CS 484 - CS 555: Introduction to Computer Vision
Spring 2020

Dr. Sedat Ozer
Homework 2: Image Classification

Due: March 25, 2020.

Instructions

- Please first read all the instructions and this entire document from beginning till the end.
- You will **submit a single python script** including all the methods for each step below (a single <YourName>_<YourID>_PA2.py file where you will replace <YourName> with your own particular full name and <YourID> with your current student ID). For each step, you will write a different method within the same python script.
- In addition to that, you will also **submit a single report** in PDF format and save it as: <YourName>_<YourID>_PA2_Report.pdf. Your report will include all the figures and/or the output values of your individual python methods as asked in each part of this assignment below.
- If you use the same code section/snippet in multiple sections, then copy and paste that code in each method individually so that that method can run by itself when copied into another file.
- Each method should be able to run by itself and be able to produce the asked results. Error producing codes may not receive credit.
- At the end of this assignment, you should compress all your files: .py file and your report file (.pdf) into a single <YourName>_<YourID>_PA2.zip file. Email your zip file to your TA (Mr. Mirzayev) with the subject: "CS484_CS555_HW2_Submission".
- Also, please be careful with your report and pay attention to its format before submitting it. Not only your results, but the quality and aesthetics of your report will also affect your final grade.
- You will implement this assignment using only **Python** (Assignments using MATLAB will not be graded for this assignment). You cannot use any code snippet or functions from Keras, TensorFlow or PyTorch in Part 1 and in Part2. You are also NOT allowed to use any code snippet from internet.
- Contact the TA: *aydamir.mirzayev@bilkent.edu.tr*, if you have any question.
- Not complying with the submission rules will result in 10 points penalty.

The Goal

The main goals of this assignment are listed below:

- Be able to implement a basic $NeuralNetwork$ with single neuron for image classification by yourself without using libraries such as PyTorch or Tensorflow.
- Familiarize yourself with essential classification concepts such as *loss function, cost function, training, testing, parameters, training data, test data and labels.*
- Compute optimal parameters and familiarize yourself with Keras for binary classification of images.

1 Dataset

The dataset that you will use in this assignment is the cat identification dataset. The data set contains two classes: cat images and noncat images. The entire data set comes in two files (one file contains all the training images and another file contains all the test images). The training set ($train_catvnoncat.h5$) contains 209 RGB images with 64x64 resolution and test set ($test_catvnoncat.h5$) contains 50 RGB images of the same size. You can use the code snippet below to extract the images from each file:

```
import h5py

test = h5py.File('test_catvnoncat.h5', 'r')
test_set_x = test['test_set_x']
test_set_y = test['test_set_y']

train = h5py.File('train_catvnoncat.h5', 'r')
train_set_x = train['train_set_x']
train_set_y = train['train_set_y']
```

2 Assignment

For computation and for practice (if needed), you can use Google Colab. In Parts 1 and 2, you are NOT allowed to use Keras, Tensorflow or PyTorch. In Part 3, you need to use KERAS.

2.1 Part 1

You will implement a binary classifier using logistic regression. In this part, you are NOT allowed to use KERAS or Tensorflow or PyTorch (you should use NumPy).

the input to your classifier will be RGB image and the binary output will be the class label (whether the given image is cat or not).

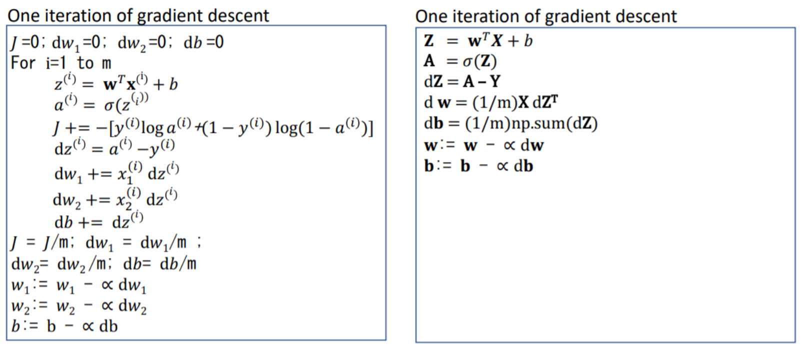
You will implement the pseudo-code that is described in the lecture slides: in Part1 and Part2 of Introduction to Deep Learning section for training. That pseudo-code is also given in Figure 1 for your convenience). There are two pseudo-codes given there: one for each image separately, and one for multiple images (the matrix form).

