Machine learning : Sheet 4

Author : Djordje Zivanovic

1.

$$\frac{\partial \ell(\mathbf{W}, \mathbf{b}, \mathbf{x}, y)}{\partial \mathbf{a}} = -\frac{\partial \log a_y}{\partial \mathbf{a}} \tag{1}$$

$$= \left[-\frac{\partial \log a_y}{\partial a_1}, ..., -\frac{\partial \log a_y}{\partial a_y}, ..., -\frac{\partial \log a_y}{\partial a_C} \right] \tag{2}$$

$$= \left[0, ..., -\frac{1}{a_y}, ..., 0 \right] \tag{3}$$

The equation 1 is a definition of the objective function for a given point. The equation 2 is a definition of a gradient of a scalar function (derivative of scalar by a vector). The equation 3 uses property of derivative of a logarithm function.

$$\frac{\partial \ell(\mathbf{W}, \mathbf{b}, \mathbf{x}, y)}{\partial \mathbf{z}} = \frac{\partial \ell(\mathbf{W}, \mathbf{b}, \mathbf{x}, y)}{\partial \mathbf{a}} \cdot \frac{\partial \mathbf{a}}{\partial \mathbf{z}} \tag{4}$$

From the equation 4 we can see that we need to calculate $\frac{\partial \mathbf{a}}{\partial \mathbf{z}}$

$$\frac{\partial \mathbf{a}}{\partial \mathbf{z}} = \frac{\partial \left[\frac{e^{z_1}}{\sum_{l=1}^{C} e^{z_l}}, ..., \frac{e^{z_i}}{\sum_{l=1}^{C} e^{z_l}}, ..., \frac{e^{z_C}}{\sum_{l=1}^{C} e^{z_l}} \right]}{\partial \mathbf{z}} \tag{5}$$

$$= \begin{bmatrix} \frac{\partial \frac{e^{z_1}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_1} & \frac{\partial \frac{e^{z_1}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_2} & \cdots & \frac{\partial \frac{e^{z_1}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_C} \\ \frac{\partial \frac{e^{z_2}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_1} & \frac{\partial \frac{e^{z_2}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_2} & \cdots & \frac{\partial \frac{e^{z_2}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_C} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \frac{e^{z_C}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_1} & \frac{\partial \frac{e^{z_C}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_2} & \cdots & \frac{\partial \frac{e^{z_C}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_C} \end{bmatrix} \tag{6}$$

The equation 5 is softmax classifier equation given in the text. The equation 6 is a Jacobian of derivatives of vectors. Let us denote:

$$S_i = \frac{e^{z_i}}{\sum_{l=1}^{C} e^{z_l}}$$

Then the equation 6 becomes:

$$\begin{bmatrix} \frac{\partial S_1}{\partial z_1} & \frac{\partial S_1}{\partial z_2} & \cdots & \frac{\partial S_1}{\partial z_1} \\ \frac{\partial S_2}{\partial z_1} & \frac{\partial S_2}{\partial z_2} & \cdots & \frac{\partial S_2}{\partial z_C} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial S_C}{\partial z_1} & \frac{\partial S_C}{\partial z_2} & \cdots & \frac{\partial S_C}{\partial z_C} \end{bmatrix} \tag{7}$$

In the equation 7 we notice that we have only two "different" type of derivatives, $\frac{\partial S_i}{\partial z_i}, i \in \{1, ..., C\}$ and $\frac{\partial S_i}{\partial z_j}, i \neq j, i \in \{1, ..., C\}, j \in \{1, ..., C\}$.

$$\frac{\partial S_i}{\partial z_i} = \frac{\partial \frac{e^{z_i}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_i} \tag{8}$$

$$= \frac{e^{z_i} \cdot \sum_{l=1}^{C} e^{z_l} - e^{z_i} e^{z_i}}{\left(\sum_{l=1}^{C} e^{z_l} \right)^2} \tag{9}$$

$$= \frac{e^{z_i} \cdot \left(\sum_{l=1}^{C} e^{z_l} - e^{z_i} \right)}{\left(\sum_{l=1}^{C} e^{z_l} \right)^2} \tag{10}$$

$$= \frac{e^{z_i}}{\sum_{l=1}^{C} e^{z_l}} \frac{\sum_{l=1}^{C} e^{z_l} - e^{z_i}}{\sum_{l=1}^{C} e^{z_l}} \tag{11}$$

$$= S_i \cdot (1 - S_i) \tag{12}$$

The equation 8 uses S_i definition. The equation 9 uses derivative of division rule. The equations 10, 11, 12 are the simple transformations of the expressions and usage of definition S_i.

