word2vec

Kristian Wichmann

June 3, 2017

This document is based on the first assignment of the Stanford CS224d course: Deep Learning for NLP.

1 A simple neural network

Consider a feed forward neural network with one hidden layer, as shown in figure 1. The input layer consists of a row[1] vector x of dimension D_x, i.e. $x \in \mathbb{R}^{1 \times D_x}$. The hidden layer has h neurons with a sigmoid activation function:

$$h = \sigma(xW^{(1)} + b^{(1)}) = \sigma(z^{(1)}), \quad z^{(1)} = xW^{(1)} + b^{(1)} \tag{1.1}$$

So $h \in \mathbb{R}^{1 \times H}, W^{(1)} \in \mathbb{R}^{D_x \times H}, b^{(1)} \in \mathbb{R}^{1 \times H}$. The output layer has D_y softmax neurons:

$$\hat{y} = s(hW^{(2)} + b^{(2)}) = s(z^{(2)}), \quad z^{(2)} = hW^{(2)} + b^{(2)} \tag{1.2}$$

Similarly $\hat{y} \in \mathbb{R}^{1 \times D_y}, W^{(2)} \in \mathbb{R}^{H \times D_y}, b^{(2)} \in \mathbb{R}^{1 \times D_y}$.

1.1 Capacity

The capacity - the number of parameters in the model - of this network is as follow:

- Weights in $W^{(1)}$: $D_x H$
- Bias terms in $b^{(1)}$: H
- Weights in $W^{(2)}$: HD_y
- Bias terms in $b^{(2)}$: D_y

Adding these up we get a total of $(D_x + 1)H + (H + 1)D_y$ parameters.

[1] Note that this is different from the column vector convention usually used.

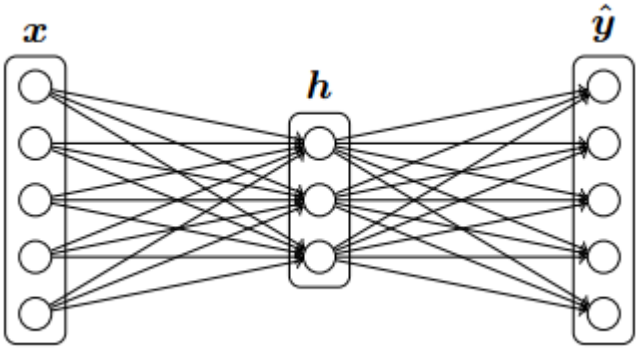

Figure 1: The neural network.

1.2 Error function

Assuming a labelled dataset x with the correct label c encoded as a one-hot vector $t \in \mathbb{R}^{1\times D_y}$, so $t_i = \delta_{ic}$. We will use the cross-entropy error function:

$$J(x) = -\sum_{i=1} t_i \log \hat{y}_i \tag{1.3}$$

Since t is one-hot encoded, only the correct label c will contribute to the sum, so:

$$J(x) = -\log \hat{y}_c \tag{1.4}$$

This does not mean that the other components of $\hat{y}$ will not matter, since the softmax indirectly depends on all components.

1.3 Derivatives with respect to input

We now wish to compute the derivative of $J(x)$ with respect to x. In shorthand, the chain rule gives us:

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial h} \frac{\partial h}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial x} \tag{1.5}$$

Writing out indices explicitly:

$$\frac{\partial J}{\partial x_m} = \sum_{i=1}^{D_y} \sum_{j=1}^{D_y} \sum_{k=1}^{H} \sum_{l=1}^{H} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_j^{(2)}} \frac{\partial z_j^{(2)}}{\partial h_k} \frac{\partial h_k}{\partial z_l^{(1)}} \frac{\partial z_l^{(1)}}{\partial x_m} \tag{1.6}$$

Let's compute these partial derivatives one by one:

$$\frac{\partial J}{\partial \hat{y}_i} = -\frac{\partial}{\partial \hat{y}_i} \log \hat{y}_c = -\frac{\delta_{ic}}{\hat{y}_c} \tag{1.7}$$

The second is a standard result for the softmax function:

$$\frac{\partial \hat{y}_i}{\partial z_j^{(2)}} = s_i(z^{(2)})(\delta_{ij} - s_j(z^{(2)}) = \hat{y}_i(\delta_{ij} - \hat{y}_j) \tag{1.8}$$

