PROGRAMMING ASSIGNMENT 5

TAs : Bahar GEZİCİ, Burcu YALÇINER
Due Date : 31.12.2021 (23:00)

1 Introduction

In this experiment, you will implement a 2 Layer Neural Network to solve a real world machine learning problem with a real world data. In this experiment, you will get familiar Python data science libraries such as; Pandas, NumPy, and Matplotlib. At the end, you will build a deep neural network that classifies cat vs. non-cat images.

2 Assignment

Packages

- numpy is the fundamental package for scientific computing with Python.

- matplotlib is a library to plot graphs in Python.

- h5py is a common package to interact with a dataset that is stored on an H5 file.

- PIL and scipy are used here to test your model with your own picture at the end.

- np.random.seed(1) is used to keep all the random function calls consistent. It will help us grade your work.

In this assignment, you will build a 2 neural network to solve a real world problem.

3 Part 1. Reading Data

You will use the same "Cat vs non-Cat" dataset. You are given a dataset ("data.h5") containing

- a training set of m train images labelled as cat (1) or non-cat (0)

- a test set of m test images labelled as cat and non-cat

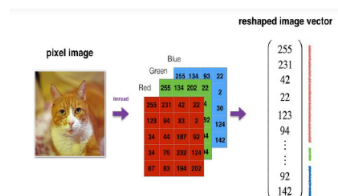- each image is of shape (num px, num px, 3) where 3 is for the 3 channels (RGB).

Figure 1: Image to vector conversion.

You will reshape and standardize the images before feeding them to the network.

4 Part 2. Implementing a 2-layer Neural Network

Neural networks are simply modeled after the human neuron. Just as the brain is made up of numerous neurons, Neural Networks are made up of many activation units. In this assignment you will build a 2-layer neural network. The architecture of the neural network is given in Figure 2.
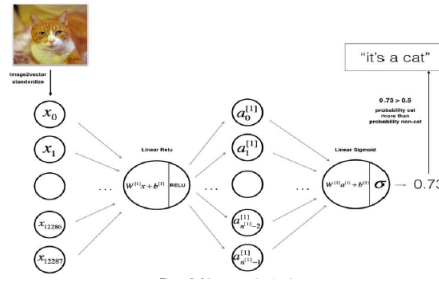

Figure 2: 2-layer neural network.

The model can be summarized as: INPUT - LINEAR - RELU - LINEAR - SIGMOID - OUTPUT.

Detailed Architecture of Figure 2:

- The input is a (64,64,3) image which is flattened to a vector of size (12288,1).

- The corresponding vector: $[x_0, x_1, ..., x_{12287}]^{[T]}$ is then multiplied by the weight matrix $W^{[1]}$ of size $(n^{[1]}; 12288)$.

- You then add a bias term and take its relu to get the following vector: $[a_0^{[1]}, a_1^{[1]}, ..., a_{n^{[1]}-1}^{[1]}]^{[T]}$

- You then repeat the same process.

- You multiply the resulting vector by $W^{[2]}$ and add your intercept (bias).

- Finally, you take the sigmoid of the result. If it is greater than 0.5, you classify it to be a cat.

5 General Methodology

To build the model you will follow the steps given below:

1. Initialize parameters / Define hyperparameters

2. Loop for num iterations:

   - Forward propagation

   - Compute cost function

   - Backward propagation

   - Update parameters (using parameters, and grads from backprop)

3. Use trained parameters to predict labels

Forward propagation

After initialized your parameters, you will do the forward propagation module. You will apply LINEAR and ACTIVATION where ACTIVATION will be either ReLU or Sigmoid. The forward propagation module computes the following equations:

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A = RELU(Z^{[1]}) = max(0, Z^{[1]})$$

$$Z^{[2]} = W^{[1]}A + b^{[2]}$$

$$AL = \sigma(Z^{[2]}) = \frac{1}{1 + e^{-Z^{[2]}}}$$

where X is the input, $Z^{[1]}$ is the output of the first LINEAR, A is the output of the Relu ACTIVATION, $Z^{[2]}$ is the output of the second LINEAR, and $\sigma(AL)$ is the output of the neural network. The details of the activation functions are given below:

$Sigmoid : \sigma(Z) = \sigma(WA + b) = \frac{1}{1+e^{-(WA+b)}}$

$RELU : A = RELU(Z) = max(0, Z)A = RELU(Z) = max(0, Z).$

Cost Function

You need to compute the cost function, because you want to check if your model is actually learning.

Compute the cross-entropy cost J, using the following formula:

$$-\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\log(AL^{(i)}) + (1 - y^{(i)})\log(1 - AL^{(i)}))$$

where m is the number of samples in your training set, y^i is the ith output and $AL^{(i)}$ is the output of the neural network for ith sample.

