Project update of handwritten digit recognition

Xin Wang

November 2020

After the first progress update, I have implemented the neural network algorithm. The experiment on the neural network can be found in here.

1 Structure of the neural network

The network has three hidden layers. The first layer has more nodes than the remaining two layers.

Three hidden layer, output is a 10-by-1 vector. For digit i, the i^{th} element of the label is 1. All other elements are 0. For example, the label for 2 is $[0010000000]^T$.

$$x \xrightarrow{w^{(1)}} h^{(1)} \xrightarrow{w^{(2)}} h^{(2)} \xrightarrow{w^{(3)}} h^{(3)} \xrightarrow{v} d$$

The activation function is the logistic function,

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$$

2 Equation

First, the gradient equations are write into a more compact matrix format, which is more clear and easier to transform into code.

loss function is $f(w) = \frac{1}{2}(\hat{d}-d)^2$, where $\hat{d}$ is the prediction, d is the actual label.

2.1 gradient over v

$$\frac{\partial f}{\partial v_{ij}} = (\hat{d}_i - d_i)\hat{d}_i(1-\hat{d}_i)h_j^{(3)}$$

with $\tilde{d}_i = (\hat{d}_i - d_i)\hat{d}_i(1-\hat{d}_i), \partial\tilde{v_{ij}} = \frac{\partial f}{\partial v_{ij}}$,

$$\partial\tilde{v} = \tilde{d}h^{(3)^T}$$

2.2 gradient over $w^{(3)}$

$$\frac{\partial f}{\partial w_{ab}^{(3)}} = \sum_i (\hat{d}_i - d_i)\hat{d}_i(1-\hat{d}_i)v_{ia}h_a^{(3)}(1-h_a^{(3)})h_b^{(2)}$$

with $\tilde{h}_a^{(3)} = h_a^{(3)}(1-h_a^{(3)}), \partial\tilde{w}_{ab}^{(3)} = \frac{\partial f}{\partial w_{ab}^{(3)}}$,

$$\partial\tilde{w}^{(3)} = (\tilde{d}^T v)^T \circ \tilde{h}^{(3)} h^{(2)^T}$$

where $\circ$ represents element-wise multiplication. For $B_{n\times m} = A_{n\times m} \circ x_{1\times m}$ where dimensions do not match, $B_{ij} = A_{ij}x_j$.

2.3 gradient over $w^{(2)}$

$$\frac{\partial f}{\partial w_{cd}^{(2)}} = \sum_i (\hat{d}_i - d_i)\hat{d}_i(1-\hat{d}_i)\sum_a v_{ia}h_a^{(3)}(1-h_a^{(3)})w_{ac}^{(3)}h_c^{(2)}(1-h_c^{(2)})h_d^{(1)}$$

with $\tilde{h}_c^{(2)} = h_a^{(2)}(1-h_c^{(2)}), \partial\tilde{w}_{cd}^{(2)} = \frac{\partial f}{\partial w_{cd}^{(2)}}$,

$$\partial\tilde{w}^{(2)} = (\tilde{d}^T v \circ (\tilde{h}^{(3)})^T w^{(3)})^T \circ \tilde{h}^{(2)} h^{(1)^T}$$

2.4 gradient over $w^{(1)}$

$$\frac{\partial f}{\partial w_{ef}^{(1)}} = \sum_i (\hat{d}_i - d_i)\hat{d}_i(1-\hat{d}_i)\sum_a v_{ia}h_a^{(3)}(1-h_a^{(3)})\sum_c w_{ac}^{(n)}h_c^{(2)}(1-h_c^{(2)})w_{ce}^{(2)}h_e^{(1)}(1-h_e^{(1)})x_f$$

with $\tilde{h}_e^{(1)} = h_e^{(1)}(1-h_e^{(1)}), \partial\tilde{w}_{ef}^{(1)} = \frac{\partial f}{\partial w_{ef}^{(1)}}$,

$$\partial\tilde{w}^{(1)} = (\tilde{d}^T v \circ (\tilde{h}^{(3)})^T w^{(3)} \circ (\tilde{h}^{(2)})^T w^{(2)})^T \circ \tilde{h}^{(1)} x^T$$

3 Results

At first, the weights are all initialized to zero before training. But the training is unsuccessful. $w^{(1)}$ is always zeros.

