Autoencoder

Omkar Kaptan
ASU ID: 1209385070
Email: omkar.kaptan@asu.edu

Gerard Lawrence Pinto
ASU ID: 1209619577
Email: gpinto3@asu.edu

***Abstract*—This report summarizes the implementation of an Autoencoder. A summary of the work done - namely the current implementation progress, challenges, future work, time line and references.**

I. Introduction to Autoencoders

Autoencoder is a type of neural network for unsupervised learning. In this learning approach, inferences from an unlabeled training dataset are used to train a learning model. A set of 1 or more hidden layers are used to learn a compressed representation (encoding) of the input data. Typically, this is a lower dimensional representation of the input data. The objective of an Autoencoder is to learn to produce an encoding for a given input which will best describe the input as closely as possible.

II. Project Details

This project requires implementation of an Autoencoder from scratch that could learn a lower dimensional representation of 32x32 images of human faces. This implementation is to be done without the use of any existing neural net libraries. Our efforts are directed towards an implementation that could work with an arbitrary number of layers and neurons per layer.

The major components involved in the implementation of an Autoencoder are described below.

A. Neural Net Structure and Weight Matrices

The basic components of a neural net are:

- Neuron: This is the basic unit of a neural network which is characterized by an activation function. The neuron is fired by producing an output when an input value is provided to the activation function.
- Layer and weight matrix: A layer is composed of one or more neurons. The neurons in two consecutive layers are connected by weighted edges between a given layer and its next layer. The weighted edges between 2 layers can be viewed as a weight matrix which is an MxN matrix where M is the number of neurons in a given layer and N is the number of neurons in its subsequent layer. A weight matrix is shown below.

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

The layers in an Autoencoder have a specific organization. The first layer is called an input layer which is the input to the neural network. The last layer is called the output layer which has the same dimension as that of the input layer. This layer attempts to reconstruct the input from the encoding that the neural net generates so that it can be compared with the original input to evaluate the performance of the encoder.

B. Activation Function

The neuron in a layer is a computational unit that takes inputs and provides an output based on an activation function. The activation functions are non linear in nature and typically have a derivative value which is simple to calculate during backpropagation. There are many activation functions like sigmoid or tanh.

The sigmoid function is chosen for this project which is represented as:

$$f(n) = \frac{1}{1 + e^{-x}} \tag{1}$$

C. Forward Feed

The forward feed of the neural network is the first step in training an Autoencoder. In this step, a batch of training images passes through the neural network. Each image produces an output which is measured against the original image to check accuracy of the Autoencoder.

In this step, each neuron in an arbitrary layer receives an input from the output of the previous layer along with a bias input, each multiplied with a weight which indicates the importance given to this value while providing it as an input to the neuron. Thus, a cumulative input to a given layer is easily calculated by taking a dot product of the output of its previous layer with the input weight matrix for the layer in consideration. The activation function of each neuron in the layer is then fired for the cumulative input provided to it to get an output for this layer which is then propagated to the next layer. In this way, the input is propagated all the way through the neural net to produce an output.

D. Error Function

The error function is used to evaluate the accuracy with which the Autoencoder can reconstruct the original input from the encoding. This error is calculated by comparing the result produced by the output layer with the original input provided to the input layer. A backpropagation algorithm is used to reduce the overall error of the neural net so that the reconstruction is as close to the input as possible. The error

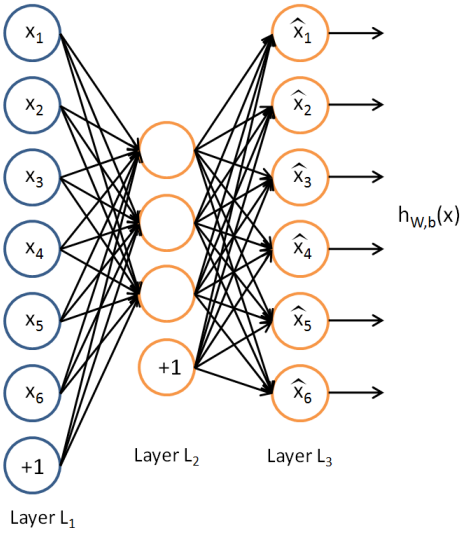

Fig. 1. Simulation of an Autoencoder

function chosen for this project is the root mean squared error function which is represented as:

$$MSE = \frac{1}{n}\sum_{1}^{n}(input - output)^2 \tag{2}$$

E. Back Propagation

An Autoencoder needs to be trained to produce a lower dimensional encoding which will represent the original input as closely as possible. After completing a forward feed for a batch input, the error function will provide a measure of how far off the neural network is from the input. The back propagation phase will use this error measure to update the weight matrices in the neural net in a direction such that the weight change will help reduce the error. To achieve this, a partial differential has to be calculated of the error with respect to the weight at each layer. This can be calculated by using the chain rule for partial derivatives.

III. Implementation Details

This section of the report describes about the technical implementation including design and architecture in building the Autoencoder.

A. Languages and libraries

The language was chosen as Python for this project. The numpy library was extensively used for performing mathematical operations like dot products or matrix operations for example. The PyUnit framework is used to write unit tests for this project.

B. Design

An object oriented approach is chosen to design the Autoencoder. An object is created for the neural net and for each entity of the neural network, namely, a layer and the weight matrix. This approach allows for a flexible and modular design. It is possible to initialize a neural network containing any number of layers and any number of neurons per layer. The design of the neural net is made inherently extensible by appropriate use of python features and design principles. For example, the activation function and error function to be used by the neural network follow a common signature and can be passed to the neural network as a parameter. Along with this, several utility functions have been implemented to handle common tasks like file handling and image handling.

C. Implementation Progress

The present implementation of the Autoencoder has a working neural network which can perform a single pass including a feedforward operation and a back propagation operation on a single image input without any programming error. However, the implementation of backpropagation has some issues which are being resolved at present. The bias has also been accounted for in this implementation.

IV. Problems faced

One of the first challenges faced was to understand how a neural network and Autoencoder works. Since neither member in the team had studied neural nets in the past, the first step was to do that. Several online resources and video lectures were referred to get a clear understanding of how a neural net works. It was easy to understand the principles of an Autoencoder once the functioning of a neural network was clear.

The second challenge that is being dealt with is understanding the subtleties of the backpropagation algorithm. This is key in designing a dynamically sized Autoencoder.

Another challenge being faced presently is the validation of the results obtained when an image is passed.

V. Next steps and time line

Although a single pass can be performed on a single image, the learning phase is not complete yet. After perfecting the implementation of the backpropagation algorithm, the next steps in the implementation are to add batch processing, learning rate and an epoch. Measures will then be taken to improve the accuracy of the Autoencoder and obtain the required visualizations for the trained neural net. The implementation is expected to complete by November 25th.

VI. Distribution of Responsibilities

1) Together: A majority of the learning and design phases have been done as a pair. Once that was clear, the implementation responsibilities were distributed.

2) Omkar Kaptan: Neural net design and data structures, feed forward algorithm.

3) Gerard Lawrence Pinto: Backpropagation, utility functions, activation functions, error functions

VII. Conclusion

In this document a description of the present understanding of Autoencoders, its design and implementation approach, data structures and functions has been provided.

REFERENCES

[1] http://morse.uml.edu/Activities.d/CDM/docs/neuralnets.pdf
[2] http://ufldl.stanford.edu/wiki/index.php/Neural_Networks
[3] http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/
[4] https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/
[5] https://www.coursera.org/learn/machine-learning/lecture/1z9WW/backpropagation-algorithm