Structural Machine Learning Models and Their Applications

Homework 01

K.V.D.T.T. Abeywardhana
7108053024

Department of Applied Mathematics
National Chung Hsing University
Taichung
Taiwan

1. Derive the forward and backward schemes for the Regression Problem, where the cost is $f(w) = \sum_{i=1}^{n} \|\hat{y^i} - y^i\|^2$.

   For Forward scheme: Then
$$Z^{(h)} = (W^{(h)})^T A^{(in)}$$
   for k samples and thus
$$A^{(h)} = \Phi(Z^{(h)})$$
   .
   Where Φ, Z, A, W are Activation function, Layers, Nodes values and Weights respectively.
   Similarly for the out put layer (final layer)
$$Z^{(out)} = (W^{(out)})^T A^{(h)}$$
   and
$$A^{(out)} = \Phi(Z^{(out)})$$
   For the Back-propagation:
   Target - y ; Output - $\hat{y}$
   Error term of the outer layer
$$\delta^{(out)} = A^{(out)} - y$$
   and the gradient
$$\frac{\partial F(w)}{\partial {w_{i,j}}^{(out)}} = A^{(h)}.\delta^{(out)}.$$
   Error term for Hidden layers
$$\delta^{(h)} = \delta^{(out)}.(w^{(out)})^T.\frac{\Phi(z^{(h)})}{\partial z^{(h)}}$$
   and the gradient
$$\frac{\partial F(w)}{\partial w^{(h)}} = A^{(in)}.\delta^{(h)}.$$
   Apply chain rule to the composite function $F(w)$.
   Uses Gradient Descent method to update the weight or parameters at every node, initial, we took random weights and for sure using those weight out model is going to perform like trash, But these random weights are a method for us to calculate loss. So that we can update weight late and check the model accuracy.

2. Prepare the house data set, and then do the prepossessing to the data.
   I have uploaded the regard python code.

3. Implement the 3-layer MLP (including the cost, forward and backward schemes) for the regression problem.
   The final results are given below.

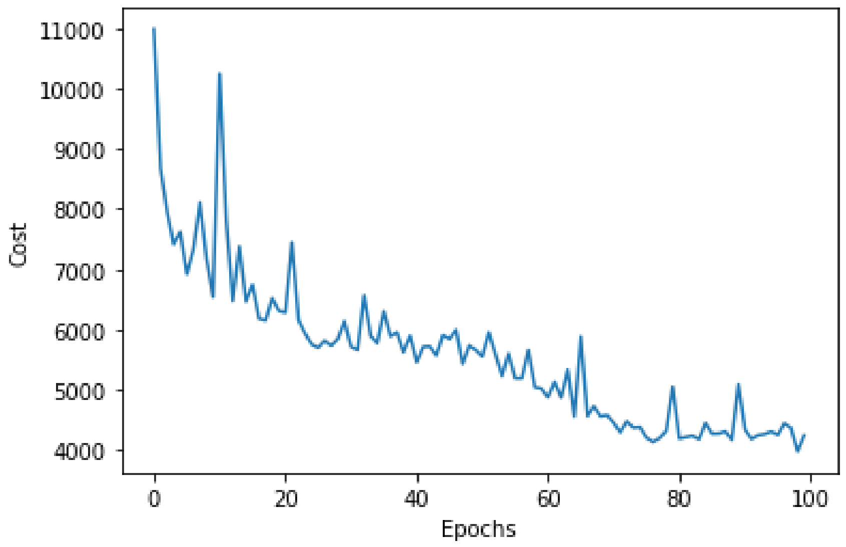
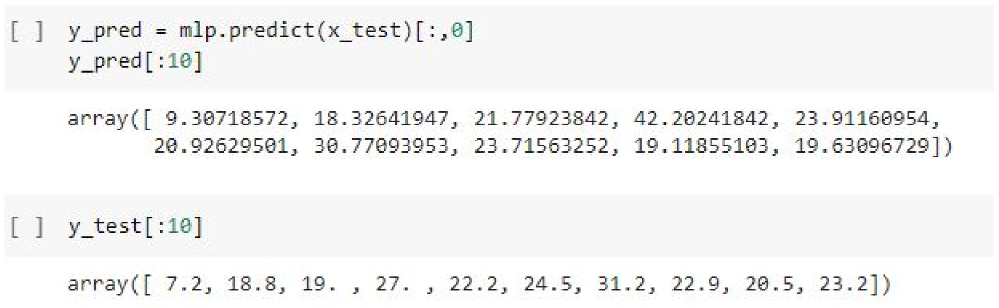