Backward Propagation

You have already calculated the derivative $dZ^{[l]} = \frac{\partial L}{\partial Z^{[l]}}$. You want to get $(dW^{[l]}, db^{[l]}, dA^{[l-1]})$. The formulas you need: The three outputs $(dW^{[l]}, db^{[l]}, dA^{[l-1]})$ are computed using the input $dZ^{[l]}$.Here are the formulas you need:

$$dW^{[2]} = \frac{\partial J}{\partial W^{[2]}} = \frac{1}{m}dZ^{[2]}A^T$$

$$db^{[2]} = \frac{\partial J}{\partial b^{[2]}} = \frac{1}{m}\sum_{i=1}^{m}dZ^{[2](i)}$$

$$dW^{[1]} = \frac{\partial J}{\partial W^{[2]}} = \frac{1}{m}dZ^{[2]}A^T$$

$$db^{[1]} = \frac{\partial J}{\partial b^{[1]}} = \frac{1}{m} \sum_{i=1}^{m} dZ^{[1](i)}$$

Update Parameters

$$W^{[2]} = W^{[2]} - \alpha dW^{[2]}$$
$$b^{[2]} = b^{[2]} - \alpha db^{[2]}$$
$$W^{[1]} = W^{[1]} - \alpha dW^{[1]}$$
$$b^{[1]} = b^{[1]} - \alpha db^{[1]}$$

where α is the learning rate.

Plot Loss
In your training part, in each iteration you get cost value and append to costs array. You will plot loss figure with this array.

Calculate Accuracy
In test set you will use the linear activation forward function to calculate the accuracy. As mentioned above, if the output value is greater than 0.5, then your prediction is 1 and else is 0.

Accuracy will be calculated as; $accuracy = TruePositiveCount(TP)/TotalNumber of Samples$

TP = is an outcome that model predicts the true class correctly.

For instance; the predictions matrix and label matrices are for 4 sample;

predictions matrix; $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ label matrix; $\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ TP count=2 and Total Number of Samples=4

Accuracy = 2/4 = 50%

You will fill the functions with the given parameters given in the below:

$def\ initialize\_parameters(n\_x, n\_h, n\_y):$
...
$return\ parameters$

$def\ linear\_activation\_forward(X, parameters):$
...
$return\ AL$

$def\ compute\_cost(AL, y):$
...
$return\ cost$

$def\ linear\_activation\_backward(X, cost):$
...
$return\ grads$

$def\ update\_parameters(parameters, grads, learning\_rate):$

$...$

$return\ parameters$

$def\ predict(test\_x, test\_y, parameters):$

$...$

$return\ accuracy$

$def\ plot\_loss(costs):$

$...$

$def\ two\_layer\_model(X, Y, layers\_dims, learning\_rate, num\_iterations):$

$...$

$return\ parameters, costs$

$def\ main():$

$...$

Inputs to a unit is the features from training set. Weights $W^{[1]}$ and $W^{[2]}$; $b^{[1]}$ and $b^{[2]}$ are the parameters that you will learn. Activation functions (RELU and Sigmoid] modify the input data in order to learn complex relations between input and output data.

In Neural Networks, you forward propagate to predict output and compare it with the real value and calculate the loss. Loss value tells us how good our model make its predictions which tells us how to change our parameters to make our model predict more accurate.

In order to minimize the loss, you propagate backwards with respect to each weight value by finding the derivative of the error. Thus, weights will be tuned. Neural networks are trained iteratively. After each iteration error is calculated via the difference between prediction and target.

You are expected to plot loss in each iteration.

Activation Functions

Sigmoid functions:

$$h_\theta(x) = \frac{1}{1 + e^x}$$

The derivative of the Sigmoid function is:

$$\frac{\partial h_\theta(x)}{\partial x} = x * (1 - x)$$

Relu function:

$$h_\theta(x) = \begin{Bmatrix} x & x > 0 \\ 0 & x <= 0 \end{Bmatrix}$$

The derivative of the Relu function is;

$$\frac{\partial h_\theta(x)}{\partial x} = \begin{Bmatrix} 1 & x > 0 \\ 0 & x < 0 \end{Bmatrix}$$

Helper Links

- *Transpose. https://chortle.ccsu.edu/VectorLessons/vmch13/vmch13$_1$4.html*

- *Transpose. https://numpy.org/doc/stable/reference/generated/numpy.transpose.html*

- *Dot product. https://www.mathsisfun.com/algebra/vectors-dot-product.html*

- *Dot product. https://numpy.org/doc/stable/reference/generated/numpy.dot.html*

6   Grading

Task	Point
initialize parameters	2
linear activation forward	25
compute cost	15
linear activation backward	25
update parameters	5
predict	10
plot loss	5
two layer model	10
main	3

7   Important Notes

- You will use the given code for the assignment. Please fill the functions given in the code.

- Do not miss the submission deadline.

- Save all your work until the assignment is graded.

- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating. You can ask your questions via Piazza and you are supposed to be aware of everything discussed on Piazza. You cannot share algorithms or source code. All work must be individual! Assignments will be checked for similarity, and there will be serious consequences if plagiarism is detected.

- You must submit your work with the file as stated below:

  assignment5.ipynb