COL341: Machine Learning
Assignment 2 Report
Neural Networks

Utkarsh N. Singh, 2015ME10686

September 9, 2018

Objective

To develop and train a Neural Network to classify Devanagari Handwritten Characters. The dataset consists of 32x32 grayscale images and each image belongs to 46 different classes: 36 Devanagari characters and 10 Devanagari numerals.

Part (a)

For this part, a general neural network framework was to be implemented for the above problem from scratch using only numpy and pandas, which supports the following functionalities:

1. Adding any number of hidden layers to the network with any number of hidden units
2. Choosing the activation function across all the hidden units - ReLU, sigmoid or tanh
3. Xavier initialization of weights for better convergence
4. Setting batch size and initial learning rate
5. Loss function is Mean Squared Error (MSE): $\frac{1}{N}\sum_{i=1}^{N}||Y_i^2 - \hat{Y_i}^2||^2$

Above framework supports only fully connected architectures. Learning rate is decreased based on the average loss per epoch. This ensures that convergence is reached with minimal overshooting. The above design also ensures that the user can create any architecture they want for training over the Devanagari dataset.

Sigmoid was used as the activation function in the output layer.

Part (b)

Given the training dataset of 78,200 examples, the objective was to obtain the best possible architecture for classifying the Devanagari Handwritten Characters into their respective classes and also to generalize well on unseen data.

Based on the training data size, an architecture with at most 2 hidden layers would be sufficient. Sigmoid was fixed as the activation function across all hidden units, and softmax activation function will be used in the output layer.

Additionally, the loss function to be used was now the cross-entropy function:

$$L(Y_i, \hat{Y_i}) = -\frac{1}{N} \sum_{i=1}^{N} Y_i log(\hat{Y_i})$$

With Y_i being the one-hot encoding of the actual label, and $\hat{Y_i}$ the predicted probabilities for all the classes.

Using the method of holdout with 80:20 split of the training data, over 25 different architectures with varying batch sizes, learning rates and epochs were explored. Some of the promising models were: (bs - batch size, lr - initial learning rate)

1. Layers = [500, 200], bs = 100, epochs = 100, lr = 1.4. Validation Accuracy = 94.72%
2. Layers = [400, 200], bs = 100, epochs = 100, lr = 1.4. Validation Accuracy = 94.61%
3. Layers = [500, 300], bs = 100, epochs = 100, lr = 1.4. Validation Accuracy = 94.79%
4. Layers = [700, 300], bs = 100, epochs = 50, lr = 1.4. Validation Accuracy = 94.82%
5. Layers = [700, 300], bs = 100, epochs = 50, lr = 2.0. Validation Accuracy = 94.84%

Since a balance between performance and speed is desirable, the final model architecture was set at:

Layers = [500, 300], bs = 100, epochs = 100, lr = 1.4

On the separate (public) test data, the above model gave accuracies in between 95% and 96% (since random seed wasn't fixed), which seems to be a reasonable generalization.

Part (c)

Some of the popular feature extraction techniques like Gabor Filters, Sobel Filters, FFT, HOG were tried out as an attempt towards improving the accuracy of the model.

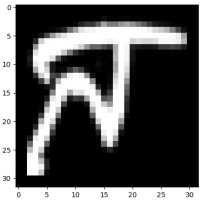

Figure 1: Original image from the dataset.

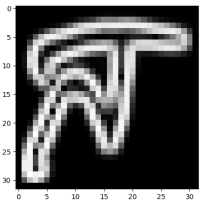

Figure 2: Sobel filter applied to original image.

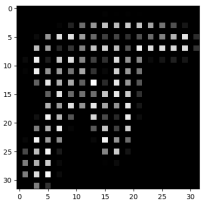

Figure 3: HOG descriptor applied to original image.

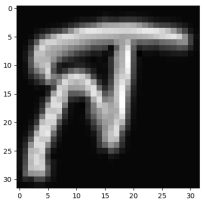

Figure 4: 4 Gabor filters applied to original image with max-pooling.

ReLU will be used as the activation function across all the hidden layers because of the problem of saturation in sigmoid (and tanh to some extent) and also because ReLU-based architectures tend to perform better than other activations.
Softmax activation is used at the output layer, and cross-entropy will be used as the loss function.

After trying out the various methods of feature extraction mentioned above, Gabor filters seem to be giving the best validation accuracy. More specifically, for the model: Layers = [500, 300], bs = 100, epochs = 120, lr = 0.2 and 4 Gabor filters with max-pooling, validation accuracy comes out to be 94.5%, which is the best accuracy that came out from the various architectures tested. So this model was chosen as the final one.

This model gave an accuracy of 94.9% on the public test data.