Let's pause for a moment and combine the two:

$$\frac{\partial J}{\partial z_j^{(2)}} = \sum_{i=1}^{D_y} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_j^{(2)}} = -\sum_{i=1}^{D_y} \frac{\delta_{ic}}{\hat{y}_c} \hat{y}_i(\delta_{ij} - \hat{y}_j) \tag{1.9}$$

The Kronecker delta removes the sum and we get:

$$-\frac{1}{\hat{y}_c} \hat{y}_c(\delta_{jc} - \hat{y}_j) = -(\delta_{jc} - \hat{y}_j) = \hat{y}_j - \delta_{jc} \tag{1.10}$$

But $\delta_{jc} = t_j$, and we get that the derivative is simply equal to the *error* in the output layer:

$$\frac{\partial J}{\partial z_j^{(2)}} = \delta_j^{(2)} \tag{1.11}$$

Here, $\delta_j^{(2)} = \hat{y}_j - \delta_{jc}$. Going back to the original derivation, the third derivative is:

$$\frac{\partial z_j^{(2)}}{\partial h_k} = W_{kj}^{(2)} \tag{1.12}$$

The fourth uses a standard result for the sigmoid:

$$\frac{\partial h_k}{\partial z_l^{(1)}} = \delta_{kl}\sigma(z_l^{(1)})(1 - \sigma(z_l^{(1)})) = \delta_{kl} h_l(1 - h_l) \tag{1.13}$$

And finally:

$$\frac{\partial z_l^{(1)}}{\partial x_m} = W_{ml}^{(1)} \tag{1.14}$$

Inserting, letting the delta functions cancel, and renaming indices, this becomes:

$$\frac{\partial J}{\partial x_i} = \sum_{j=1}^{D_y} \sum_{k=1}^{H} \delta_j^{(2)} W_{kj}^{(2)} h_k(1 - h_k) W_{ik}^{(1)} \tag{1.15}$$

The "wrong" order of the W indices can be reverted by using the transpose:

$$\frac{\partial J}{\partial x_i} = \sum_{j=1}^{D_y} \sum_{k=1}^{H} \delta_j^{(2)} \left(W^{(2)}\right)_{jk}^t h_k(1 - h_k) \left(W^{(1)}\right)_{ki}^t \tag{1.16}$$

We may rephrase this in terms of the backpropagated errors in the hidden layer:

$$\delta_k^{(1)} = \sum_{j=1}^{D_y} \delta_j^{(2)} \left(W^{(2)}\right)_{jk}^t h_k(1-h_k) = \left[\delta^{(2)} \left(W^{(2)}\right)^t\right]_k h_k(1-h_k) \tag{1.17}$$

Now equation 1.16 can be rewritten as:

$$\frac{\partial J}{\partial x_i} = \frac{\partial J}{\partial x_i} = \sum_{j=1}^{D_y} \sum_{k=1}^{H} \delta_j^{(2)} \left(W^{(2)}\right)_{jk}^t h_k(1-h_k) \left(W^{(1)}\right)_{ki}^t = \sum_{k=1}^{H} \delta_k^{(1)} \left(W^{(1)}\right)_{ki}^t \tag{1.18}$$

1.4 Derivatives with respect to weights and biases

Similarly, we may calculate derivatives corresponding to variations in the neural network parameters. For weights in $W^{(2)}$:

$$\frac{\partial J}{\partial W_{ij}^{(2)}} = \sum_{k=1}^{D_y} \frac{\partial J}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial W_{ij}^{(2)}} = \sum_{k=1}^{D_y} \delta_k^{(2)} \frac{\partial}{\partial W_{ij}^{(2)}} \left(\sum_{l=1}^{H} h_l W_{lk}^{(2)} + b_k^{(2)} \right) \tag{1.19}$$

