حل تکلیف چهارم

طبقه‌بندی کننده‌های غیر خطی

(1)

(a) We have $y_k(i) = \hat{y}_k(i) + \epsilon_k(i)$, where $\hat{y}$ is actual output, y is target output, and ϵ is absolute error. Therefore,

$$\begin{aligned} J &= -\sum_{i=1}^{N}\sum_{k=1}^{k_L} y_k(i)\ln\frac{\hat{y}_k(i)}{y_k(i)} \\ &= \sum_{i=1}^{N}\sum_{k=1}^{k_L} y_k(i)\ln\frac{y_k(i)}{\hat{y}_k(i)} \\ &= \sum_{i=1}^{N}\sum_{k=1}^{k_L} y_k(i)\ln\frac{\hat{y}_k(i)+\epsilon_k(i)}{\hat{y}_k(i)} \\ &= \sum_{i=1}^{N}\sum_{k=1}^{k_L} y_k(i)\ln\left(1+\frac{\epsilon_k(i)}{\hat{y}_k(i)}\right) \end{aligned}$$

where, $\frac{\epsilon_k(i)}{\hat{y}_k(i)}$ is relative error, thus, J is a function of the relative error, not the absolute one.

(b) If $y_k = \hat{y}_k$, we have $\epsilon_k = 0$, and

$$J = -\sum_{i=1}^{N}\sum_{k=1}^{k_L} y_k(i)\ln(1) = 0.$$

Since $y_k = 0$ or 1 , $0 \le \hat{y}_k \le 1$, and $\epsilon_k(i) = y_k(i) - \hat{y}_k(i)$,

- if $y_k = 0$, then $J = 0$;
- if $y_k = 1 \Rightarrow 0 \le \epsilon_k \le 1 \Rightarrow y_k \ln(1+\epsilon_k/\hat{y}_k) > 0 \Rightarrow J > 0$

Therefore,

$$\min J = 0$$

(2)
(backpropagation with cross-entropy)

The cross entropy is

$$J = -\sum_{i=1}^{N}\sum_{k=1}^{k_L} y_k(i)\ln\left(\frac{\hat{y}_k(i)}{y_k(i)}\right).$$

Thus we see that $\mathcal{E}(i)$ in this case is given by

$$\mathcal{E}(i) = -\sum_{k=1}^{k_L} y_k(i)\ln\left(\frac{\hat{y}_k(i)}{y_k(i)}\right).$$

Thus we can evaluate $\delta_j^L(i)$ as

$$\delta_j^L(i) \equiv \frac{\partial\mathcal{E}(i)}{\partial v_j^L(i)} = \frac{\partial}{\partial v_j^L(i)}\left[-\sum_{k=1}^{k_L} y_k(i)\ln\left(\frac{f(v_k^L)}{y_k(i)}\right)\right].$$

This derivative will select the $k = j$th element out of the sum and gives

$$\delta_j^L(i) = -y_j(i)\frac{\partial}{\partial v_j^L(i)}\left(\ln\left(\frac{f(v_j^L)}{y_k(i)}\right)\right) = -y_j(i)\frac{f'(v_j^L)}{f(v_j^L)}.$$

If the activation function $f(\cdot)$ is the sigmoid function Equation 99 then its derivative is given in Equation 100 where we have $f'(v_j^L) = -af(v_j^L)(1-f(v_j^L))$ and the above becomes

$$\delta_j^L(i) = ay_j(i)(1-f(y_j^L)) = ay_j(i)(1-\hat{y}_j(i)).$$

When our activation function $f(x)$ is the *sigmoid function* defined by

$$f(x) = \frac{1}{1+e^{-ax}}, \tag{99}$$

we find its derivative given by

$$\begin{aligned} f'(x) &= \frac{-(-a)}{(1+e^{-ax})^2}e^{-ax} = f(x)(a)\frac{e^{-ax}}{1+e^{-ax}} = af(x)\left[\frac{1+e^{-ax}-1}{1+e^{-ax}}\right] \\ &= af(x)(1-f(x)). \end{aligned} \tag{100}$$

With all of these pieces we are ready to specify the *backpropagation algorithm.*

(3)
(backpropagation with softmax)

The softmax activation function has its output $\hat{y}_k$ given by

$$\hat{y}_k \equiv \frac{\exp(v_k^L)}{\sum_{k'} \exp(v_{k'}^L)}\,.$$

Note that this expression depends on v_j^L in both the numerator and the denominator. Using the result from the previous exercise we find

$$\begin{aligned}
\delta_j^L(i) &\equiv \frac{\partial \mathcal{E}(i)}{\partial v_j^L(i)} \\
&= \frac{\partial}{\partial v_j^L(i)} \left(-\sum_{k=1}^{k_L} y_k(i) \ln\left(\frac{\hat{y}_k}{y_k(i)}\right)\right) \\
&= -\frac{\partial}{\partial v_j^L(i)} \left(y_j(i) \ln\left(\frac{\hat{y}_j}{y_j(i)}\right)\right) - \frac{\partial}{\partial v_j^L(i)} \left(\sum_{k=1;k\neq j}^{k_L} y_k(i) \ln\left(\frac{\hat{y}_k}{y_k(i)}\right)\right) \\
&= -\frac{y_j(i)}{\hat{y}_j} \frac{\partial \hat{y}_j}{\partial v_j^L(i)} - \sum_{k=1;k\neq j}^{k_L} \frac{y_k(i)}{\hat{y}_k} \frac{\partial \hat{y}_k}{\partial v_j^L(i)}\,.
\end{aligned}$$