$$\frac{\partial S_i}{\partial z_j} = \frac{\partial \frac{e^{z_i}}{\sum_{l=1}^{C} e^{z_l}}}{\partial z_j} \tag{13}$$

$$= -\frac{e^{z_i} \cdot e^{z_j}}{\left(\sum_{l=1}^{C} e^{z_l}\right)^2} \tag{14}$$

$$= -S_i \cdot S_j \tag{15}$$

The equation 13 uses S_i definition. The equations 14 and 15 are simple arithmetic transformations of expressions expressions and usage of definition S_i. Finally we have:

$$\frac{\partial \mathbf{a}}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial S_1}{\partial z_1} & \frac{\partial S_1}{\partial z_2} & \dots & \frac{\partial S_1}{\partial z_1} \\ \frac{\partial S_2}{\partial z_1} & \frac{\partial S_2}{\partial z_2} & \dots & \frac{\partial S_2}{\partial z_C} \\ \dots & \dots & \dots & \dots \\ \frac{\partial S_C}{\partial z_1} & \frac{\partial S_C}{\partial z_2} & \dots & \frac{\partial S_C}{\partial z_C} \end{bmatrix}$$

$$= \begin{bmatrix} S_1 \cdot (1 - S_1) & -S_1 \cdot S_2 & \dots & -S_1 \cdot S_C \\ -S_2 \cdot S_1 & S_2 \cdot (1 - S_2) & \dots & -S_2 \cdot S_C \\ \dots & \dots & \dots & \dots \\ -S_C \cdot S_1 & -S_C \cdot S_1 & \dots & S_C \cdot (1 - S_C) \end{bmatrix} \tag{16}$$

The equation 16 uses equations 12 and 15. Further we have:

$$\frac{\partial \ell(\mathbf{W}, \mathbf{b}, \mathbf{x}, y)}{\partial \mathbf{z}} = \frac{\partial \ell(\mathbf{W}, \mathbf{b}, \mathbf{x}, y)}{\partial \mathbf{a}} \cdot \frac{\partial \mathbf{a}}{\partial \mathbf{z}} \tag{17}$$

$$= \left[0, ..., -\frac{1}{a_y}, ..., 0\right] \cdot \begin{bmatrix} S_1 \cdot (1 - S_1) & -S_1 \cdot S_2 & \dots & -S_1 \cdot S_C \\ -S_2 \cdot S_1 & S_2 \cdot (1 - S_2) & \dots & -S_2 \cdot S_C \\ \dots & \dots & \dots & \dots \\ -S_C \cdot S_1 & -S_C \cdot S_1 & \dots & S_C \cdot (1 - S_C) \end{bmatrix} \tag{18}$$

$$= \left[\frac{S_y \cdot S_1}{a_y}, \frac{S_y \cdot S_2}{a_y}, \dots, \frac{S_y \cdot (S_y - 1)}{a_y}, \dots, \frac{S_y \cdot S_C}{a_y}\right] \tag{19}$$

$$= [S_1, S_2, \dots, S_y - 1 \dots, S_C] \tag{20}$$

The equation 17 is just another way to write the same derivative. The equation 18 replaces derivatives using equations 3 and 16. The equation 19 is a matrix multiplication. The equation 20 is a a simplification of the equation 19 using the property that $a_i = S_i$ [1]. The Generalizing the formula $\frac{\partial \ell}{\partial w_{ij}^2} = \frac{\partial \ell}{\partial z_i^2} \cdot \frac{\partial z_i^2}{\partial w_{ij}^2} = \frac{\partial \ell}{\partial z_i^2} \cdot x_j$ we get:

$$\frac{\partial \ell(\mathbf{W}, \mathbf{b}, \mathbf{x}, y)}{\partial \mathbf{W}} = \left(\mathbf{x} \frac{\partial \ell}{\partial \mathbf{z}}\right)^{\mathrm{T}}$$

$$= \begin{bmatrix} x_1 \cdot S_1 & x_2 \cdot S_1 & \dots & x_D \cdot S_1 \\ x_1 \cdot S_2 & x_2 \cdot S_2 & \dots & x_D \cdot S_2 \\ \dots & \dots & \dots & \dots \\ x_1 \cdot (S_y - 1) & x_2 \cdot (S_y - 1) & \dots & x_D \cdot (S_y - 1) \\ \dots & \dots & \dots & \dots \\ x_1 \cdot S_C & x_2 \cdot S_C & \dots & x_D \cdot S_C \end{bmatrix} \tag{21}$$

The equation 21 is a matrix multiplication of matrices.

$$\frac{\partial \ell(\mathbf{W}, \mathbf{b}, \mathbf{x}, y)}{\partial \mathbf{b}} = \frac{\partial \ell(\mathbf{W}, \mathbf{b}, \mathbf{x}, y)}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{b}} \tag{22}$$

The equation 22 is just another way to write derivative. If we notice that $\frac{\partial \mathbf{z}}{\partial \mathbf{b}}$ is an identity matrix (b_i appears only in equality with the z_i), from the equation 22 we get that $\frac{\partial \ell(\mathbf{W}, \mathbf{b}, \mathbf{x}, y)}{\partial \mathbf{b}}$ is the same as $\frac{\partial \ell(\mathbf{W}, \mathbf{b}, \mathbf{x}, y)}{\partial \mathbf{z}}$. Using gradient descent for minibatch of B training examples we get the following update equations:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \frac{1}{B} \sum_{i=1}^{B} \frac{\partial \ell(\mathbf{x}_i, y_i, \mathbf{W}_t, \mathbf{b}_t)}{\partial \mathbf{w}}$$

