1 Artificial Neural Network

1.1 Theory

In machine learning, artificial neural network is a supervised learning method providing a robust approach to approximating real-valued, discrete-valued and vector-valued target functions. It is one of the most effective methods for certain types of problems such as learning to interpret complex real-world sensor data. Inspired by biological neural system in the human brain, basically neural network contains a densely interconnected set of simple units, where each unit takes a number of real-valued inputs and produces a single real-valued output (which may become the input to many other units).

To train a neural network with an optimization method, e.g, gradient descent, we need to know the partial derivatives of our cost function with regard to each network's parameters. For a network with no hidden layer (similar to logistic regression models), this is quite trivial. However, using the same logic with multi-layer networks will only give us the gradient of the last layer. Luckily, backpropagation provides a way to calculate gradient of all the layers using chain rule. To understand more about the intuition of backprogragation, we recommend this tutorial.

1.2 Python Exercise

In this exercise, you will have to build your neural network, do forward and backward calculation. Then, you will update the network's parameters using gradient descent. This exercise uses the `digits` dataset from `sklearn`, you can see some samples in Fig. 1. You will work with the file *neural_network.ipynb*. The architecture of our network is: $\texttt{input} -> \texttt{fully\_connected} -> \texttt{sigmoid} -> \texttt{fully\_connected} -> \texttt{sigmoid} -> \texttt{output}$

Step 1: Load, split data and convert labels to one-hot vectors. Like exercise 1, you will load the dataset and split it into training and testing sets. You also need to add bias term to the input features, and convert the training output to a one-hot vector.

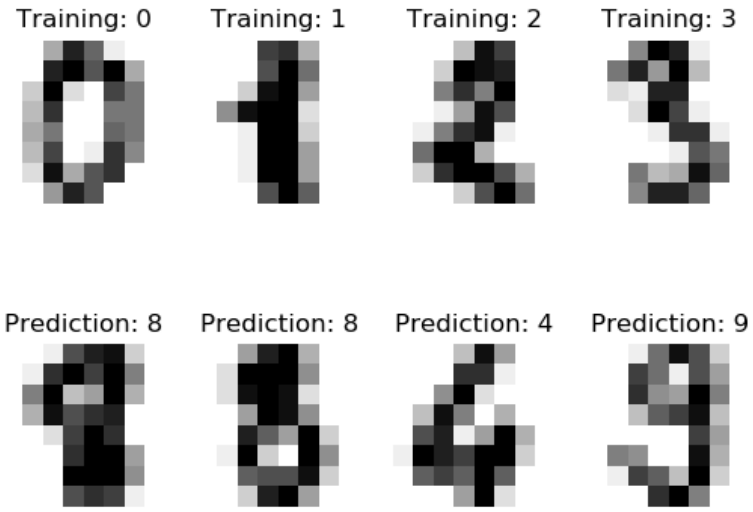

Figure 1: The digit dataset from sklearn.

Step 2: Forward computation. In this step, you will implement several building blocks of a neural network, including the fully-connected layer, the sigmoid activation function and the mean square error cost function. Afterwards, you need to stack the building block sequentially to construct the above network architecture.

Step 3: Backpropagation Using backpropagation, you will calculate the gradient for each parameter of the network in this step. First, we need to calculate the backward gradient for each building block of the network. Afterwards, we traverse the network in reverse order to backpropagate the gradient using the above functions.

Step 4: Network training. After we have forward and backward passes working, we will train the network using batch gradient descent. You need to loop over your training set many times. For each iteration, we do one forward pass, calculate the loss, and do one backward pass to calculate all the neccessary gradients. Then, you update the network's parameters using gradient descent rule.

Step 5: Evaluation In this step, you will evaluate the performance of your network. You will calculate the classification on the test set. You should be able to achieve more than 90% accuracy