Coding-based tutorial about Artificial Neural Networks (ANNs)

Guo-Wei Wei[1,2,3] and Rui Wang[1]
[1] Department of Mathematics, Michigan State University, MI 48824, USA
[2] Department of Biochemistry and Molecular Biology
Michigan State University, MI 48824, USA
[3] Department of Electrical and Computer Engineering
Michigan State University, MI 48824, USA

Contents

1 Structure of ANN (One hidden layer)

In this tutorial, we will employ ANN (with only one hidden layer) to solve the classification problem. First, we will give a brief introduction about Digits dataset and One-Hot Encoding. Next, the feed-forward and back-propagation will be introduced.

1.1 Digits Dataset

Digits Dataset is made up of 1797 8×8 images. We randomly spilt the dataset into training set and test set.

- X_train.shape = (1347, 64)
- X_test.shape = (450, 64)
- y_train.shape = (1347,1) y_train_ohe.shape = (1347,10)
- y_test.shape = (450, 1) y_test_ohe.shape = (450,10)

Here, 1347 is the #of training samples, 450 is the #of test samples, 64 is the feature size, and 10 is the #of classes.

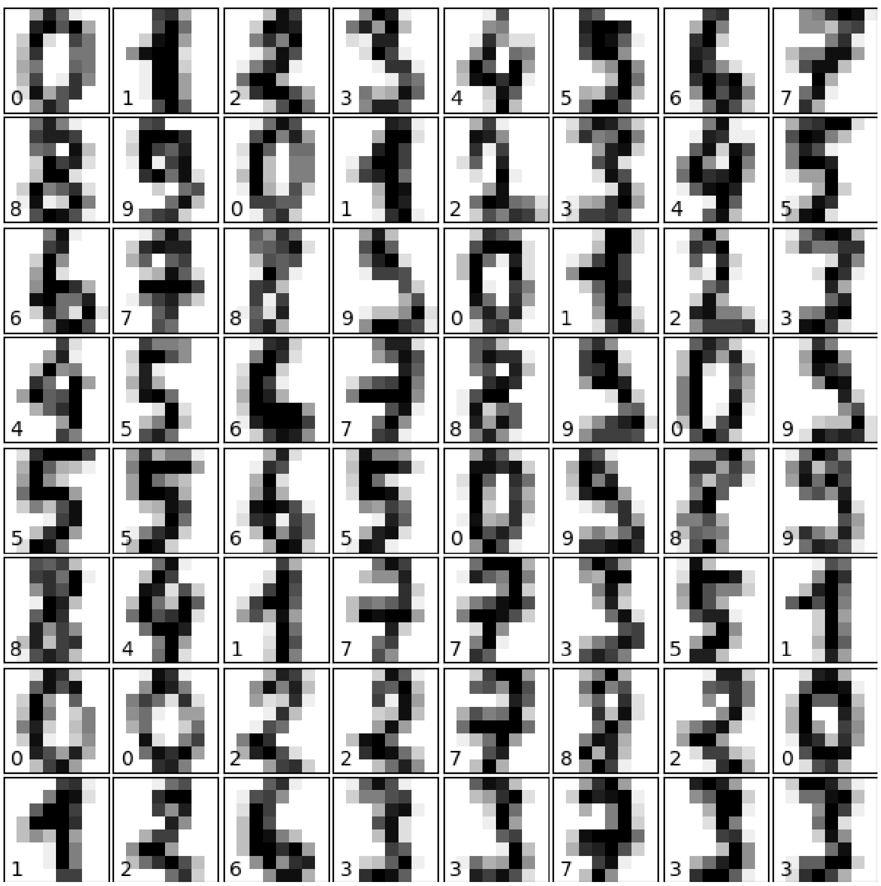

Figure 1: Digits Dataset.

1.2 One-Hot Encoding

One-Hot code can represent a vector $[0, 1, 2]$ as $[100, 010, 001]$. In the Digits Dataset, we can one-hot encode a categorical data 7 to 0000000100.

1.3 Feed-Forward

We will construct a deep neural network with only one hidden layer as illustrated in Figure 2. The input layer has 64 neurons: $x_1, x_2, ..., x_{64}$.

1. In 1st hidden layer:
$$z_1 = xW_1 + b_1, \tag{1}$$
where $W_1 \in \mathbb{R}^{64\times N_1}$, $b_1 \in \mathbb{R}^{N_1}$, $z_1 \in \mathbb{R}^{M\times N_1}$. Note: N_1 is # of neurons in this hidden layer and M is the number of samples. Next, we will add activation function. For example, relu function or $\tanh$ function:
$$f_1 = \tanh(z_1) \in \mathbb{R}^{M\times N_1}. \tag{2}$$

2. In the output layer:
$$z_2 = f_1W_2 + b_2, \tag{3}$$
where $W_2 \in \mathbb{R}^{N_1\times 10}$ and $b_2 \in \mathbb{R}^{10}$, $z_2 \in \mathbb{R}^{M\times 10}$.

