PA1 Report

Saturday, March 6, 2021 1:32 PM

Model Description

For Multi-class logistic regression, the softmax function is given as

$$p(y=k|x) = \sigma_M(f_k(x)) = \frac{e^{f_k(x)}}{\sum_{k=1}^{K} e^{f_k(x)}}.$$

Where $f_k(x) = X W_k + W_{k0}$

if $\theta = \begin{bmatrix} W_k \\ W_{k0} \end{bmatrix}$ and $X = [x \;\; 1]$

then $f_k(x) = X\theta_k$

Hence

$$p(y=k|x) = \frac{\exp(X\theta_k)}{\sum_{k=1}^{K} \exp(X\theta_k)}$$

Using cross entropy loss function

$$\text{Loss} = -\sum_{m=1}^{M}\sum_{k=1}^{K} t[m][k] \log p(y[m]=k \,|\, X[m], \theta)$$

$$\text{Gradient} = -\sum_{m=1}^{M} \{t[m][k] - \sigma_m(X[m]\theta_k)\} X[m]$$

Where $X \in \mathbb{R}^{M \times N}$

$\theta \in \mathbb{R}^{N \times K}$

$t[m] = [0, 0, 1, 0, 0]$ if $X[m]$ belongs to class 3

Weight update:

$$\theta^{t+1} = \theta^{t} - \eta \times \text{Gradient}$$

Where η is the learning rate.

Experimental Settings

- Python version 3.8 was used for coding.
- Training and testing data were imported and processed using the hints provided. The data were normalized by dividing with 255.0
- Training and testing labels were used to obtain $t[m]$
- Tensorflow version 2.4.0 was used to build the softmax function and gradient and weight update
- Training was done by calling the classifier function and training and loss accuracies and loss were computed.
- Weights are saved.

- Testing data were separated into individual digits and their classification errors calculated. Overall errors are also calculated.

- Weights of individual classes and testing, training errors and accuracies plotted.

Hyper-parameters

learning rate = 0.75
iteration = 2000 (was decided from the results obtained using higher values).

Figures to attach here

(1) training testing error curve

(2) training testing accuracy curve

(3) Weights (1 to 5)

Classification Errors:

Digit 1 = 0.0141
Digit 2 = 0.0525
Digit 3 = 0.0680
Digit 4 = 0.0254
Digit 5 = 0.0776

average = 0.0468