MULTI LAYER PERCEPTRON FOR MNIST DATABASE

TIANBO YANG

The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset. It is a dataset of 60,000 small square 28 $\times$ 28 pixel grayscale images of handwritten single digits between 0 and 9.

MNIST is a very well-studied data set of 28x28 images of isolated digits (0-9), each pixel value in the range 0-255. There are 60,000 training images and 10,000 validation images.

Last week we implemented a framework for building neural networks from scratch. We trained our models using stochastic gradient descent. In this problem, we explore how we can implement batch normalization as a module BatchNorm in our framework.

Our data consists of the form $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_{784} \end{bmatrix} \in \mathbb{R}^{784}$ and $\mathbf{y} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{10}$, where the ith entry is 1 if the data represent the number i, otherwise, the entries are zero.

To classify 10 digits, we define a function $h_{\mathbf{w}} : \mathbb{R}^{784} \to \mathbb{R}^{10}$

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + \boldsymbol{w_0} = \begin{bmatrix} \sum_{i=1}^{784} w_{i,1} x_i + w_{0,1} \\ \vdots \\ \sum_{i=1}^{784} w_{i,10} x_i + w_{0,10} \end{bmatrix} = \begin{bmatrix} z_1 \\ \vdots \\ z_{10} \end{bmatrix} \in \mathbb{R}^{10}$$

where $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_{784} \end{bmatrix} \in \mathbb{R}^{784}$, $\mathbf{w} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,10} \\ \vdots & \vdots & \vdots \\ w_{784,1} & \cdots & w_{784,10} \end{bmatrix}$, and $\mathbf{w}_0 = \begin{bmatrix} w_{0,1} \\ \vdots \\ w_{0,10} \end{bmatrix}$.

Rectified linear unit

$$\mathrm{ReLU}(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise} \end{cases} = \max\{0, z\}$$

Convolutional layers: Filters in the first convolutional layers are responsible for detecting low-level features (e.g., edges, color, contrast). Later convolutional layers are responsible for detecting mid-level features (e.g., ears, eyes).

Max pooling layers: Max pooling layers detect the strongest response within a given window. This property allows the network to be less sensitive to feature locations.

Fully connected layers: Fully connected layers allow combining features from the entire image and provide the final network output.

Softmax function $\mathbf{z} \in \mathbb{R}^{10} \to P \in [0,1]^{10}$ with $\sum_{i=1}^{10} P_i = 1$ (a probability distribution over 10 items)

$$\boldsymbol{a} = \mathrm{softmax}(\mathbf{z}) = \begin{bmatrix} \frac{\exp(z_1)}{\sum_{i=1}^{10} \exp(z_i)} \\ \vdots \\ \frac{\exp(z_{10})}{\sum_{i=1}^{10} \exp(z_i)} \end{bmatrix}$$

The loss function $\mathrm{Loss}(\boldsymbol{a}, \boldsymbol{y})$

$$\mathrm{NLL}(\boldsymbol{a}, \boldsymbol{y}) = -\sum_{j=1}^{10} y_j \ln(a_j)$$

$$z_j = \sum_{i=1}^{784} w_{i,j} x_i + w_{0,j} \quad 1 \leq j \leq 10$$

$$\frac{\partial \mathrm{NLL}(\boldsymbol{a}, \boldsymbol{y})}{\partial w_{i,j}} = x_i(a_j - y_j), \quad 1 \leq i \leq 784,\ 1 \leq j \leq 10$$

$$\frac{\partial \mathrm{NLL}(\boldsymbol{a}, \boldsymbol{y})}{\partial w_{0,j}} = a_j - y_j, \quad 1 \leq j \leq 10$$

References

[1] Al-Omari, S. A. K., Sumari, P., Al-Taweel, S. A., & Husain, A. J. (2009). Digital Recognition using Neural Network. Journal of Computer Science, 5(6), 427-434. https://doi.org/10.3844/jcssp.2009.427.434

[2] Block, H. (1970). A review of perceptrons: An introduction to computational geometry. Information and Control, 17(5), 501-522. https://doi.org/10.1016/s0019-9958(70)90409-2

[3] Verma, J. (2022, August 3). MNIST Dataset in Python - Basic Importing and Plotting. DigitalOcean Community. https://www.digitalocean.com/community/tutorials/mnist-dataset-in-python