

6. Appendix

6.1. Modified Backpropagation

For notion simplicity, we consider stochastic gradient descent. The objective function we want to minimize is as following:

$$E = L(\mathbf{y}, \phi(\mathbf{x})) + \lambda_1 \sum_{ij} \left\| \frac{\partial \phi(\mathbf{x})}{\partial x_i} - \frac{\partial \phi(\mathbf{x})}{\partial x_j} \right\|^2 S_{ij} \quad (12)$$

Notice that the objective function includes derivative of the learned function with respect to the input features, if we use neural network to learn the model, the conventional back-propagation algorithm can't be applied directly. Therefore, we developed a modified version of the backpropagation algorithm to find the gradient of the objective.

We keep the notation consistent with the notation used in the book of (Bishop, 1995). n is the total layers (including input and out put layer) number of the network, a^k is the pre-activation units in layer k , k_1 is the number of hidden units in hidden layer k , m is the number of output units, and $h(x)$ stands for the non-linear activation function.

$$\begin{aligned} \mathbf{z}^0 &= \mathbf{x} \\ \mathbf{a}^k &= \mathbf{W}^k \mathbf{z}^{k-1} + \mathbf{b}^k \\ \mathbf{z}^k &= h(\mathbf{a}^k) \\ \phi(\mathbf{x}) &= \mathbf{z}^n \end{aligned} \quad (13)$$

To find the gradient of (12), we define δ^k as the Jacobian of the learned function with respect to pre-activations at the layer k :

$$\delta^k = \begin{bmatrix} \frac{\partial \phi_1}{\partial a_1^k} & \frac{\partial \phi_2}{\partial a_1^k} & \dots & \frac{\partial \phi_m}{\partial a_1^k} \\ \frac{\partial \phi_1}{\partial a_2^k} & \frac{\partial \phi_2}{\partial a_2^k} & \dots & \frac{\partial \phi_m}{\partial a_2^k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \phi_1}{\partial a_{k_1}^k} & \frac{\partial \phi_2}{\partial a_{k_1}^k} & \dots & \frac{\partial \phi_m}{\partial a_{k_1}^k} \end{bmatrix} \quad (14)$$

δ^k for all k can be achieved by the following backpropagation equation.

$$\delta^k = ((\mathbf{W}^{k+1})^T \delta^{k+1}) \odot h'(a^k) \quad \forall k = 1, 2, \dots, n-1 \quad (15)$$

Where \odot stands for the element wise multiplication of a column vector to every column of the matrix.

$$\delta^n = \begin{bmatrix} h'(a_1^n) & 0 & \dots & 0 \\ 0 & h'(a_2^n) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & h'(a_m^n) \end{bmatrix} \quad (16)$$

Defining the term δ in such a away, we can rewrite the regularizer term in equation (12) as following:

$$\sum_{ij} \|(\mathbf{W}^1(:, i)) - \mathbf{W}^1(:, j)\|^T \delta^1\|^2 S_{ij} \quad (17)$$

If the network only has one hidden layer, we can derive derivative of the regularizer with respect to weights using δ and (15). When hidden layer's number is more than one, we need to introduce two more term, one to the backward path and one to the forward path: Define \mathbf{G}^k as the jacobian of pre-activation unit at layer k with respect to pre-activation at first hidden layer, note layer $k = 1$ corresponding to first hidden layer.

$$G_{mg}^k = \frac{\partial a_m^k}{\partial a_g^1} \quad \forall k = 1, 2, 3, \dots, n \quad (18)$$

We know that:

$$G_{mg}^1 = \frac{\partial a_m^1}{\partial a_g^1} = \begin{cases} 1 & \text{if } m=g \\ 0 & \text{others} \end{cases} \quad (19)$$

And \mathbf{G}^k for all k can be achieved during forward path by the following forward propagation equation and \mathbf{G}^1

$$G_{mg}^k = \sum_l W_{ml}^k G_{lg}^{k-1} h'(a_l^{k-1}) \quad \forall k = 2, 3, \dots, n \quad (20)$$