To evaluate this we first consider the first term or $\frac{\partial \hat{y}_j}{\partial v_j^L(i)}$ where we find

$$\begin{aligned}
\frac{\partial \hat{y}_j}{\partial v_j^L(i)} &= \frac{\partial}{\partial v_j^L(i)} \left(\frac{\exp(v_j^L)}{\sum_{k'} \exp(v_{k'}^L)} \right) \\
&= \frac{\exp(v_j^L)}{\sum_{k'} \exp(v_{k'}^L)} - \frac{\exp(v_j^L)\exp(v_j^L)}{\left(\sum_{k'} \exp(v_{k'}^L)\right)^2} = \hat{y}_j - \hat{y}_j^2\,.
\end{aligned}$$

While for the second term we get (note that $j \neq k$)

$$\frac{\partial \hat{y}_k}{\partial v_j^L(i)} = \frac{\partial}{\partial v_j^L} \left(\frac{\exp(v_k^L)}{\sum_{k'} \exp(v_{k'}^L)} \right) = -\frac{\exp(v_k^L)\exp(v_j^L)}{\left(\sum_{k'} \exp(v_{k'}^L)\right)^2} = -\hat{y}_k \hat{y}_j\,.$$

Thus we find

$$\begin{aligned}
\delta_j^L &= -\frac{y_j(i)}{\hat{y}_j}(\hat{y}_j - \hat{y}_j^2) - \sum_{k=1;k\neq j}^{k_L} \frac{y_k(i)}{\hat{y}_k}(-\hat{y}_k \hat{y}_j) \\
&= -y_j(i)(1-\hat{y}_j) + \hat{y}_j \sum_{k=1;k\neq j}^{k_L} y_k(i)\,.
\end{aligned}$$

Since $\hat{y}_k(i)$ and $y_k(i)$ are probabilities of class membership we have

$$\sum_{k=1}^{k_L} y_k(i) = 1\,,$$

and thus $\sum_{k=1;k\neq j}^{k_L} y_k(i) = 1 - y_j(i)$. Using this we find for $\delta_j^L(i)$ that

$$\delta_j^L(i) = -y_j(i) + y_j(i)\hat{y}_j + \hat{y}_j(1-y_j) = \hat{y}_j - y_j(i)\,,$$

the expression we were to show.

(4)
(the maximum number of polyhedral regions)

$$M = \sum_{m=0}^{l} \left(\begin{array}{c} k \\ m \end{array} \right) \quad \text{with} \quad \left(\begin{array}{c} k \\ m \end{array} \right) = 0 \quad \text{if} \quad m > k\,.$$

where M is the maximum number of polyhedral regions possible for a neural network with one hidden layer containing k neurons and an input feature dimension of l. Assuming that $l \geq k$ then

$$M = \sum_{m=0}^{l} \left(\begin{array}{c} k \\ m \end{array} \right) = \sum_{m=0}^{k} \left(\begin{array}{c} k \\ m \end{array} \right) = 2^k\,,$$

where we have used the fact that $\left(\begin{array}{c} k \\ m \end{array} \right) = 0$ to drop all terms in the sum when $m = k+1, k+2, \cdots, l$ if there are any. That the sum of the binomial coefficients sums to 2^k follows from expanding $(1+1)^k$ using the binomial theorem.

(5)
(an iteration dependent learning parameter μ)

A Taylor expansion of $\frac{1}{1+\frac{t}{t_0}}$ or

$$\frac{1}{1+\frac{t}{t_0}} \approx 1 - \frac{t}{t_0} + \frac{t^2}{t_0^2} + \cdots\,.$$

Thus when $t \ll t_0$ the fraction $\frac{1}{1+\frac{t}{t_0}} \approx 1$ to leading order and thus $\mu \approx \mu_0$. On the other hand when $t \gg t_0$ we have that $1 + \frac{t}{t_0} \approx \frac{t}{t_0}$ and the fraction above is given by

$$\frac{1}{1+\frac{t}{t_0}} \approx \frac{1}{\frac{t}{t_0}} = \frac{t_0}{t}\,.$$

Thus in this stage of the iterations $\mu(t)$ decreases inversely in proportion to t.

