A Training Process by Stochastic Gradient Descent Approach from a Single Layer Model to a Consequence : Sigmoid Activation Function for Instance

Shaun Leo

School of Public Affairs and Administration, University of Electronic Science and Technology of China

Contents

1 The processing of template

1.1 A brief preview

The normal process to train a single-layered neural network model from A to Z with each epoch of weight updating included usually appears to be:

1. Input the data $\begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$.
2. Generate the output data $\begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix}$.
3. Update the present weight $\begin{bmatrix} w_1 & \cdots & w_n \end{bmatrix}$ according to the offset from output data in the step 2 to the label assigned along with the input data.
4. Repeat the epoch from step 2 to step 3 till the network is able to mimicry the ideal model.

1.2 Generating a datum for next input from the previous layer

As for a single layer model, all the neurons in the single layer make a difference to a certain consequence for output. Considering the single layer $\mathbf{w}$ and the input data $\mathbf{x}$ as follows.

$$\mathbf{w} = \begin{bmatrix} w_1 & \cdots & w_n \end{bmatrix} \tag{1}$$

$$\mathbf{x} := \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \tag{2}$$

The single layer $\mathbf{w}$ is able to generate a certain consequence y its activation function $\varphi(x)$. The consequence for output, y, thus appears to be

$$y = \varphi(v) \tag{3}$$

$$v = \langle \mathbf{w}, \mathbf{x} \rangle \tag{4}$$

Here the vector $\mathbf{x}$ states the input data, locating in the same-dimensional space as the vector $\mathbf{w}$.

1.3 Calculating the offset from the output datum to the label

As for supervision learning, a label is assigned for measurement over the skewing from the label to factual output datum, usually marked as d in short. The skewing, marked as e in short, often considered error to be corrected by learning. The following equation states the relevance between the three parameters:

$$e = d - y \tag{5}$$

1.4 Updating the weight of a certain layer with normalized delta principle

As the word *machine learning* states, the process of *learning* here perform as the dynamic updating of the weight to each neuron in the single layer. A mathematical approach marks this processing as

$$w_i \leftarrow w_i + \Delta w_i \tag{6}$$

According to **delta principle**[1], the variance Δw_i, which involve but a certain neuron amid the single layer, shall be generated during the process of updating by the step $\aleph$, the offset e and the input datum x_i as follows.

$$\Delta w_i = \alpha e x_i, \alpha \in (0, 1) \tag{7}$$

A normalized form contains an additional parameter δ, which lies as:

$$\Delta w_i = \alpha \delta e x_i, \alpha \in (0, 1) \tag{8}$$

where the parameter δ is

$$\begin{aligned} \delta &= \frac{\mathrm{d}}{\mathrm{d\,x}} \varphi(v) \\ &= \frac{\mathrm{d}}{\mathrm{d\,x}} \varphi\big(\langle \mathbf{w},\ \mathbf{x} \rangle\big) \\ &= \frac{\mathrm{d}}{\mathrm{d\,x}} \varphi\left(\langle \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}, \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \rangle \right) \\ &= \frac{\mathrm{d}}{\mathrm{d\,x}} \varphi(w_1 x_1 + \cdots + w_n x_n) \\ &= \prod_{k=M}^{n} \frac{\partial}{\partial x_j} \varphi^{(k)}(v), j \in \{1, \ldots, n\} \end{aligned} \tag{9}$$

While taking the derivative of the activation function $\varphi\big(\langle \mathbf{w},\ \mathbf{x} \rangle\big)$, considering it a M-th order differential function.

[1] As for more acknowledgment upon **delta principle**, **gradient descent algorithm** and **optimization strategy**, please check additional reference.

1.5 Summery

Take a review of all the steps above, conclude a weight-updating algorithm as follows.

1. $\mathbf{x} := \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$
2. $\mathbf{y} = \varphi\big(\langle \mathbf{x}, \mathbf{w} \rangle\big)$
3. $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$. Besides,
$$\begin{cases} e = y - d_i \\ \Delta w_i = \alpha \delta e x_i, \alpha \in (0,1) \\ \delta = \frac{\mathrm{d}}{\mathrm{d}\ \mathrm{x}} \varphi(v) \end{cases}$$
4. Repeat the epoch from step 2 to step 3 till the offset e reach the ideal value.

