Applied Machine Learning SEE Lab Evaluation

Date: 12th May 2016

Image Classification using Auto Encoders

The objective of this problem is to perform image classification given images from the Caltech image dataset using Auto encoders. This problem has 2 parts: Convert the given neural network into an auto-encoder with softmax output layer and tune the baseline auto encoder network by increasing number of hidden layers and report accuracies for each of the cases.

You are provided with the Caltech image dataset (which is a large database of classified images commonly used for training various image classification systems) for this task.

The broad steps to be followed are as follows:

- Normalize the images to a standard size
- Implement the Auto encoder
- Implement the output softmax layer
- Build and train the network.
- Test the baseline network developed above and report the accuracy
- Tune the network by increasing the number of layers and report accuracies for each of the cases.

Part 1: Perform Image Classification using Auto Encoder Network

Phase 1 – Image Normalization

Normalize the images using an image processing library to a 16x16 common format.

Checkpoint #1: Normalized JPEG (double click the image and observe)

Phase 2 – Implement the Auto-Encoder

1. You are given the implementation of an Artificial Neural Network (ANN.py). Extend the ANN to implement the auto encoder network with 2 hidden layers with number of hidden units to be 16 and 8 respectively.
   Note: An auto encoder network is a deep neural network which needs to be trained layer-wise such that the input and targets to the network are the same. (Unsupervised learning)
2. Stack a softmax output layer on top of the auto encoder.
3. Form the features such that each pixel is a feature. Note that the pixel is a vector of RGB. Thus if you have 16x16 image, you will form 256 x 3 = 768 input units or features. Normalize the input before feeding it into the network.
4. Since 5 classes of images are given to you, the image needs to be classified into one of these classes.
   For training purposes, let the target vector be a one-hot vector in accordance with the class of the image.
5. Partition the data into training and test datasets in the ratio 80% and 20% respectively.
6. Train the network and test the network for accuracy.

Checkpoint #2: Auto encoder implementation, datasets, test results

Part 2: Tune the Auto encoder by modifying the number of layers and report results for each case

Tuning of the baseline model can be done in the following ways:

1. Try with 3 hidden layers of the auto encoder network.
2. Modify the number of hidden units (among 8,16,32) in the different layers

Report accuracies of the developed model for the different cases tried.

Checkpoint #3: Results achieved by tuning the network.

Deliverables:

1. By the end of the lab, you are expected to have demonstrated the baseline Auto encoder model.
2. A zip named 'team_id_SEE.zip' with source code of all your py modules, pickle files of the networks developed and a README file.
3. A Facebook post by Friday 9:30pm describing the performance of the auto encoder on the image classification task. Also, compare the baseline network results with the tuned network (all scenarios tested by you) and give your inferences on the same.

Best wishes from your faculty and seniors ☺