(6)
(when $N = 2(l+1)$ the number of dichotomies is 2^{N-1})

We have

$$O(N,l) = 2\sum_{i=0}^{l}\left(\begin{array}{c} N-1 \\ i \end{array}\right),$$

where N is the number of points embedded in a space of dimension l and $O(N,l)$ is the number of groupings that can be formed by hyperplanes in $\mathbb{R}^l$ to separate the points into two classes. If $N = 2(l+1)$ then

$$O(2(l+1),l) = 2\sum_{i=0}^{l}\left(\begin{array}{c} 2l+1 \\ i \end{array}\right).$$

Given the identity

$$\left(\begin{array}{c} 2n+1 \\ n-i+1 \end{array}\right) = \left(\begin{array}{c} 2n+1 \\ n+i \end{array}\right),$$

by taking $i = n+1, n, n-1, \cdots, 1$ we get the following equivalences

$$\begin{array}{rcl}
\left(\begin{array}{c} 2n+1 \\ 0 \end{array}\right) & = & \left(\begin{array}{c} 2n+1 \\ 2n+1 \end{array}\right) \\
\left(\begin{array}{c} 2n+1 \\ 1 \end{array}\right) & = & \left(\begin{array}{c} 2n+1 \\ 2n \end{array}\right) \\
\left(\begin{array}{c} 2n+1 \\ 2 \end{array}\right) & = & \left(\begin{array}{c} 2n+1 \\ 2n-1 \end{array}\right) \\
 & \vdots & \\
\left(\begin{array}{c} 2n+1 \\ n-1 \end{array}\right) & = & \left(\begin{array}{c} 2n+1 \\ n+2 \end{array}\right) \\
\left(\begin{array}{c} 2n+1 \\ n \end{array}\right) & = & \left(\begin{array}{c} 2n+1 \\ n+1 \end{array}\right)
\end{array}$$

Now write $O(2(l+1),l)$ as

$$\sum_{i=0}^{l}\left(\begin{array}{c} 2l+1 \\ i \end{array}\right) + \sum_{i=0}^{l}\left(\begin{array}{c} 2l+1 \\ i \end{array}\right),$$

or two sums of the same thing. Next note that using the above identities we can write the second sum as

$$\begin{aligned}
\sum_{i=0}^{l}\left(\begin{array}{c} 2l+1 \\ i \end{array}\right) &= \left(\begin{array}{c} 2l+1 \\ 0 \end{array}\right) + \left(\begin{array}{c} 2l+1 \\ 1 \end{array}\right) + \cdots + \left(\begin{array}{c} 2l+1 \\ l-1 \end{array}\right) + \left(\begin{array}{c} 2l+1 \\ l \end{array}\right) \\
&= \left(\begin{array}{c} 2l+1 \\ 2l+1 \end{array}\right) + \left(\begin{array}{c} 2l+1 \\ 2l \end{array}\right) + \cdots + \left(\begin{array}{c} 2l+1 \\ l+2 \end{array}\right) + \left(\begin{array}{c} 2l+1 \\ l+1 \end{array}\right) \\
&= \sum_{i=l+1}^{2l+1}\left(\begin{array}{c} 2l+1 \\ i \end{array}\right).
\end{aligned}$$

Thus using this expression we have that

$$O(2(l+1),l) = \sum_{i=0}^{l}\left(\begin{array}{c} 2l+1 \\ i \end{array}\right) + \sum_{i=l+1}^{2l+1}\left(\begin{array}{c} 2l+1 \\ i \end{array}\right) = \sum_{i=0}^{2l+1}\left(\begin{array}{c} 2l+1 \\ i \end{array}\right) = 2^{2l+1}.$$

Since $2l+1 = N-1$ we have that $O(2(l+1),l) = 2^{N-1}$ as we were to show.

(7)

(the kernel trick)

From the given mapping $\phi(x)$ we have that

$$\begin{aligned} y_i^T y_j &= \phi(x_i)^T \phi(x_j) \\ &= \frac{1}{2} + \cos(x_i)\cos(x_j) + \cos(2x_i)\cos(2x_j) + \cdots + \cos(kx_i)\cos(kx_j) \\ &+ \sin(x_i)\sin(x_j) + \sin(2x_i)\sin(2x_j) + \cdots + \sin(kx_i)\sin(kx_j)\,. \end{aligned}$$

Since $\cos(\alpha)\cos(\beta)+\sin(\alpha)\sin(\beta) = \cos(\alpha-\beta)$ we can match cosigns with sines in the above expression and simplify a bit to get

$$y_i^T y_j = \frac{1}{2} + \cos(x_i - x_j) + \cos(2(x_i - x_j)) + \cdots + \cos(k(x_i - x_j))\,.$$

To evaluate this sum we note that by writing the cosigns above in terms of their exponential representation and using the geometric series we can show that

$$1 + 2\cos(\alpha) + 2\cos(2\alpha) + 2\cos(3\alpha) + \cdots + 2\cos(n\alpha) = \frac{\sin\left(\left(n+\frac{1}{2}\right)\alpha\right)}{\sin\left(\frac{x}{2}\right)}\,.$$

Thus using this we can show that $y_i^T y_j$ is given by

$$\frac{1}{2}\frac{\sin\left(\left(k+\frac{1}{2}\right)(x_i - x_j)\right)}{\sin\left(\frac{x}{2}\right)}\,,$$

as we were to show.