# How to write code for a Lambda function (Ref. workshop-code-2)

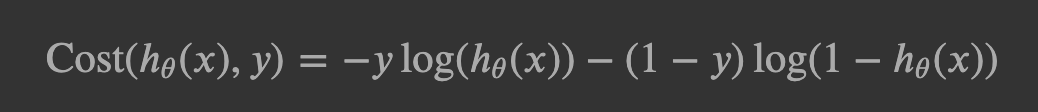
Pre-Reqs:

1. AWS CLI installed and configured.
2. You should have Python 3.x installed and should be able to install any other packages/modules if required.
3. Have access to github
4. Prefer that the participant have their own AWS account to work with, if not, they should be able to create IAM roles, create lambda functions, access a Cloud9 environment if required.

Clone the repo : github….

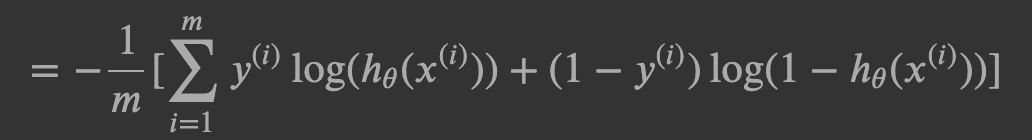
## Python

We will be deploying a Python 3.x lambda function that takes a file, processes it and produces an output.



Courtesy : https://www.internalpointers.com/post/cost-function-logistic-regression

Can be re-written as:



Courtesy : https://www.internalpointers.com/post/cost-function-logistic-regression

This refers to when there are m examples for which the cost has to be computed.

You should have a file with you called inputdata.zip, this will contain the input data you need to perform the computation, this data will be in : w.npz, b.npz,X.npz, Y.npz

Copy the inputdata.zip file to an S3 location of your choice, note the lambda functions you create need to have read as well as write access to that location. Make use of the lambda execution role to get access to that location.

Create a Lambda Execution Role that has authorization to the following:

* Read and Write access to the above Amazon S3 location.
* Should be able to write to cloudwatch logs.
* Should be able to launch stepfunction state machines

To accomplish these:

1. Login to the AWS console
2. Go to the IAM console
3. Click on Roles.
4. Click on Create role
5. In the next page, choose AWS Service, and then choose Lambda. Click on Next: Permissions
6. Choose the following policies: AmazonS3FullAccess, CloudWatchLogsFullAccess, StepFunctionsFullAccess and AWSXrayFullAccess. \*\*These liberal policies are only for the purpose of this workshop and not a recommended best practice. The best practice is to grant authorization on the basis of the principle of least privilege.
7. Additionally, for this exercise, perform the following steps,
   1. In the IAM console, go to Policies
   2. Click on Create policy
   3. Choose the JSON tab and copy and paste the following policy,

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "PublishLayers",

"Effect": "Allow",

"Action": [

"lambda:PublishLayerVersion"

],

"Resource": "arn:aws:lambda:\*:\*:layer:\*"

},

{

"Sid": "ManageLayerVersions",

"Effect": "Allow",

"Action": [

"lambda:GetLayerVersion",

"lambda:DeleteLayerVersion"

],

"Resource": "arn:aws:lambda:\*:\*:layer:\*:\*"

}

]

}

* 1. Give it a name with your initials <initials>-Layer-Maker-Policy, provide a description i.e. this will enable a user to create and manage lambda layers and then click on Create policy.
  2. Click on Users in your IAM console, choose your user, click on Add inline policy, in the next page choose Import managed policy, search for the policy you just create in point d above and click Import.

1. You should find your python files in the cloned repo, by the name mylayer.py & lambda\_handler.py. The mylayer.py file has your layer code and lambda\_handler.py has your actual lambda function code.
2. Open a Cloud9 terminal, or a terminal of your choice.
3. Create a virtual environment

virtualenv myenv

1. Activate the environment

cd myenv;source ./bin/activate

1. Install the required libraries

pip install numpy

pip install pandas

1. If you are looking to see where your particular package got installed, use the following command,

pip show <package\_name>

1. Deactivate the environment

deactivate

1. Go back to the parent directory of the myenv directory and make a directory named ‘python’.

mkdir python

cp -r <location shown from `pip show <package>` output>/\* ./python

cp mylayer.py ./python/

zip -r my-layer-package.zip ./python/

1. Add the function code file i.e. lambda\_handler.py in our case to the deployment package.

zip -g lambda\_function.zip mylayer.py

1. Now, create the layer, like so:

aws lambda publish-layer-version --layer-name propagation-layer --description "A layer that can do forward propagation" --compatible-runtimes "python3.6" --zip-file fileb://my-layer-package.zip

Check if your layer is now available,

aws lambda list-layers --compatible-runtime "python3.6"

1. Now, create the function,

aws lambda create-function --function-name testfunc1 --runtime python3.6 --role arn:aws:iam::<***AWS Account number***>:role/lambda\_basic\_execution --handler lambda\_handler.handler --zip-file fileb://lambda\_function.zip

If you later want to change the code that the lambda function uses,

aws lambda update-function-code --function-name testfunc --zip-file fileb:// lambda\_function.zip

1. Now, let’s get our lambda function ‘testfunc1’ to use the layer we created.

Go to the Lambda console, click on testfunc1. Click on layers (you should see this right below the box in which is written the Lambda function name), choose to add a layer. In the next page choose custom layers, you should see the layer you created earlier. Choose the layer, choose the appropriate version and click Add.

1. Now let’s test the function with layers. Click on the test button on the top right corner and set up the test like so:

Give a name to the test you are about to configure, set up the JSON like so:

{

"bucket": "rns-lk",

"key": "lambda\_input/inputdata.zip"

}

You can see that this maps to the expected input for your function. Save this test, and execute and see how your lambda function performs.