Switching the order of derivative and summation, the innermost derivative is $h_l \delta_{il} \delta_{jk}$. The sums then cancel and we end up with:

$$\frac{\partial J}{\partial W_{ij}^{(2)}} = h_i \delta_j^{(2)} \tag{1.20}$$

For the biases between hidden and output layer:

$$\frac{\partial J}{\partial b_i^{(2)}} = \sum_{j=1}^{D_y} \frac{\partial J}{\partial z_j^{(2)}} \frac{\partial z_j^{(2)}}{\partial b_i^{(2)}} = \sum_{j=1}^{D_y} \delta_j^{(2)} \delta_{ij} = \delta_i^{(2)} \tag{1.21}$$

We can now use these in calculating the derivatives with respect to the weights between input and hidden layer:

$$\frac{\partial J}{\partial W_{ij}^{(1)}} = \sum_{k=1}^{H} \frac{\partial J}{\partial z_k^{(1)}} \frac{\partial z_k^{(1)}}{\partial W_{ij}^{(1)}} \tag{1.22}$$

The first derivative is simply the error term for the hidden layer as defined above. I.e.:

$$\frac{\partial J}{\partial W_{ij}^{(1)}} = \sum_{k=1}^{H} \delta_k^{(1)} \frac{\partial z_k^{(1)}}{\partial W_{ij}^{(1)}} \tag{1.23}$$

The remaining derivative is:

$$\frac{\partial}{\partial W_{ij}^{(1)}}\left(\sum_{l=1}^{D_x} x_l W_{lk}^{(1)} + b_k^{(1)}\right) = \sum_{l=1}^{D_x} \frac{\partial}{\partial W_{ij}^{(1)}} x_l W_{lk}^{(1)} = x_i \delta_{jk} \tag{1.24}$$

Here, one of the Kronecker deltas resulting from the derivative have been absorbed into the sum. The other delta is removed by the k summation, so:

$$\frac{\partial J}{\partial W_{ij}^{(1)}} = x_i \delta_j^{(1)} \tag{1.25}$$

Finally, the derivative with respect to the bias between the input and hidden layers is almost the same calculation. Here the last differentiation with respect to $b^{(1)}$ simply turns into a Kronecker delta, so that:

$$\frac{\partial J}{\partial b_i^{(1)}} = \delta_i^{(1)} \tag{1.26}$$

1.4.1 Vectorization

The above is for one input point x. But often we will have a set of n data points. These may be collected in an $n \times D_x$ matrix X:

$$X = \begin{pmatrix} - & x_1 & - \\ \vdots & \vdots & \vdots \\ - & x_n & - \end{pmatrix} \in \mathbb{R}^{n \times D_x} \tag{1.27}$$

Multiplying X by $W^{(1)}$ we get:

$$XW^{(1)} = \begin{pmatrix} - & x_1 W^{(1)} & - \\ \vdots & \vdots & \vdots \\ - & x_n W^{(1)} & - \end{pmatrix} \in \mathbb{R}^{n \times H} \tag{1.28}$$

We now need to add $b^{(1)}$ to all the row vectors. This can be done by adding:

$$\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} b^{(1)} = J_n b^{(1)} \tag{1.29}$$

Here, $J_n \in \mathbb{R}^{n \times 1}$ denotes the n-dimensional vector of all ones. So:

$$Z_1 = XW^{(1)} + J_n b^{(1)} \in \mathbb{R}^{n \times H} \tag{1.30}$$

Now apply the sigmoid to all entries:

$$H = \sigma(Z_1) = \begin{pmatrix} - & h_1 & - \\ \vdots & \vdots & \vdots \\ - & h_n & - \end{pmatrix} \in \mathbb{R}^{n \times H} \tag{1.31}$$

Similarly, we now multiply by $W^{(2)}$ and add $b^{(2)}$ to all rows:

$$Z_2 = HW^{(2)} + J_n b^{(2)} \in \mathbb{R}^{n \times D_y} \tag{1.32}$$

Finally apply the softmax rowwise:

$$\hat{Y} = s(Z_2) \in \mathbb{R}^{n \times D_y} \tag{1.33}$$

Similarly, the labels can be one-hot encoded in a $n \times D_y$ matrix T:

$$T = \begin{pmatrix} - & t_1 & - \\ \vdots & \vdots & \vdots \\ - & t_n & - \end{pmatrix} \in \mathbb{R}^{n \times D_y} \tag{1.34}$$