[1] I saw really late that $S_i = a_i$

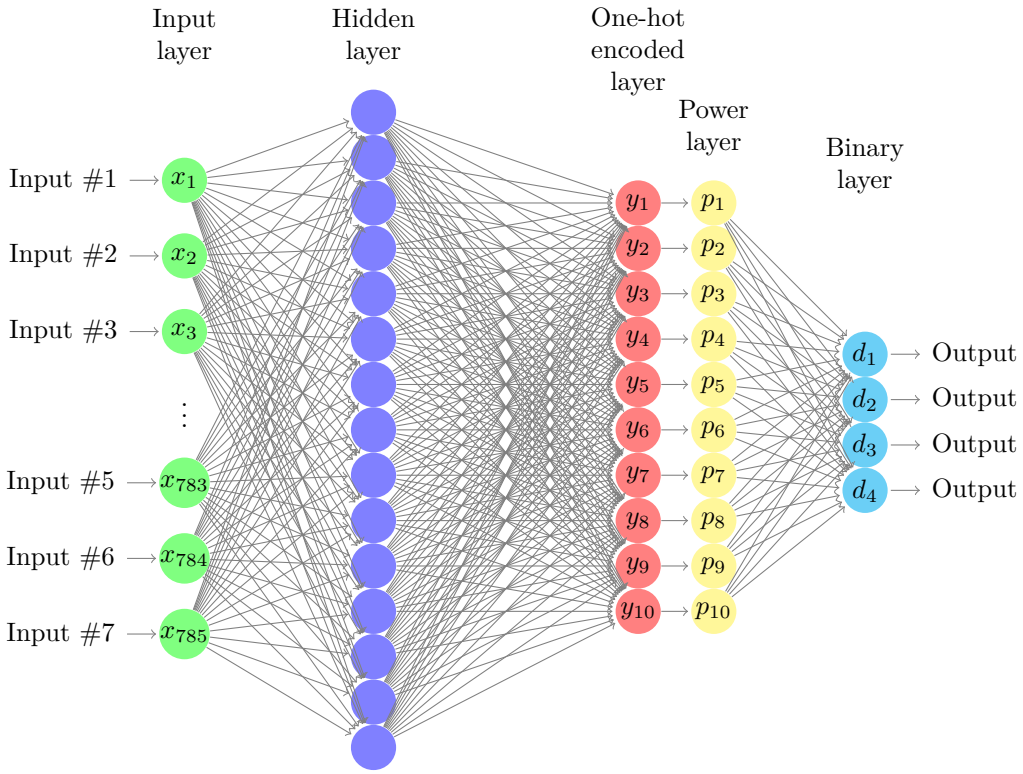

Figure 1: Neural network that transforms one hot-encoded output to binary output

$$\mathbf{b}_{t+1} = \mathbf{b}_t - \frac{1}{B}\sum_{i=1}^{B}\frac{\partial \ell(\mathbf{x}_i, y_i, \mathbf{W}_t, \mathbf{b}_t)}{\partial \mathbf{b}}$$

We just need to adjust orientation of $\mathbf{b}$ and the derivative in the last update rule so the addition is possible.

2. 1. I do not agree with the suggestion for several reasons. Let us notice that loss function of one training example $(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2$ is just a sum of squares of errors (component-wise) of wrongly calculated outputs. The first reason is that in the binary coding approach not all errors are equal. If we encode 0 as 0000 and our neural network outputs 7, the error will be 3. On the other hand for the output 1 the error will be 1. This means, not all errors are equal, and they are related to the encoding. This could lead to unnecessarily longer training. Moreover, it could lead to stimulating errors with "lower" value in order to decrease the number of errors with bigger values.
   The next problem is that our algorithm even when it is trained properly it could classify something incorrectly. For example in one hot encoding output with two ones is incorrect (we know there is some problem by seeing the output). On the other hand, in binary encoding we do not have "obvious" errors in encoding, because all outputs are valid (except the encodings for numbers bigger than 10). The final problem is that it could stimulate some neurons to have lower values because specific digit in encoding is not frequent enough. For example the first digit in binary encoding appears only in 2 numbers (9 and 10), but the network may believe it is not so important and it could decrease the value.
   2. On the figure 1 is shown the neural network that fulfills requested in the task. $p_i(x) = bs(x-p)$ is the activation function, which for $x > p$ returns 1, and for $x \leq p$ returns 0. We choose p that suits the best, $p = 0.5$ is a reasonable choice. Further, weight (p_i, d_j) is 1 only in those cases where in binary representation of number i on j-th place is 1. So for example for p_7 the weights towards d_1, d_2, d_3, d_4 will be $1, 1, 1, 0$, for p_9 the weights will be $1, 0, 0, 1$ etc. We can say we added two additional layers, because power layer has just activation functions binary step, and the input weights are all one. The output layer is just linear activation function, whose weights are previously stated. Why the network gives the correct output is obvious, because only in cases where $y_i > p$ p_i-th node will be 1, and further binary layer will generate corresponding representation of the i-th number.
   3. The same problem as for the first subproblem of this problem, because we minimize the objective functions, thus the different errors will have the different influence on nodes.

   1. sdfsdfdf