Softmax

1c) Prove that $\text{softmax}(x) = \text{softmax}(x+c)$ where $\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$

$$\text{softmax}(x+c)_i = \frac{e^{x_i+c}}{\sum_j e^{x_j+c}}$$

$$\Leftrightarrow \text{softmax}(x+c)_i = \frac{e^{x_i} \cdot e^{c}}{\sum_j e^{x_j} \cdot e^{c}}$$

$$\Leftrightarrow \text{softmax}(x+c)_i = \frac{e^{x_i} \cdot e^{c}}{e^{c} \sum_j e^{x_j}}$$

$$\Leftrightarrow \text{softmax}(x+c)_i = \frac{e_c}{e_c} \cdot \frac{e^{x_i}}{\sum_j e^{x_j}}$$

$$\Leftrightarrow \text{softmax}(x+c)_i = 1 \cdot \frac{e^{x_i}}{\sum_j e^{x_j}} = \text{softmax}(x)_i$$

Neural Network Basics

(2a) Derive the gradients of the sigmoid function and show that it can be rewritten as a function of the function value

The sigmoid function is given by

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

But we can also rewrite it as

$$\sigma(x) = v(z(x)) \qquad \text{where} \qquad v(z) = \frac{1}{z}$$
$$z(x) = 1 + e^{-x}$$

The derivative can thus be written as

$$\sigma'(x) = v'(z(x)) \cdot z'(x) \qquad \text{where} \qquad v'(z) = -\frac{1}{z^2}$$
$$z'(x) = -e^{-x}$$

Therefore, inserting $v'(z)$ and $z'(x)$ we get:

$$\sigma'(x) = -\frac{1}{(1+e^{-x})^2} \cdot -e^{-x}$$

$$\Leftrightarrow \quad \sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2}$$

Keeping in mind that $\sigma(x) = \frac{1}{1+e^{-x}}$ and considering that we can rewrite e^{-x} as $e^{-x} = \frac{1}{\sigma(x)} - 1$ we can rewrite $\sigma'(x)$ as a function of $\sigma(x)$

$$\sigma'(x) = \sigma(x)^2 \cdot \left(\frac{1}{\sigma(x)} - 1\right)$$

$$\Leftrightarrow \quad \sigma'(x) = \sigma(x) - \sigma(x)^2$$

$$\Leftrightarrow \quad \sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$$

Neural Networks

2b) Derive the gradient with regards to the inputs of a softmax function when cross entropy loss is used for evaluation, i.e. find the gradients with respect to the softmax input vector θ, when the prediction is made by $\hat{y} = \text{softmax}(\theta)$

Cross Entropy : $CE(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i)$

Softmax : $\text{softmax}(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$

Instead of writing θ for all the inputs we will take the derivative of input vector x

$$\frac{\partial CE}{\partial x_k} = -\sum_i y_i \frac{1}{\hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial x_k}$$

Therefore we first need the derivative of the softmax function with respect to x_k. For this we will differentiate between the two cases where $i=k$ and $i \neq k$

Case 1: $i = k$

$$\frac{\partial \hat{y}_i}{\partial x_k} = \frac{e^{x_k}}{\sum_j e^{x_j}} - \frac{e^{x_k}}{(\sum_j e^{x_j})^2} \cdot e^{x_k}$$

$$\frac{\partial \hat{y}_i}{\partial x_k} = \hat{y}_k - \hat{y}_k^2$$

$$\boxed{\frac{\partial \hat{y}_i}{\partial x_k} = \hat{y}_k (1 - \hat{y}_k)}$$

Case 2: $i \neq k$

$$\frac{\partial \hat{y}_i}{\partial x_k} = -\frac{e^{x_i}}{(\sum_j e^{x_j})^2} \cdot e^{x_k}$$

$$\frac{\partial \hat{y}_i}{\partial x_k} = -\frac{e^{x_i}}{\sum e^{x_j}} \cdot \frac{e^{x_k}}{\sum e^{x_j}}$$

$$\boxed{\frac{\partial \hat{y}_i}{\partial_k} = -\hat{y}_i \hat{y}_k}$$

Plugging in these derivatives we can get the following

$$\frac{\partial CE}{\partial x_k} = -y_k \cdot \frac{1}{\hat{y}_k} \cdot (\hat{y}_k(1-\hat{y}_k)) - \sum_{i \neq k} y_i \cdot \frac{1}{\hat{y}_i} \cdot (-\hat{y}_i \hat{y}_k)$$