Then the weights are initialized using Gauss distribution. And the training works fine.

3.1 One image per iteration

3.1.1 Change the number of iterations

Iterations(10^4)	Nodes of layer 1	Nodes of layer 2	Nodes of layer 3	error rate
100	50	50	50	0.199
80	50	50	50	0.197
60	50	50	50	0.213
40	50	50	50	0.234
20	50	50	50	0.39
6	50	50	50	0.542

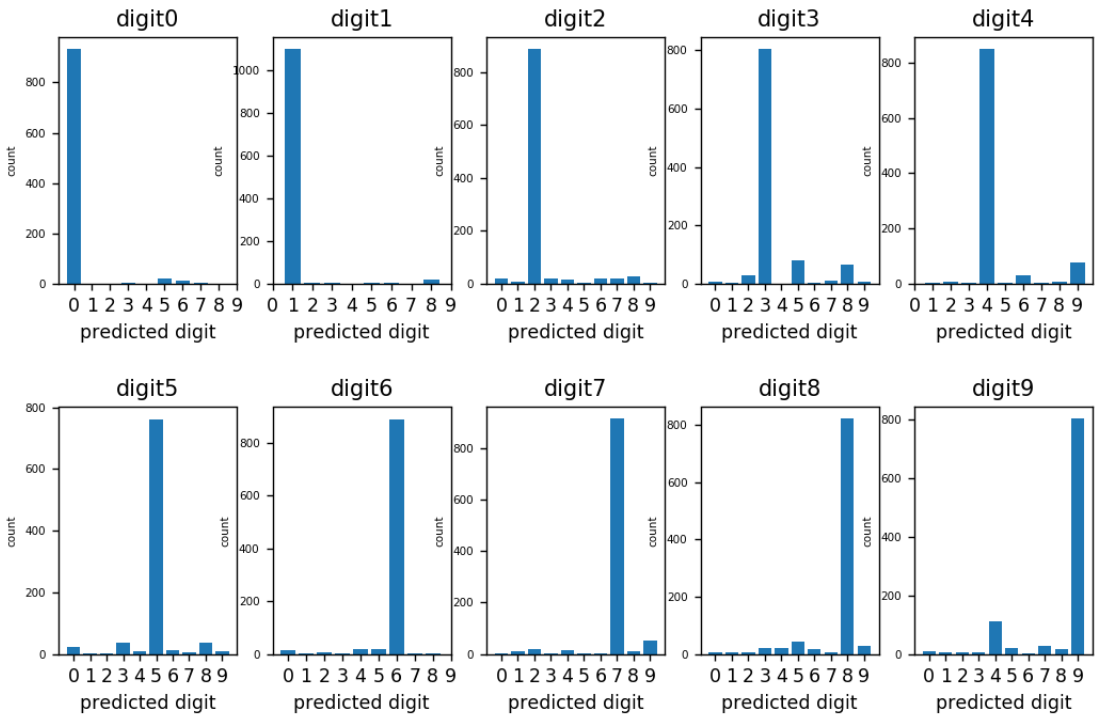

Figure 1: Error rate of each digit for neural network. The parameter is the 3rd row of 3.2.

3.1.2 Change the number of nodes of the first layer

Iterations(10^4)	Nodes of layer 1	Nodes of layer 2	Nodes of layer 3	error rate
80	50	50	50	0.197
80	60	50	50	0.1915
80	70	50	50	0.188
80	80	50	50	0.171
80	90	50	50	0.187
80	100	50	50	0.170
80	150	50	50	0.155
80	300	50	50	0.140

3.2 Batch training

The gradient would be more stable using batch training.

Iterations(10^4)	Nodes of layer 1	Nodes of layer 2	Nodes of layer 3	Batch size	error rate
1	150	50	50	80	0.1519
1	150	50	50	128	0.137
1	150	50	50	256	0.124