Define \mathbf{B}^k which gives the derivative of the δ^k with respect to the pre-activation units in the first hidden layers:

$$B_{lg}^k = \frac{\partial \delta_{lg}^k}{\partial a_g^1} \quad \forall k = 1, 2, \dots, n \quad (21)$$

We know that:

$$B_{lg}^n = \frac{\partial \delta_{lg}^n}{\partial a_g^1} = h''(a_l^n) \mathbf{1}_{lg} G_{lg}^n \quad (22)$$

\mathbf{B}^k for all k can be obtained by the following propagating equation during backward path using \mathbf{B}^n as following:

$$B_{lg}^k = h''(a_l^k) G_{lg}^k \sum_p \delta_{pj}^{k+1} W_{pl}^{k+1} + h'(a_l^k) \sum_p W_{pl}^{k+1} B_{pjg}^{k+1} \quad (23)$$

$$\forall k = 1, 2, \dots, n-1$$

Finally, the gradient of the regularizer, i.e. second term of the equation (12), can be calculated as follwoing:

For $k = 1$, i.e. first hidden layer:

$$\begin{aligned} \frac{\partial R}{\partial W_{lm}^1} &= 4\lambda_1 \sum_s S_{ms} \sum_j (\mathbf{W}^1(:, m) - \mathbf{W}^1(:, s))^T \delta^1(:, j) \delta^1(l, j) \\ &+ 2\lambda_1 \sum_{ks} S_{ks} \sum_j (\mathbf{W}^1(:, k) - \mathbf{W}^1(:, s))^T \delta^1(:, j) \sum_g (W^1(g, k) - W^1(g, s)) B_{lg}^1 z_m^0 \end{aligned} \quad (24)$$

For $k = 2, \dots, n$:

$$\begin{aligned} \frac{\partial R}{\partial W_{lm}^k} &= 2\lambda_1 \sum_{ks} S_{ks} \sum_j (\mathbf{W}^1(:, k) - \mathbf{W}^1(:, s))^T \delta^1(:, j) \\ &\sum_g (W^1(g, k) - W^1(g, s)) (z_m^{k-1} B_{lg}^k + \delta_{lg}^k h'(a_m^{k-1}) G_{mg}^{k-1}) \end{aligned} \quad (25)$$

Gradient with respect to bias term, for all $k = 1, \dots, n$:

$$\begin{aligned} \frac{\partial R}{\partial b_{lm}^k} = & 2\lambda_1 \sum_{ks} S_{ks} \sum_j (\mathbf{W}^1(:, k) - \mathbf{W}^1(:, s))^T \boldsymbol{\delta}^1(:, j) \\ & \sum_g (W^1(g, k) - W^1(g, s)) B_{l j g}^k \end{aligned} \quad (26)$$

The gradient of the first part of the objective which is some loss function we chose, is same as in the standard Back-propagation algorithm, here we just need to rewrite it in terms of the newly defined $\boldsymbol{\delta}$. For example, if we use sigmoid on all layers as activation function and cross entropy loss, we have the following:

$$E = - \sum_{i=1}^m (y_i \log \phi(x)_i + (1 - y_i) \log(1 - \phi(x)_i)) \quad (27)$$

$$\frac{\partial E}{\partial \mathbf{W}^k} = \boldsymbol{\delta}^k \frac{\phi - \mathbf{y}}{\phi(1 - \phi)} (\mathbf{z}^{k-1})^T \quad (28)$$

$$\frac{\partial E}{\partial \mathbf{b}^k} = \boldsymbol{\delta}^k \frac{\phi - \mathbf{y}}{\phi(1 - \phi)} \quad (29)$$

Now we can find the gradient of the loss with respect to weights in all layers. Compared to the conventional back propagation algorithm, except we have $\boldsymbol{\delta}$ term which is defined differently than the conventional backprop algorithm, we have one more extra term \mathbf{B}^k to add to the backward path and one more term \mathbf{G}^h to the forward path.