2 The processing of instance

2.1 Instantiating of Sigmoid activation function

Here we take the Sigmoid function for instance, which is defined as follows.

$$\varphi(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{10}$$

Sigmoid function involves first order derivative which lies

$$\varphi'(x) = \varphi(x)\big(1 - \varphi(x)\big) \tag{11}$$

Thus the updating process is instantiated from the template equation **??** to

$$\begin{aligned} w_i \leftarrow w_i + \Delta w_i &= w_i + \alpha \delta e x_i \\ &= w_i + \alpha \varphi'(v)(y \qquad - \qquad d_i) x_i \\ &= w_i + \alpha \varphi(v)\big(1 - \varphi(v)\big) x_i, \varphi(v) = \big(\langle \mathbf{w}, \mathbf{x} \rangle\big), w_i \in \mathbf{w} \end{aligned} \tag{12}$$

with the previously defined equation 4, 8 and 11 inserted.

2.2 Instantiated algorithms

1. $\mathbf{x} := \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$
2. $y = \varphi(v) : v = \langle \mathbf{w}, \mathbf{x} \rangle, \varphi(x) = \frac{1}{1+\mathrm{e}^{-x}}$
3. $w_i \leftarrow w_i + \Delta w_i : \Delta w_i = \alpha \delta e x_i, \alpha \in (0,1), \delta = \varphi'(x) = \varphi(x)\big(1 - \varphi(x)\big), e = d - y, x_i \in \mathbf{x}$
4. Repeat the epoch from step 2 to step 3 till the offset e reach the ideal value.

3 Exercise

A single-layer network activated by Sigmoid function, with 3 neurons and 1 output, awaits training. There are several sets of data provided for training, in the form of $\begin{bmatrix} w_1, & w_2, & w_3, & \text{label} \end{bmatrix}$. Considering the initialized weight of the single layer lies $\begin{bmatrix} 0, & 0, & 0 \end{bmatrix}$ and the learning step set 0.8. Processing the details of the network learning in a mathematical approach with the following data set.

$$\begin{bmatrix} 0, & 0, & 1, & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0, & 1, & 1, & 0 \end{bmatrix}$$

$$\begin{bmatrix}1, & 0, & 1, & 1\end{bmatrix}$$
$$\begin{bmatrix}1, & 1, & 1, & 1\end{bmatrix}$$

solution With the first set of data generated the output data, which is described as

$$\begin{aligned}
y &= \varphi\left(\langle \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix} \rangle\right) \\
&= \varphi(0) \\
&= \frac{1}{2}
\end{aligned}$$

Thus able is to calculate the offset from the output $y = \frac{1}{2}$ to the label $d = 0$, which writes

$$\begin{aligned}
e &= \frac{1}{2} - 0 \\
&= \frac{1}{2}
\end{aligned}$$

Update the 3 neurons in the single layer as

$$\begin{aligned}
w_1 \leftarrow w_1 + \Delta w_1 &= 0 + 0.8 \times \frac{1}{2} \times \left(1 - \frac{1}{2}\right) \times \frac{1}{2} \times 0 \\
&= 0 \\
w_2 \leftarrow w_2 + \Delta w_2 &= 0 + 0.8 \times \frac{1}{2} \times \left(1 - \frac{1}{2}\right) \times \frac{1}{2} \times 0 \\
&= 0 \\
w_3 \leftarrow w_3 + \Delta w_3 &= 0 + 0.8 \times \frac{1}{2} \times \left(1 - \frac{1}{2}\right) \times \frac{1}{2} \times 1 \\
&= -0.1
\end{aligned}$$

Repeat the epoch above and get the following weight distribution.

$$\begin{matrix}
[& 0 & 0 & -0.1 &] \\
[& 0 & -0.10473378 & -0.20473378 &] \\
[& 0.08886445 & -0.10473378 & -0.11586933 &] \\
[& 0.18188286 & -0.01171538 & -0.02285093 &]
\end{matrix}$$

The data above is calculated by a custom Python program.