The output layer errors are then:

$$\Delta_2 = \hat{Y} - T \in \mathbb{R}^{n \times D_y} \tag{1.35}$$

The derivatives for the $b^{(2)}$ bias for data point i are then the i'th row of Δ_2. Since we average over the data points, we get the derivative by averaging Δ_2 columnwise:

$$\frac{\partial J}{\partial b^{(2)}} = \frac{1}{n} J_n^t \Delta_2 \in \mathbb{R}^{1 \times D_y} \tag{1.36}$$

For the derivatives of the $W^{(2)}$ weights calculate $H^t \Delta_2$:

$$H^t E_2 = \begin{pmatrix} | & \cdots & | \\ h_1 & \cdots & h_n \\ | & \cdots & | \end{pmatrix} \begin{pmatrix} - & \delta_1^{(2)} & - \\ \vdots & \vdots & \vdots \\ - & \delta_n^{(2)} & - \end{pmatrix} \in \mathbb{R}^{H \times D_y} \tag{1.37}$$

We now see, that each entry is the sum of the derivatives of the relevant entry of $W^{(2)}$ over all the n data points. Again, we normalize:

$$\frac{\partial J}{\partial W^{(2)}} = \frac{1}{n} H^t \Delta_2 \in \mathbb{R}^{H \times D_y} \tag{1.38}$$

Now, we wish to calculate the errors in the hidden layer. Equation 1.17 shows that this can be written as a hadamard product:

$$\Delta_1 = \left[\Delta_2 \left(W^{(2)} \right)^t \right] \odot H \odot (J_{nH} - H) \in \mathbb{R}^{n \times H} \tag{1.39}$$

Now, the calculation of the remaining derivatives is done analogously:

$$\frac{\partial J}{\partial b^{(1)}} = \frac{1}{n} J_n^t \Delta_1 \in \mathbb{R}^{1 \times H} \tag{1.40}$$

And:

$$\frac{\partial J}{\partial W^{(1)}} = \frac{1}{n} X^t \Delta_1 \in \mathbb{R}^{D_x \times H} \tag{1.41}$$

2 The word2vec algorithm

2.1 The situation

Imagine a large corpus of consecutive tokens (words) of length T. Each token comes from a vocabulary of size W. We wish to encode the tokens as vectors. To do this, we look at the words surrounding a given word in the corpus probabilistically. If c is the center word, we can consider the probability of a word j places away being o. We should really denote this probability $p_j(o|c)$, but for now we will simply call it $p(o|c)$.

2.2 Softmax and word vectors

If we have each word represented by vectors, an appropriate model for the probability could be a softmax over the entire vocabulary size:

$$p(o|c) = \frac{\exp(u_o^t v_c)}{\sum_{w=1}^{W} \exp(u_w^t v_c)} \tag{2.1}$$

Note that each word w has two vectors: u_w and v_w. u_w is for w as an 'outside' word, and v_w is for w as a 'center' word.

2.3 Error function and derivatives

An appropriate error function for the softmax is the cross-entropy error. The situation is entirely analogous to the propagation from the hidden layer to the output layer in the neural network described in last section. I.e. it is described by equations 1.9 and 1.11 with the following replacements:

- $j \mapsto o$
- $z_j^{(2)} \mapsto z_o = u_o^t v_c$
- $i \mapsto w$

- $\hat{y}_i \mapsto p(o|c)$

So, the derivative with respect to v_c is:

$$\frac{\partial J}{\partial v_c} = \sum_{o=1}^{W} \frac{\partial J}{\partial z_0} \frac{\partial z_0}{\partial v_c} = \sum_{o=1}^{W} \delta_o^{(2)} u_o = \sum_{o=1}^{W} u_o \delta_o^{(2)} \tag{2.2}$$

Remember, that the error is a scalar, while u_o is a vector:

$$u_o = \begin{pmatrix} u_{o1} \\ \vdots \\ u_{oW} \end{pmatrix} \tag{2.3}$$

Let's collect all the u vectors into a matrix:

$$U = \begin{pmatrix} | & \cdots & | \\ u_1 & \cdots & u_W \\ | & \cdots & | \end{pmatrix} \tag{2.4}$$