You are asked to implement both of those above-mentioned pseudo codes individually. You can use NumPy library (but you are NOT allowed to use KERAS, PyTorch, TensorFlow or any other similar library for this part).

- Use all the training data to train your neuron and to learn the optimal parameters. You need to train your algorithm for 50 epoch (you can set learning rate to 0.0001 initially for this step). Also report the accuracy.
- Find the optimal learning rate that works best for the training data. You need to train your algorithm for 50 epoch. Report the accuracy for that optimal learning rate as well.
- Remember, the total number of epochs to be used, the initial values of w vector and b, the learning rate and other similar hyper-parameters has to be decided before training the algorithm. (Note: when you may need to go through more than one epoch, you will need to add additional loop to the pseudo-code given in the lecture slides).

2.2 Part 2

In this part, you will start with the implementation that you used in the previous step. But this time, you will **find the best hyper-parameters**: the best epoch number top stop training, the value of the best learning rate to obtain the best parameters yielding the best accuracy. In this part, you are NOT allowed to use KERAS or Tensorflow or PyTorch (you should use NumPy).



One iteration of gradient descent

```
$J$ =0; $dw_1$=0; $dw_2$=0; $db$ =0
For i=1 to m
        $z^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)} + b$
        $a^{(i)} = \sigma(z^{(i)})$
        $J \mathrel{+}= -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$
        $dz^{(i)} = a^{(i)} - y^{(i)}$
        $dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$
        $dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$
        $db \mathrel{+}= dz^{(i)}$
$J = J/m$; $dw_1 = dw_1/m$ ;
$dw_2 = dw_2/m$; $db = db/m$
$w_1 := w_1 - \propto dw_1$
$w_2 := w_2 - \propto dw_2$
$b := b - \propto db$
```

One iteration of gradient descent

```
$\mathbf{Z} = \mathbf{w}^T \boldsymbol{X} + b$
$\mathbf{A} = \sigma(\mathbf{Z})$
$d\mathbf{Z} = \mathbf{A} - \mathbf{Y}$
$d\,\mathbf{w} = (1/m)\mathbf{X}\, d\mathbf{Z}^\mathbf{T}$
$d\mathbf{b} = (1/m)$np.sum($d\mathbf{Z}$)
$\mathbf{w} := \mathbf{w} - \propto d\mathbf{w}$
$\mathbf{b} := \mathbf{b} - \propto d\mathbf{b}$
```

Figure 1: Backpropogation algorithms provided in the course slides. Iterative approach (left), Matrix operation approach (right)

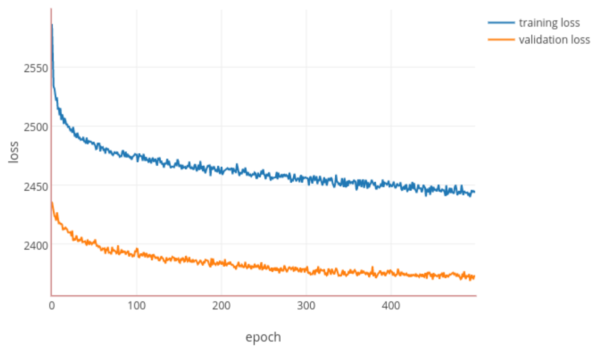


Figure 2: Plot showing how training and test loss (cost) values change at each epoch

You need to **plot "Cost vs. epoch"** for both training and test data sets separately for each learning rate value that you use (show the plots for at least 3 different learning rate values). See Figure 2 for an example. Remember that in the literature, the terms loss and cost are being used interchangeably referring to the cost function. (Note: However, as we studied in the class, both terms: loss and cost also have different meanings in general. In this assignment, we will assume that they both refer to cost value). To do that, during the training, compute the cost at each epoch, and generate epoch vs. loss (cost) plot at the end of the training. Epoch vs. loss plot should display how the value of your cost function changes at every epoch. (See example plot in Figure 2). Note that you should record loss for both, training and test sets, however, only training data should be used in the training process.