$$\iff \frac{\partial CE}{\partial x_k} = -y_k + y_k \hat{y}_k + \sum_{i \neq k} y_i \hat{y}_k$$

$$\iff \frac{\partial CE}{\partial x_k} = -y_k + \sum_i y_i \hat{y}_k$$

cont'd →

$$\frac{\partial CE}{\partial x_h} = -y_h + \hat{y}_h \underbrace{\sum_i y_i}$$

This equals 1 because there is only on correct class label

$$\frac{\partial CE}{\partial x_h} = \hat{y}_h - y_h$$

Something missing

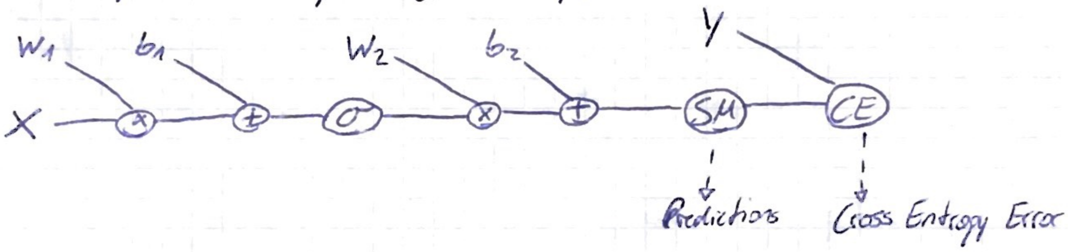
Neural Network Basics

(2c) Derive the gradients with respect to the inputs x to a one-hidden layer neural network (that is, find $\frac{\partial J}{\partial x}$ where $J = CE(y, \hat{y})$ is the cost function for the neural network). The neural network employs sigmoid activation function for the hidden layer, and softmax for the output layer. Assume the one-hot label vector is y, and cross entropy is used. (Feel free to use $\sigma'(x)$ as the shorthand for the sigmoid gradient, and feel free to define any variables whenever you see fit.)

The given functions for the neural net are:

$$J = CE(y, \hat{y}) \qquad \hat{y} = \text{softmax}(hW_2 + b_2) \qquad h = \text{sigmoid}(xW_1 + b_1)$$

The Computation Graph (Visual help) looks like this:



For the purpose of the task I will rewrite the functions as follows:

$$J = CE(y, \text{softmax}(\hat{y})) \qquad \hat{y} = hW_2 + b_2 \qquad h = \text{sigmoid}(z) \qquad z = xW_1 + b_1$$

The Gradients with respect to x can thus be defined as follows

$$\frac{\partial J}{\partial x} = \frac{\partial CE}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial x}$$
$$= \frac{\partial CE}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h} \frac{\partial h}{\partial x}$$
$$= \frac{\partial CE}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h} \frac{\partial h}{\partial z} \frac{\partial z}{\partial x}$$

where

$$\frac{\partial CE}{\partial \hat{y}} = \hat{y} - y$$
$$\frac{\partial \hat{y}}{\partial h} = W_2$$
$$\frac{\partial h}{\partial z} = \sigma'(z)$$
$$\frac{\partial z}{\partial x} = W_1$$

The Gradients are thus given by

$$\delta_1 = \hat{y} - y$$
$$\delta_2 = \delta_1 \cdot W_2^T$$
$$\delta_3 = \delta_2 \circ \sigma'(z)$$
$$\delta_4 = \delta_3 \cdot W_1^T \implies \boxed{\frac{\partial J}{\partial x} = \delta_3 \cdot W_1^T}$$

Neural Network Basics

(2d) How many parameters are there in this neural network (2c), assuming the input is D_x -dimensional, the output is D_y dimensional and there are H hidden units?

The trainable parameters of this Network are given by W_1, b_1, W_2, and b_2

The Dimensions of the input X and Output Y for a batch size of n are

$$X \in \mathbb{R}^{n \times D_x}$$
$$Y \in \mathbb{R}^{D_y \times n}$$

The dimensions of our parameters must thus be as follows

$$W_1 \in \mathbb{R}^{D_x \times H} \qquad W_2 \in \mathbb{R}^{H \times D_y}$$
$$b_1 \in \mathbb{R}^{1 \times H} \qquad b_2 \in \mathbb{R}^{1 \times D_y}$$

The total amount of parameters P is thus:

$$P = D_x \cdot H + H + H \cdot D_y + D_y$$

$$\boxed{P = H \cdot (1 + D_x) + D_y \cdot (1 + H)}$$

Word2Vec

(3a) **Q:** Assume you are given a "predicted" word vector v_c corresponding to the center word c for Skip-Gram, and word prediction is made with the softmax function found in word2vec models

$$\hat{y}_o = p(u_o \mid v_c) = \frac{\exp(u_o^T v_c)}{\sum_w^V \exp(u_w^T v_c)}$$

where u_w $(w = 1, \dots, V)$ are the "output" word vectors for all words in the vocabulary. Assuming Cross Entropy cost is applied to the prediction and word o is the predicted (the o-th element of the one-hot vector is one), derive the gradients with respect to v_c.

