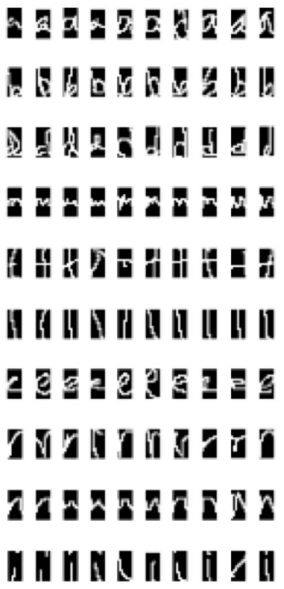
Introduction to Neural Network Model

1. Datasets and File Format

Datasets: We will be using a subset of an Optical Character Recognition (OCR) dataset. This data includes images of all 26 handwritten letters; our subset will include only the letters “a,” “e,” “g,” “i,” “l,” “n,” “o,” “r,” “t,” and “u.” The handout contains three datasets drawn from this data: a small dataset with 60 samples per class (50 for training and 10 for validation), a medium dataset with 600 samples per class (500 for training and 100 for validation), and a large dataset with 1000 samples per class (900 for training and 100 for validation). Figure below shows a random sample of 10 images of few letters from the dataset.



File Format: Each dataset (small, medium, and large) consists of two csv files—train and validation. Each row contains 129 columns separated by commas. The first column contains the label and columns 2 to 129 represent the pixel values of a 16 * 8 image in a row major format. Label 0 corresponds to “a,” 1 to “e,” 2 to “g,” 3 to “i,” 4 to “l,” 5 to “n,” 6 to “o,” 7 to “r,” 8 to “t,” and 9 to “u.” Because the original images are black-and-white (not grayscale), the pixel values are either 0 or 1. However, you should write your code to accept arbitrary pixel values in the range [0,1]. The images in the figure above were produced by converting these pixel values into .png files for visualization. Observe that no feature engineering has been done here; instead the neural network you build will learn features appropriate for the task of character recognition.

2. Model Definition

You will implement a single-hidden-layer neural network with a sigmoid activation function for the hidden layer, and a softmax on the output layer. Let the input vectors x be of length M, the hidden layer z consist of D hidden units, and the output layer ^y be a probability distribution over K classes. That is, each element y_k of the output vector represents the probability of x belonging to the class k.

$$\hat{y}_k = \frac{\exp(b_k)}{\sum_{l=1}^{K} \exp(b_l)}$$

$$b_k = \beta_{k,0} + \sum_{j=1}^{D} \beta_{kj} z_j$$

$$z_j = \frac{1}{1 + \exp(-a_j)}$$

$$a_j = \alpha_{j,0} + \sum_{m=1}^{M} \alpha_{jm} x_m$$

The objective function we will use for training the neural network is the average cross entropy. In equation below, J is a function of the model parameters α and β because ^yk is implicitly a function of x(i). Of course, ^yk and y(i)k are the kth components of ^y and y(i) respectively.

$$J(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_k^{(i)} \log(\hat{y}_k)$$

In order to use a deep network, we must first initialize the weights and biases in the network. This is typically done with a random initialization, or initializing the weights from some other training procedure. For this assignment, we will be using two possible initialization: RANDOM The weights are initialized randomly from a uniform distribution from -0.1 to 0.1. The bias parameters are initialized to zero. ZERO All weights are initialized to 0

To train, you should optimize this objective function using **stochastic gradient descent (SGD)**, where the gradient of the parameters for each training example is computed via **backpropagation.**

3. Command Line Argument

Nine command-line arguments:<train input><validation input> <train out> <validation out> <metrics out> <num epoch><hidden units> <init flag> <learning rate>. These arguments are described in detail below:

1. <train input>: path to the training input .csv file
2. <validation input>: path to the validation input .csv file
3. <train out>: path to output .labels file to which the prediction on the training data should be written
4. <validation out>: path to output .labels file to which the prediction on the validation data should be written

5. <metrics out>: path of the output .txt file to which metrics such as train and validation error should be written

6. <num epoch>: integer specifying the number of times backpropogation loops through all of the training data (e.g., if <num epoch> equals 5, then each training example will be used in backpropogation5 times).

7. <hidden units>: positive integer specifying the number of hidden units.

8. <init flag>: integer taking value 1 or 2 that specifies whether to use RANDOM or ZERO initialization that is, if init_flag==1 initialize your weights randomly from a uniform distribution over the range [-0.1,0.1] (i.e. RANDOM), if init_flag==2 initialize all weights to zero (i.e. ZERO). For both settings, always initialize bias terms to zero.

9. <learning rate>: float value specifying the learning rate for SGD.