Custom_OneLayer_NN_SGD_Report - Hanwen Hu

September 7, 2018

In my Homework 1 code, I have achieved a test accuracy of 98.04%.
A brief introduction of my implementation process is given below.
Basic Design of Forward Propagation Suppose there's an input $X \in \mathbb{R}^p$ indicating input features, and $Y \in \{0, 1, \dots, K-1\}$ as the corresponding class, and h hidden units in the single-layer neural network. Then, our parameter set is given by (W, b_1, C, b_2), where $W \in \mathbb{R}^{h \times p}$, $b_1 \in \mathbb{R}^h$, $C \in \mathbb{R}^{K \times h}$ and $b_2 \in \mathbb{R}^K$.
As for the activation function part, ReLu is applied as that of the hidden layer, and Softmax as that of the output layer.
Therefore, the basic design of network is given as below, where $H \in \mathbb{R}^h$ is the input towards the output layer, and $\hat{Y}$ is the output probability distribution over all possible classes.

$$\begin{aligned} Z &= WX + b_1, \\ H &= \sigma_{ReLu}(Z), \\ U &= CH + b_2, \\ \hat{Y} &= F_{softmax}(U). \end{aligned}$$

0.0.1 Basic Design of Backward Propagation (SGD)

The basic design of backward propagation is given below, where dU, dC, dZ, dW corresponds to the negative gradient of U, C, Z and W. Note that Y is recoded by one-hot and transformed to a K-dimension vector.

$$\begin{aligned} dU &= \hat{Y} - Y, \\ db_2 &= dU, \\ dC &= dUH^T, \\ dZ &= C^T dU \bigodot \mathbb{1}_{Z>0}, \\ db_1 &= dZ, \\ dW &= dZX^T. \end{aligned}$$

With learning rate α, the parameters can be updated in the following way:

$$\begin{aligned} b_2 &= b_2 + \alpha * db_2, \\ C &= C + \alpha * dC, \\ b_1 &= b_1 + \alpha * db_1, \\ W &= W + \alpha * dW, \end{aligned}$$

0.0.2 Hyperparameter Settings

My network has set the number of hidden units to be 100, number of epochs as 10, and original learning rate as 0.01, which decays to 10% for every 5 epochs. ### Parameter Initialization Parameter initialization turns out to be an important element affecting test accuracy. b_1, b_2 can be easily initialized as zeros, and W and C are randomly initialized. I've tried 4 sets of random initialization, i.e. uniform(-1,1), uniform(-1,1) divided by $\sqrt{p}$, standard Gaussian and standard Gaussian divided by $\sqrt{p}$, and get them implemented with the same hyperparameter settings above. The first set has a test accuracy of 96.31%, the second 98.03%, the third 94.27%, and the fourth 98.04% (which is adopted in the final design).