Figure 1: Convergence of Cost value for 100 Epochs



```
[ ]  y_pred = mlp.predict(x_test)[:,0]
     y_pred[:10]

     array([ 9.30718572, 18.32641947, 21.77923842, 42.20241842, 23.91160954,
            20.92629501, 30.77093953, 23.71563252, 19.11855103, 19.63096729])
```

```
[ ]  y_test[:10]

     array([ 7.2, 18.8, 19. , 27. , 22.2, 24.5, 31.2, 22.9, 20.5, 23.2])
```

Figure 2: Cost values of test and prepared

Here in the Fig.(1) we can see the convergence of Cost function. Also, the cost values is 26.595278656865993. So, it is good enough value for the cost value and we accept the final results.

In Fig.(2),Clearly, we can see, there are not much difference between Y_test values and $Y_prepared$ values.

Also Fig.(3), we can see there are only few values have been deviation from the regression line. So, regression is acceptable.

4. (Bouns) Implement the n-layer MLP for the regression problem, where n is an integer to be specified.
   So, I have taken $n = 10$

5. Compare the performance of the above models.
   Clearly, we can see the convergences cost of 10-layer MLP is so quick and less than the 3-layer MLP comparing the Fig(1) and Fig(4).

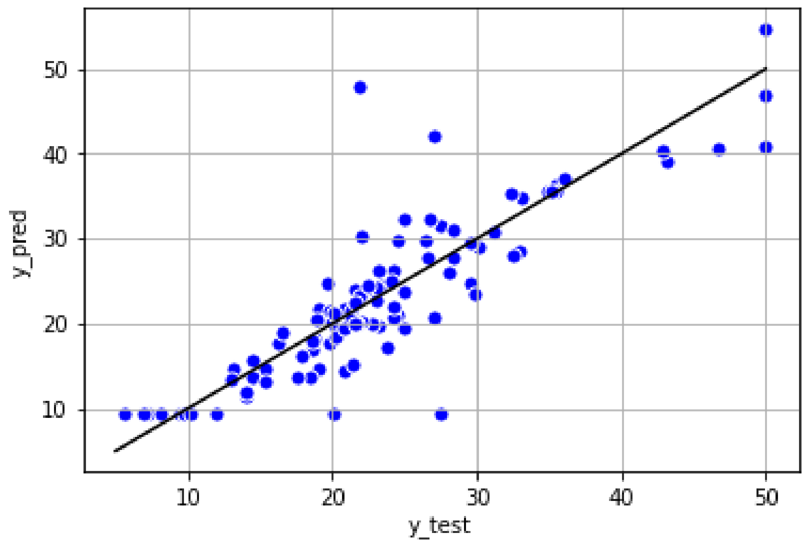


Figure 3: Regression line

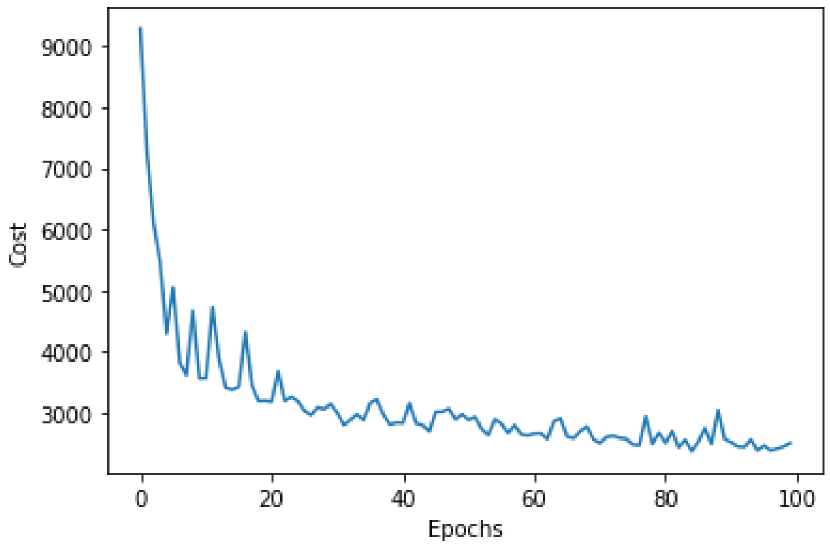


Figure 4: Cost for 10-layer MLP