## 2.2 Visualization

```
def plot(val_accs, losses):
```

Use this function to plot the training loss, and validation accuracy across epochs.

# 3 Assessment

We will be checking the following:

- End-to-End working of the ML model
- correctness of the loss function implementations
- correctness of the evaluation function implementations
- execution time of the all functions.
  **Make sure to avoid for loops in the loss functions**.

# 4 Submission instructions

Complete the functions in `assignment.py`. Make changes only in the places mentioned in comments. Do not modify the function signatures. Keep the file in a folder named `<ROLL_NUMBER>_L4` and compress it to a tar file named `<ROLL_NUMBER>_L4.tar.gz` using the command

```
tar -zcvf <ROLL_NUMBER>_L4.tar.gz <ROLL_NUMBER>_L4
```

Submit the tar file on Moodle. The directory structure should be -

```
<ROLL_NUMBER>_L4
| - - - - assignment.py
```

Replace `ROLL_NUMBER` with your own roll number. If your Roll number has alphabets, they should be in *"small"* letters.