A: We rewrite the functions $CE(y, \hat{y})$ and $\hat{y}_o$ as follows:

$$CE(y, \hat{y})_o = - \sum y \cdot \hat{y}_o \qquad \hat{y}_o = \log\left(\frac{\exp(u_o^T v_c)}{\sum_w^V \exp(u_w^T v_c)}\right) = u_o^T v_c - \log \sum_w^V \exp(u_w^T v_c)$$

Then:
$$\frac{\partial CE}{\partial v_c} = - \sum y \frac{\partial \hat{y}_o}{\partial v_c}$$

$$\frac{\partial \hat{y}_o}{\partial v_c} = u_o - \frac{1}{\sum_w^V \exp(u_w^T v_c)} \cdot \sum_x^V \exp(u_x^T v_c) \cdot u_x$$

$$\frac{\partial \hat{y}_o}{\partial v_c} = u_o - \sum_x^V \frac{\exp(u_x^T v_c)}{\sum_w^V \exp(u_w^T v_c)} \cdot u_x$$

$$\frac{\partial \hat{y}_o}{\partial v_c} = u_o - \sum_x^V \hat{y}_x \cdot u_x$$

Inserting this into $\frac{\partial CE}{\partial v_c}$ we get our result

This is always 0 except in our case where o is the correct expected word

$$\frac{\partial CE}{\partial v_c} = - \sum y \cdot u_o - \sum_x^V \hat{y}_x u_x$$

$$\boxed{\Leftrightarrow \quad \frac{\partial CE}{\partial v_c} = - u_o + \sum_x^V \hat{y}_x u_x}$$

Word2Vec

(3b) Q: As in the previous part, derive gradients for the "output" word vectors U_k's (including u_o)

A: We again rewrite the functions $CE(y,\hat{y})$ and $\hat{y}$ as follows

$$CE(y,\hat{y}) = -\sum_i y_i \hat{y}_i \qquad \hat{y}_i = \log \frac{\exp(U_k^T V_c)}{\sum_w^V \exp(U_w^T V_c)} = U_k^T V_c - \log \sum \exp(U_w^T V_c)$$

The derivative wrt U_k is given by

$$\frac{\partial CE}{\partial U_k} = -\sum_i y_i \frac{\partial \hat{y}_i}{\partial U_k}$$

To calculate the derivative of $\hat{y}$ we have to differentiate between the two cases where $k = i$ and $k \neq i$

For $k = i$:

$$\frac{\partial y_i}{\partial U_k} = V_c - \frac{1}{\sum_w^V \exp(U_w^T V_c)} \cdot \exp(U_k^T V_c) \cdot V_c$$

$$= V_c - \frac{\exp(U_k^T V_c)}{\sum_w^V \exp(U_w V_c)} \cdot V_c$$

$$= V_c \cdot (1 - \hat{y}_k)$$

For $k \neq i$

$$\frac{\partial y_i}{\partial V_k} = -\frac{1}{\sum_w^V \exp(U_w^T V_c)} \cdot \exp(U_k^T V_c) \cdot V_c$$

$$= -V_c \hat{Y}_k$$

Cont'd →

Inserting these derivatives into $\frac{\partial CE}{\partial u_k}$ we get the following:

$$\frac{\partial CE}{\partial u_k} = -y_k \cdot v_c(1-\hat{y}_k) + \sum_{i \neq k} y_i \hat{y}_k v_c$$

$$= -y_k v_c + y_k v_c \hat{y}_k + \sum_{i \neq k} y_i \hat{y}_k v_c$$

$$= -y_k v_c + \sum_i y_i \hat{y}_k v_c$$

$$= -y_k v_c + \hat{y}_k v_c \cdot \underbrace{\sum_i y_i}_{=1}$$

$$\boxed{\frac{\partial CE}{\partial u_k} = v_c(\hat{y} - y)^T}$$