Now equation 2.2 may be re-written:

$$\frac{\partial J}{\partial v_c} = U^t \delta^{(2)} \tag{2.5}$$

Similarly, we will be interested in the derivative with respect to u_w:

$$\frac{\partial J}{\partial u_w} = \sum_{o=1}^{W} \frac{\partial J}{\partial z_o} \frac{\partial z_o}{\partial u_w} = \sum_{o=1}^{W} \delta_o^{(2)} \delta_{ow} v_c = \delta_w^{(2)} v_c \tag{2.6}$$

2.4 Derivatives of $\log p$

We will also need derivatives of the logarithm of p:

$$\log p(o|c) = \log \left(\frac{\exp(u_o^t v_c)}{\sum_{w=1}^{W} \exp(u_w^t v_c)} \right) = \log(\exp(u_o^t v_c)) - \log \left[\sum_{w=1}^{W} \exp(u_w^t v_c) \right] \tag{2.7}$$

This simplifies to:

$$\log p(o|c) = u_o^t v_c - \log \left[\sum_{w=1}^{W} \exp(u_w^t v_c) \right] \tag{2.8}$$

2.4.1 With respect to v_c

First, let's differentiate with respect to v_c:

$$\frac{\partial \log p(o|c)}{\partial v_c} = u_0 - \frac{\partial}{\partial v_c} \log \left[\sum_{w=1}^{W} \exp(u_w^t v_c) \right] \tag{2.9}$$

Let's consider the last term by itself. It can be calculated using the chain rule. It is basically of the form:

$$\frac{\partial}{\partial x} \log(f(x)) = \frac{1}{f(x)} \frac{\partial f}{\partial dx} \tag{2.10}$$

Here, this means:

$$\frac{\partial}{\partial v_c} \log \left[\sum_{w=1}^{W} \exp(u_w^t v_c) \right] = \frac{\frac{\partial}{\partial v_c} \sum_{x=1}^{W} \exp(u_x^t v_c)}{\sum_{w=1}^{W} \exp(u_w^t v_c)} \tag{2.11}$$

Consider the numerator:

$$\frac{\partial}{\partial v_c} \sum_{x=1}^{W} \exp(u_x^t v_c) = \sum_{x=1}^{W} \frac{\partial}{\partial v_c} \exp(u_x^t v_c) = \sum_{x=1}^{W} \exp(u_x^t v_c) \frac{\partial}{\partial v_c} u_x^t v_c \tag{2.12}$$

So the numerator is $\sum_{x=1}^{W} \exp(u_x^t v_c) u_x$. Now, we may write the entire derivative as:

$$\sum_{x=1}^{W} \frac{\exp(u_x^t v_c)}{\sum_{w=1}^{W} \exp(u_w^t v_c)} u_x = \sum_{x=1}^{W} p(x|c) u_x \tag{2.13}$$

Summing it all up (and changing the index back to w):

$$\frac{\partial}{\partial v_c} \log p(o|c) = u_o - \sum_{w=1}^{W} p(w|c) u_w \tag{2.14}$$

2.4.2 With respect to u_w

Similarly, the derivative with respect to u_w is:

$$\frac{\partial \log p(o|c)}{\partial u_w} = \frac{\partial}{\partial u_w} \left(u_o^t v_c - \log \left[\sum_{x=1}^{W} \exp(u_x^t v_c) \right] \right) \tag{2.15}$$

The derivative of the first term is non-zero only when $w = o$, so the derivative is $\delta_{ow} v_c$. The derivative of the second term can be used by using equation 2.10 and the chain rule again:

$$\frac{\partial}{\partial u_w} \log \left[\sum_{x=1}^{W} \exp(u_x^t v_c) \right] = \frac{\frac{\partial}{\partial u_w} \sum_{x=1}^{W} \exp(u_x^t v_c)}{\sum_{y=1}^{W} \exp(u_y^t v_c)} \tag{2.16}$$

The numerator is:

$$\sum_{x=1}^{W} \exp(u_x^t v_c)\delta_{xw} v_c = \exp(u_w^t v_c) v_c \tag{2.17}$$

All in all, this means that the derivative is:

$$\frac{\partial}{\partial u_w} \log p(o|c) = [\delta_{ow} - p(w|c)]\, v_c \tag{2.18}$$