HW2 Individual

Jimmy Ye
CSE 190: Neural Networks
University of California, San Diego
jiy162@ucsd.edu

1

Prove that finding the optimal parameter w for linear regression for modeling $t = h(x) + \epsilon$ with ϵ Gaussian-random is equivalent to finding the w^* that minimizes the SSE.

First, we assume that we are looking for the optimal parameter in regards to maximum likelihood, which is equivalent to minimizing the negative log likelihood:

$$
\begin{aligned}
w^* &= \operatorname{argmin}_w - \sum_{n=1}^{N} (\ln p(t^n|x^n) + \ln p(x^n)) && && (1)\\
&= \operatorname{argmin}_w - \sum_{n=1}^{N} (\ln p(t^n|x^n)) && p(x^n) \text{ constant w.r.t. } w && (2)\\
&= \operatorname{argmin}_w - \sum_{n=1}^{N} \Big(\ln\Big(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t^n - h(x^n))^2}{2\sigma^2}} \Big)\Big) && \text{By distribution of } t && (3)\\
&= \operatorname{argmin}_w - \sum_{n=1}^{N} \Big(\ln\Big(\frac{1}{\sqrt{2\pi\sigma^2}} \Big) - \frac{(t^n - h(x^n))^2}{2\sigma^2} \Big) && \text{Logarithm properties} && (4)\\
&= \operatorname{argmin}_w - \sum_{n=1}^{N} \Big(- \frac{(t^n - h(x^n))^2}{2} \Big) && \text{Constants w.r.t } w && (5)\\
&= \operatorname{argmin}_w \frac{1}{2} \sum_{n=1}^{N} (t^n - h(x^n))^2 && && (6)
\end{aligned}
$$

2

For this problem, we are using a network with three layers: an input layer, a hidden layer consisting of J units with the sigmoid activation function, and a softmax output layer.

Note that we will use E instead of E^n in (2a) and (2b) to simplify notation.

2.1 Derivation

Derive the expression for δ for both the units of output layer (δ_k) and the hidden layer (δ_j). Recall that the definition of δ is $\delta_i = -\frac{\partial E}{\partial a_i}$, where a_i is the weighted sum of the inputs to unit i.

For the output layer:

$$-\frac{\partial E}{\partial w_{jk}} = -\frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial w_{jk}} \qquad \text{Chain rule} \tag{7}$$

$$= -\frac{\partial E}{\partial a_k} y_j \tag{8}$$

$$= (t_k - y_k) y_j \qquad \text{From previous HW: softmax regression} \tag{9}$$

$$\delta_k = -\frac{\partial E}{\partial a_k} = t_k - y_k \tag{10}$$

For the hidden layer:

$$\delta_j = -\frac{\partial E}{\partial a_j} \tag{11}$$

$$= -\sum_{k \in \text{outputs}} \Big(\frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial a_j}\big) \qquad \text{Chain rule} \tag{12}$$

$$= \sum_{k \in \text{outputs}} \left(\delta_k \frac{\partial a_k}{\partial a_j}\right) \tag{13}$$

$$= \sum_{k \in \text{outputs}} \left(\delta_k \frac{\partial a_k}{\partial y_j}\frac{\partial y_j}{\partial a_j}\right) \qquad \text{Chain rule} \tag{14}$$

$$= \sum_{k \in \text{outputs}} (\delta_k w_{jk} y_j') \tag{15}$$

$$= y_j' \sum_{k \in \text{outputs}} (\delta_k w_{jk}) \tag{16}$$

$$= y_j(1 - y_j) \sum_{k \in \text{outputs}} (\delta_k w_{jk}) \qquad \text{From previous HW: logistic regression} \tag{17}$$

2.2 Update rule

Derive the update rule for w_{ij} and w_{jk} using learning rate α, starting with the gradient descent rule. The derivative should take into account all of the outputs.

Starting with gradient descent:

$$w_{jk} = w_{jk} - \alpha\frac{\partial E}{\partial w_{jk}} \tag{18}$$

$$= w_{jk} - \alpha\frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial w_{jk}} \qquad \text{Chain rule} \tag{19}$$

$$= w_{jk} + \alpha\delta_k y_j \tag{20}$$

And similarly, for w_{ij}, we get $w_{ij} + \alpha\delta_j x_i$.

Plugging in δ from the previous part, for the output layer we get the update rule

$$w_{jk} = w_{jk} + \alpha(t_k - y_k)y_j$$

and for the hidden layer we get

$$w_{ij} = w_{ij} + \alpha x_i y_j(1 - y_j) \sum_{k \in \text{outputs}} (\delta_k w_{jk})$$

2.3 Vectorize computation

The computation is much faster when you update all the weights at the same time using matrix multiplication rather than for loops. Please show the update rule for the weight matrix from hidden layer to output layer and the matrix from input layer to hidden layer, using matrix/vector notation.

Notes:
y_j is now used for the vector of the hidden layer's outputs, rather than one specific hidden unit's.
t is used for the vector of the target output, and y is used for the vector of the output layer's outputs.
1 is the $j \times 1$ vector with all elements equal to 1.
$A \circ B$ is the element-wise product of vectors/matrices.

Let W^0 and W^1 be the input-to-hidden and hidden-to-output weight matrices, respectively.

Then the hidden-to-output matrix update rule is:

$W^1 = W^1 + \alpha y_j(t - y)^\top$

And the input-to-hidden matrix update rule is:

$W^0 = W^0 + \alpha x(y_j \circ (\mathbf{1} - y_j) \circ W^1(t - y))^\top$