cs224n assignment 1 solutions

Andreea Musat
andreea.a.musat@gmail.com

February 2018

1 Softmax

(a) (5 points) Prove that softmax is invariant to constant offsets in the input, that is, for any input vector x and any constant α, $softmax(x) = softmax(x + \alpha)$, where x + α means adding the constant α to every dimension of x. Remember that $softmax(x)_i = \frac{e^{x_i}}{\sum_{c=1}^{C} e^{x_c}}$

(1) Note: In practice, we make use of this property and choose $\alpha = -max_i x_i$ when computing softmax probabilities for numerical stability (i.e., subtracting its maximum element from all elements of x).

Solution: For every $1 \leq i \leq C$, we have:

$$softmax(x+\alpha)_i = \frac{e^{x_i+\alpha}}{\sum_{c=1}^{C} e^{x_c+\alpha}} = \frac{e^{x_i}e^{\alpha}}{\sum_{c=1}^{C} e^{x_c}e^{\alpha}} = \frac{e^{x_i}e^{\alpha}}{e^{\alpha}\sum_{c=1}^{C} e^{x_c}} = \frac{e^{x_i}}{\sum_{c=1}^{C} e^{x_c}} = softmax(x)_i, q.e.d.$$

2 Neural Network basics

(a) (3 points) Derive the gradients of the sigmoid function and show that it can be rewritten as a function of the function value (i.e., in some expression where only $\sigma(x)$, but not x, is present). Assume that the input x is a scalar for this question. Recall, the sigmoid function is $sigmoid(x) = \frac{1}{1+e^{-x}}$

Solution:

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

(b) (3 points) Derive the gradient with regard 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:

$$CE(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{C} y_i log(\hat{y_i}) \tag{1}$$

where y is the one-hot label vector, and $\hat{y}$ is the predicted probability vector for all classes. (Hint: you might want to consider the fact many elements of y are zeros, and assume that only the k-th dimension of y is one.)

Solution: Taking the derivative of CE with respect to a single parameter, θ_i, we get:

$$\frac{\partial CE(y,\hat{y})}{\partial \theta_i} = \frac{\partial}{\partial \theta_i}\Big[-\sum_{K=1}^{C} y_k log(\hat{y_k})\Big] = \frac{\partial}{\partial \theta_i}\Big[-\sum_{k=1}^{C} y_k log(softmax(\theta)_k)\Big] = -\sum_{k=1}^{C} y_k \frac{\partial}{\partial \theta_i}\Big[log(softmax(\theta)_k)\Big] =$$
$$-\sum_{k=1}^{C} y_k \frac{\partial}{\partial \theta_i}\Big[log(\frac{e^{\theta_k}}{\sum_{j=1}^{C} e^{\theta_j}})\Big] = -\sum_{k=1}^{C} y_k \frac{\partial}{\partial \theta_i}\Big[log(e^{\theta_k}) - log(\sum_{j=1}^{C} e^{\theta_j})\Big] = -\sum_{k=1}^{C} y_k \Big[\frac{\partial}{\partial \theta_i} log(e^{\theta_k}) - \frac{\partial}{\partial \theta_i} log(\sum_{j=1}^{C} e^{\theta_j})\Big] =$$
$$-\sum_{k=1}^{C} y_k \Big[\frac{\partial}{\partial \theta_i}\theta_k - \frac{1}{\sum_{j=1}^{C} e^{\theta_j}} \frac{\partial}{\partial \theta_i} \sum_{j=1}^{C} e^{\theta_j}\Big] = -\sum_{k=1}^{C} y_k \Big[\frac{\partial}{\partial \theta_i}\theta_k - \frac{e^{\theta_i}}{\sum_{j=1}^{C} e^{\theta_j}}\Big]$$

If i is the index of the correct class, then $y_k = 1$ and $\frac{\partial}{\partial \theta_i}\theta_k = 1$ only when i = k, so the derivative is:

$$\frac{\partial CE(y,\hat{y})}{\partial \theta_i} = -\Big[\frac{\partial}{\partial \theta_i}\theta_i - \frac{e^{\theta_i}}{\sum_{j=1}^{C} e^{\theta_j}}\Big] = \frac{e^{\theta_i}}{\sum_{j=1}^{C} e^{\theta_j}} - 1 = softmax(\theta)_i - 1 = \hat{y_i} - 1$$

If i is not the index of the correct class, then:

$$\frac{\partial CE(y,\hat{y})}{\partial \theta_i} = \frac{e^{\theta_i}}{\sum_{j=1}^{C} e^{\theta_j}} = softmax(\theta)_i = \hat{y_i} \tag{2}$$

We can write the gradient in its final form:

$$\frac{\partial CE(y,\hat{y})}{\partial \theta_i} = \begin{cases} \hat{y_i} - 1 & \text{if i is the correct class} \\ \hat{y_i} & \text{otherwise} \end{cases} \tag{3}$$

(c) (6 points) Derive the gradients with respect to the inputs x to an 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 cost is used. (Feel free to use $\sigma'(x)$ as the shorthand for sigmoid gradient, and feel free to define any variables whenever you see fit.)