At the end of training, you are required to report a set of hyper-parameters yielding the best weights and the bias term on the training data that achieves best classification results in terms of accuracy. Those hyper-parameters that you should report are: the total number of epochs, the learning rate and total number of parameters that you trained. Please also explain why you decided to use those particular hyper-parameter values. Also indicate how you initialized your weights.

2.3 Part 3

In this part, you are allowed to (and you will need to) use KERAS. For computation, if needed, you can use Google Colab.

- First, download and run the example python file from the given link below:

https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py
That particular model, uses a simple ConvNet model (let's call this model as: the Basic ConvNet Model) and trains it on the MNIST dataset for classification. You will use this file (mainly the model) as template in this assignment. However, you will need to change the data part in that template to use our cat identification dataset.

- Your first task: edit that template code to train it on our cat vs. noncat dataset and simply write down (report) the accuracy value you obtained on the test data after 12 epochs in your report file. You need to make the minimal change in the code for this task. Report your changes in your report and the accuracy result that you obtain from the code.

- Your next task is understanding the used network model in the given code. Write down the network model (the architecture) from the code in your report. Remember the architectures that we studied in the class such as: VGG-16, LeNet, AlexNet and how each of those architectures were summarized in the slides. You can use one of those formats to model the network used in the code.

  Here is an example for describing a simple model in the text form: Conv ===> Pooling ===> FC ===> softmax. Make sure you also list the number of filters, filter dims, number of units and activation functions for each layer (where applicable). As an alternative, you can draw a figure illustrating the network architecture as well (preferred). You do not need to submit a code for this task.

- Now, let's have a look at how changing epoch value affects the network's performance. Change the epoch (iteration) value to 30 in the code and plot both training and test accuracy vs. epoch. There are many ways to plot that in python. You will use Tensorboard in this part to plot that (hint: check keras.callbacks.TensorBoard()). Tensorboard can plot your loss function automatically. Include your plot in your report. Plot both the training & test (validation) losses vs. epoch over 30 epochs.

- The original code template that you downloaded from GitHub uses "Adadelta" as the default optimization algorithm. Let's see how Stochastic Gradient Descent would perform in this model instead of Adadelta. Change the optimizer to stochastic gradient descent (SGD) as shown below in your current network implementation. Plot the accuracy vs. epoch over 30 epochs for both training and testing data. Also plot the loss (cost) vs. epoch over 30 epochs (use Tensorboard for that). You can use the code snippet given below for setting the optimizer to SGD. (Optional: You are welcome to use additional parameters for SGD, if you prefer.) include your plot in your report file.

```
sgd = keras.optimizers.SGD(lr=0.001)
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=sgd,
              metrics=['accuracy'])
```

- We know that learning rate is also an important hyper-parameter. Let's study its effect here. Use 5 random (but meaningful) learning rates for SGD and train your network at each of those 5 random learning rates separately on our cat dataset. Plot your accuracy vs. epoch results over 50 epochs for each of those learning rate values in a for-loop (over the 5 different learning rates). Which learning rate value yields the best result? Include your chosen 5 learning rate values and their corresponding plots in your report.

- By looking at all your results that you obtained in this assignment, comment on which optimization algorithm (AdaDelta or SGD) worked better in your case. What did you observe by changing the learning rate to any of those 5 values in the previous step: were all of those 5 learning rates useful, is using a smaller or a higher value better for the learning rate in SGD? Did you notice any trade-off between the learning rate and convergence? If so, please describe it briefly. (In some plots, you should see that if you included more iterations/epochs, you would get even better accuracy while in others you may notice that you have already reached to a saturated point where including more iterations does not help). You do not need to submit your code for this step. Answer the questions in your report.