3. Use softmax function to get probability for each class.
$$\hat{y} = \textbf{softmax}(z_2). \tag{4}$$
Here, the softmax function is $\textbf{softmax}(z)_j = \dfrac{e^{z_j}}{\sum_k e^{z_k}}$.

Note:

- x.shape = (M, 64). M is the number of samples.
- W_1.shape = (64, N_1) b_1.shape = (1, N_1) z_1.shape = (M, N_1) f_1.shape = (M, N_1)
- W_2.shape = (N_1, 10) b_2.shape = (1, 10) z_2.shape = (M, 10) $\hat{y}$.shape = (M, 10)

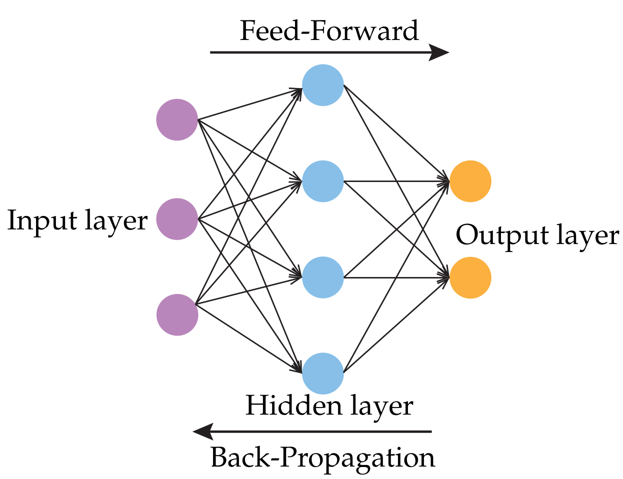

Figure 2: Illustration of ANN.

1.4 Back-Propagation

1. Loss function (cross-entropy loss)
$$L = -\sum_i y_i \log(\hat{y}_i). \tag{5}$$

2. The following derivatives need to be considered: $\dfrac{\partial L}{\partial W_2}, \dfrac{\partial L}{\partial b_2}, \dfrac{\partial L}{\partial W_1}, \dfrac{\partial L}{\partial b_1}$.

- $\dfrac{\partial L}{\partial W_2} = \dfrac{\partial L}{\partial \hat{y}} \cdot \dfrac{\partial \hat{y}}{\partial z_2} \cdot \dfrac{\partial z_2}{\partial W_2} = f_1^T(\hat{y}-y)$, and $\dfrac{\partial L}{\partial W_2}$.shape = (N_1, M) $(M, 10)$ = $(N_1$,10). (See Derivative of softmax loss function)
- $\dfrac{\partial L}{\partial b_2} = \dfrac{\partial L}{\partial \hat{y}} \cdot \dfrac{\partial \hat{y}}{\partial z_2} \cdot \dfrac{\partial z_2}{\partial b_2} = \sum\limits_{\text{axis}=0} (\hat{y}-y)$, and $\dfrac{\partial L}{\partial b_2}$.shape = $\sum\limits_{\text{axis}=0} (M$,10) = (1,10)
- $\dfrac{\partial L}{\partial W_1} = \dfrac{\partial L}{\partial \hat{y}} \cdot \dfrac{\partial \hat{y}}{\partial z_2} \cdot \dfrac{\partial z_2}{\partial f_1} \cdot \dfrac{\partial f_1}{\partial z_1} \cdot \dfrac{\partial z_1}{\partial W_1} = x^T \cdot \left[(1-f_1^2) \cdot (\hat{y}-y)W_2^T\right]$

  Here, $\dfrac{\partial L}{\partial W_1}$ = (64,M) ·[(M,N_1) · [(M,10) (10,N_1)]] = (64, N_1)
- $\dfrac{\partial L}{\partial b_1} = \dfrac{\partial L}{\partial \hat{y}} \cdot \dfrac{\partial \hat{y}}{\partial z_2} \cdot \dfrac{\partial z_2}{\partial f_1} \cdot \dfrac{\partial f_1}{\partial z_1} \cdot \dfrac{\partial z_1}{\partial b_1} = \sum\limits_{\text{axis}=0} (1-f_1^2))(\hat{y}-y)W_2^T$, and $\dfrac{\partial L}{\partial b_1}$.shape = (1,$N_1$).

Let $d_1 = (1-f_1^2)(\hat{y}-y)W_2^T, d_2 = \hat{y}-y$, then

- $\dfrac{\partial L}{\partial W_2} = f_1^T d_2$
- $\dfrac{\partial L}{\partial b_2} = d_2$
- $\dfrac{\partial L}{\partial W_1} = x^T d_1$
- $\dfrac{\partial L}{\partial b_1} = d_1$