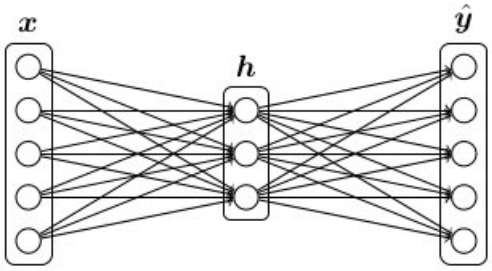

Figure 1: Neural Network

Recall that the forward propagation is as follows: $h = sigmoid(xW_1 + b_1)$ and $\hat{y} = softmax(hW_2 + b_2)$. Note that here we're assuming that the input vector (thus the hidden variables and output probabilities) is a row vector to be consistent with the programming assignment. When we apply the sigmoid function to a vector, we are applying it to each of the elements of that vector. W_i and b_i, i = (1, 2) are the weights and biases, respectively, of the two layers.

Solution:

Forward pass: Assuming that the input layer has D_x neurons, the hidden layer has H neurons and the output layer has D_y neurons and all the vectors are row-vectors, we can write:

$$x = z^{(0)} = a^{(0)}, \text{layer 0 (input layer)},$$
$$z^{(1)} = a^{(0)}W^{(1)} + b^{(1)}, \text{the weighted inputs of layer 1},$$
$$a^{(1)} = \sigma(z^{(1)}), \text{the activations of layer 1},$$
$$z^{(2)} = a^{(1)}W^{(2)} + b(2), \text{the weighted inputs of layer 2},$$
$$a^{(2)} = softmax(z^{(2)}), \text{the activations of layer 2},$$

where we have used the notation $W^{(i)}$ and $b^{(i)}$ for the weights and biases, respectively, in order to be consistent with the notation for the activations and weighted inputs of the neurons.

In order to check the correctness of the gradients later, let us write the dimensionality of each matrix/vector:

$$x, z^{(0)}, a^{(0)} \in \mathbb{R}^{1\times D_x},$$
$$z^{(1)}, a^{(1)}, b^{(1)} \in \mathbb{R}^{1\times H}$$
$$z^{(2)}, a^{(2)}, b^{(2)} \in \mathbb{R}^{1\times D_y}$$
$$W^{(1)} \in \mathbb{R}^{D_x\times H}$$
$$W^{(2)} \in \mathbb{R}^{H\times D_y}$$

Backward pass We first compute the error in the last layer:

$$\delta^{(2)} = \frac{\partial J}{\partial z^{(2)}} = \frac{\partial CE(y,\hat{y})}{\partial z^{(2)}} = \hat{y} - y, \text{ where } \delta^{(2)} \in \mathbb{R}^{1\times D_y}$$

(This is the gradient of the cross entropy loss function with respect to the inputs of the softmax function, see the previous exercise for details.)

Now we can compute the gradient with respect to $a_i^{(1)}$ using the chain rule (we'll get the component form first and then write it in matrix form):

$$\frac{\partial J}{\partial a_i^{(1)}} = \sum_k \frac{\partial J}{\partial z_k^{(2)}} \frac{\partial z_k^{(2)}}{\partial a_i^{(1)}} = \sum_k \delta_k^{(2)} \frac{\partial z_k^{(2)}}{\partial a_i^{(1)}}$$

, where the sum is over every neuron k in layer 2. Now, in order to compute $\frac{\partial z_k^{(2)}}{\partial a_i^{(1)}}$, we write the component form of $z_k^{(2)}$:

$$z_k^{(2)} = \sum_i a_i^{(1)} W_{ik}^{(2)} + b_k^{(2)}$$

We can observe that: $\frac{\partial z_k^{(2)}}{\partial a_i^{(1)}} = W_{ik}^{(2)}$, so the gradient of J with respect to $a_i^{(1)}$ is:

$$\frac{\partial J}{\partial a_i^{(1)}} = \sum_k \delta_k^{(2)} W_{ik}^{(2)}$$

We can finally write the gradient in matrix notation:

$$\delta_{a^{(1)}} = \frac{\partial J}{\partial a^{(1)}} = \delta^{(2)} (W^{(2)})^T$$

As $\delta^{(2)} \in \mathbb{R}^{1\times D_y}$ and $(W^{(2)})^T \in \mathbb{R}^{D_y\times H}$, we obtain that $\frac{\partial J}{\partial a^{(1)}} \in \mathbb{R}^{1\times H}$, which is correct, as the gradient should have the same shape as $a^{(1)}$

