Lecture Note
Training Deep Network

Jiaul Paik

1 Architectural Parameters

- Number of layers: L (say).
- Number of node in layer l: n_l (say).

Clearly, n_1 is the number of features in your data and n_L is number of classes your training data contains.

2 Model Parameters

- The connection weights: $w_{ij}^{(l)} \rightarrow$ the weight from the i-th node in layer l to the j-th node in layer $l+1$.
- The bias of each non-input node: $b_i^{(l)} \rightarrow$ the bias of i-th node in layer l.

3 Activation Function

Some of the widely used functions:

- Sigmoid: $f(x) = \frac{e^x}{1+e^x}$
- ReLU: $f(x) = max(0, x)$

4 Input and Output of a Node

$I_i^{(l)}$ denotes the input to the i-th node of layer l and is defined as

$$I_i^{(l)} = b_i^{(l)} + \sum_{j=1}^{n_{l-1}} w_{ji}^{(l-1)} \cdot O_j^{(l-1)}$$

$O_i^{(l)}$ denotes the output from the i-th node of layer l and is defined as

$$O_i^{(l)} = f(I_i^{(l)})$$

where f is the activation function.
Note that $O_i^{(1)} = x_i$, where x_i is the i-th feature value of the sample.

5 Gradient Descent using Backpropagation Algorithm

$\delta_i^{(l)}$ denotes the influence of i-th node of layer l on the error.

1. Randomly initialize the model parameters.
2. Randomly shuffle the data
3. Repeat steps **4–9** until convergence
4. For each sample (x, c) in your training data (x is the feature vector and c is label), repeat steps **5–9**
5. Take a forward pass to compute input and output of each nodes of the network (except for the nodes in the input layer)
6. For each output node i in layer L (the output layer), compute

   $$\delta_i^{(l)} = (O_i^{(L)} - y_i) \cdot f'(I_i^{(L)})$$

   (here $y_i = 1$, if $c = i$, otherwise $y_i = 0$).
7. For $l = L-1, L-2, ..., 2$
   For each node i in layer l, compute

   $$\delta_i^{(l)} = \left(\sum_{j=1}^{n_{l+1}} w_{ij}^{(l)} \cdot \delta_j^{(l+1)} \right) f'(I_i^{(l)})$$

8. Compute the partial derivative of the error function as follows:

   $$\frac{\partial E}{\partial w_{ij}^{(l)}} = O_i^{(l)} \cdot \delta_j^{(l+1)}$$

   $$\frac{\partial E}{\partial b_i^{(l)}} = \delta_i^{(l)}$$

9. Update the parameters using gradient descent rule.

Practical Guidelines for Training

1. It is important to initialize the parameters randomly. A good method for random initilization could be $Normal(0, \epsilon^2)$, where ϵ can be a small value, say 0.01.

2. Choice of learning rate is very important. Often a good start is between 0.01 to 0.1.

3. Picking good architectural parameters is very important as it will determine the non-linearity of the decision boundaries you are going to approximate. This can be tuned via cross-validation as follows. First, randomly split your training set into subsets S_1 and S_2. Then train your model on S_1 and test your model on S_2 and see the error. If the error on S_2 is very low, you can consider the current architecture is good. If the performance on S_2 is poor, but performance was good on S_1, this means the network overfits. Reduce the number of hidden layers and train again.