School of Computer Science Engineering and Technology

Course- BTech
Course Code: CSET 301
Year- 2022-23
Date-

Type- Specialization Core
Course Name- AIML
Semester- 4th
Batch-

Lab Assignment W6L1

Problem: Predict the median home value of a given census tract in the City of Boston.

Dataset: The Boston Housing Dataset | Kaggle

Instructions:

1. Download the dataset.

2. Pre-process the data by normalizing the values and splitting into training and testing sets.

3. Construct a regression-based artificial neural network with an appropriate number of layers and nodes.

4. Train the neural network using the training set.

5. Evaluate the performance of the model using the testing set.

6. Utilize the trained model to predict the median home value of a given census tract in the City of Boston.

7. Compare the predicted values to the actual values and assess the model's accuracy.

ANN

Artificial neural networks are relatively crude electronic networks of neurons based on the neural structure of the brain. They processrecords one at a time, and learn by comparing their classification of the record (i.e., largely arbitrary) with the known actual classification of the record. The errors from the initial classification of the first record is fed back into the network, and used to modify the networks algorithm for further iterations.

A neuron in an artificial neural network is

1. A set of input values (xi) and associated weights (wi).

2. A function (g) that sums the weights and maps the results to an output (y).

Parameters

We can choose the dimensionality (the number of nodes) of the hidden layer. The more nodes we put into the hidden layer the more

complex functions we will be able fit. But higher dimensionality comes at a cost. First, more computation is required to make predictions

and learn the network parameters. A bigger number of parameters also means we become more prone to overfitting our data. How to

choose the size of the hidden layer? While there are some general guidelines and recommendations, it always depends on your specific

problem and is more of an art than a science. We will play with the number of nodes in the hidden layer later on and see how it affects our output.

Activation function

We also need to pick an activation function for our hidden layer. The activation function transforms the inputs of the layer into its

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{n \in N} \sum_{i \in C} y_{n,i} \log \hat{y}_{n,i}$$

outputs. A nonlinear activation function is what allows us to fit nonlinear hypotheses. Common choices for activation functions are tanh, the sigmoid function, or ReLUs.

Learning the parameters for our network means finding parameters ( ) that minimize the error on our training data. But how do we define the error? We call the function that measures our error the loss function. A common choice with the softmax output is the cross-entropy loss. If we have training examples and classes, then the loss for our prediction with respect to the true labels is given by:

Remember that our goal is to find the parameters that minimize our loss function. We can use gradient descent to find its minimum. I will implement the most vanilla version of gradient descent, also called batch gradient descent with a fixed learning rate. Variations such as SGD (stochastic gradient descent) or minibatch gradient descent typically perform better in practice. So, if you are serious you'll want to use one of these, and ideally you would also decay the learning rate over time.

Gradient descent

As an input, gradient descent needs the gradients (vector of derivatives) of the loss function with respect to our parameters:

$$\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial b_1}, \frac{\partial L}{\partial W_2}, \frac{\partial L}{\partial b_2}$$

To calculate these gradients, we use the famous backpropagation algorithm, which is a way to efficiently calculate the gradients starting from the output. I won't go into detail how backpropagation works, but there are many excellent explanations (here or here) floating around the web.

Backpropagation formula

Applying the backpropagation formula, we find the following:

$$\delta_3 = \hat{y} - y$$
$$\delta_2 = (1 - \tanh^2 z_1) \circ \delta_3 W_2^T$$
$$\frac{\partial L}{\partial W_2} = a_1^T \delta_3$$
$$\frac{\partial L}{\partial b_2} = \delta_3$$
$$\frac{\partial L}{\partial W_1} = x^T \delta_2$$
$$\frac{\partial L}{\partial b_1} = \delta_2$$