Now we compute the gradient of J with respect to $z^{(1)}$:

$$\frac{\partial J}{\partial z_i^{(1)}} = \sum_k \frac{\partial J}{\partial a_k^{(1)}} \frac{\partial a_k^{(1)}}{\partial z_i^{(1)}}$$

As $\frac{\partial a_k^{(1)}}{\partial z_i^{(1)}} = 0$ when $i \neq k$ (the activation of a neuron only depends of its weighted input), we obtain:

$$\frac{\partial J}{\partial z_i^{(1)}} = \frac{\partial J}{\partial a_i^{(1)}} \frac{\partial a_i^{(1)}}{\partial z_i^{(1)}} = \delta_{a_i^{(1)}} \sigma'(z_i^{(1)})$$

To write this in matrix form, we use the Hadamard product:

$$\delta^{(1)} = \frac{\partial J}{\partial z^{(1)}} = \delta_{a^{(1)}} \odot \sigma'(z^{(1)}) \in \mathbb{R}^{1 \times H}$$

Finally, we compute the gradient of J with respect to the input x:

$$\frac{\partial J}{\partial x_i} = \sum_k \frac{\partial J}{\partial z_k^{(1)}} \frac{\partial z_k^{(1)}}{\partial x_i} = \sum_k \delta_k^{(1)} \frac{\partial z_k^{(1)}}{\partial x_i}$$

The computation is similar to that for $\frac{\partial J}{\partial x_i}$. We finally obtain:

$$\frac{\partial J}{\partial x} = \delta^{(1)} (W_1)^T$$

As $\delta^{(1)} \in \mathbb{R}^{1 \times H}$ and $(W_1)^T \in \mathbb{R}^{H \times D_x}$, it follows that $\frac{\partial J}{\partial x} \in \mathbb{R}^{1 \times D_x}$, which is exactly the shape of x, so the gradient has the correct shape.

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

Solution:

The total number of parameters is the sum of the sizes of all the W matrices and b vectors:

$$|dom(W_1)| + |dom(b_1)| + |dom(W_2)| + |dom(b_2)| = D_x H + H + H D_y + D_y = H(1 + D_x) + D_y(1 + H)$$

3 word2vec

a) Assume you are given a predicted word vector v_c corresponding to the center word c for skipgram, and word prediction is made with the softmax function found in word2vec models

$$\hat{y_o} = p(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}$$

where w denotes the w-th word and u_w (w = 1, . . . , W) are the "output" word vectors for all words in the vocabulary. Assume cross entropy cost is applied to this prediction and word o is the expected word (the o-th element of the one-hot label vector is one), derive the gradients with respect to v_c.

Hint: It will be helpful to use notation from question 2. For instance, letting $\hat{y}$ be the vector of softmax predictions for every word, y as the expected word vector, and the loss function

$$J_{softmax_{CE}}(o, v_c, U) = CE(y, \hat{y})$$

,
where $\boldsymbol{U} = [u_1, u_2, ..., u_W]$ *is the matrix of all the output vectors. Make sure you state the orientation of your vectors and matrices.*

Solution:

$$\frac{\partial J}{\partial v_c} = \frac{\partial J_{softmax_{CE}}(o, v_c, U)}{\partial v_c} = \frac{\partial}{\partial v_c}\Big[-\sum_{i=1}^{C} y_i log(\hat{y_i})\Big] = -\sum_{i=1}^{C} y_i \frac{\partial}{\partial v_c} log(\hat{y_i}) = -\sum_{i=1}^{C} y_i \frac{\partial}{\partial v_c} log(\frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}) =$$

$$-\sum_{i=1}^{C} y_i \frac{\partial}{\partial v_c}\Big[u_o^T v_c - log(\sum_{w=1}^{W} exp(u_w^T v_c))\Big] = -\sum_{i=1}^{C} y_i \Big[u_o - \frac{\partial}{\partial v_c} log(\sum_{w=1}^{W} exp(u_w^T v_c))\Big] =$$

$$-\sum_{i=1}^{C} y_i \Big[u_o - \frac{1}{\sum_{w=1}^{W} exp(u_w^T v_c)} \frac{\partial}{\partial v_c} \sum_{x=1}^{W} exp(u_x^T v_c)\Big] = -\sum_{i=1}^{C} y_i \Big[u_o - \frac{1}{\sum_{w=1}^{W} exp(u_w^T v_c)} \sum_{x=1}^{W} u_x exp(u_x^T v_c)\Big] =$$

$$-\sum_{i=1}^{C} y_i \Big[u_o - \sum_{x=1}^{W} \frac{u_x exp(u_x^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}\Big] = -\sum_{i=1}^{C} y_i \Big[u_o - \sum_{x=1}^{W} u_x \hat{y_x}\Big]$$

As $y_i = 1$ only if $i = o$ (is the expected word), we obtain:

$$\frac{\partial J}{\partial v_c} = -u_o + \sum_{x=1}^{W} \hat{y_x} u_x \tag{4}$$