Problem Set 2, Problem 2

Kireet

October 20, 2015

Abstract

This doc contains the derivations needed to complete the coding of the feed forward neural net problem.

Contents

1 Introduction

To run back prop, we need to calculate a bunch of derivatives with respect to the cost function. It's easiest to do the math element wise and then figure out the vectorization afterwards.

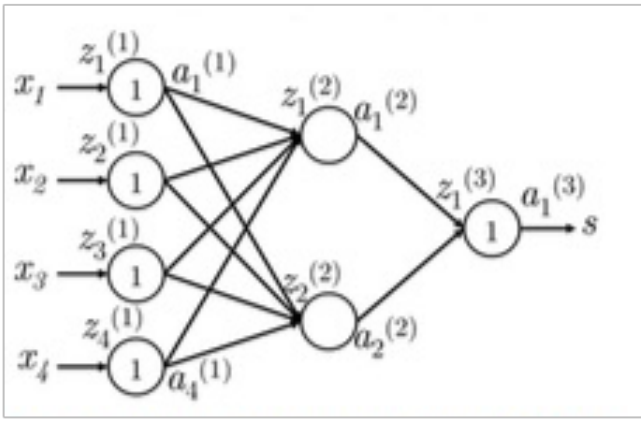

Figure 1: A typical neural network

The above figure shows a sample 3 layer neural network with typical notation. The leftmost layer is the "input" layer, it simply receives input values x_1 to x_4 and does not do any transformation. The middle or "hidden" layer combines the input signals and the final layer again combine the hidden layer output. The hidden and output layer nodes may define non linear transformations such as the sigmoid or tanh functions. Also not pictured are linear weighting factors on each of edges of the graph.

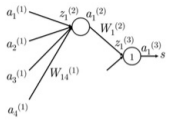

Figure 2: Zoomed in view of a node in the network

Values related to nodes are identified by their layer and position within the layer. The input to each node is identified by z and the output by a. Note if the node does no transformation, these values are identical. Using our standard notation, $n_j^{(l)}$ is node j in layer l, $z_j^{(l)}$ is the input to node to that node, and $a_j^{(l)}$ is the output of the same node. The input layer is a special case, x_j, $z_j^{(1)}$, and $a_j^{(1)}$ all refer to the same value. The aforementioned weights between each node are identified by 3 coordinates, the input layer, the output node, and the input node. E.g. $W_{ij}^{(l)}$ refers to the weight between $n_j^{(l)}$ and $n_i^{(l+1)}$. Take note that the subscript coordinates may appear "backward" in the sense that they flow right to left in the figures. This is for mathmatical convenience.

If the subscripts are not included, then the variable refers to the vectorized representation of all values of the layer. E.g. $z^{(2)}$, $a^{(2)}$ and $W^{(2)}$ refer to all of the inputs, outputs, and weights for layer 2, respectively.

2 Setup

The problem states we are given a 3 layer network, the same general architecture as in Figure 1. However, we have 150 inputs at the input layer, 100 hidden nodes, and 5 output nodes. The hidden layer uses the tanh function, and the output layer uses softmax. The problem set also uses different notation:

- The inputs are referred to as L or L_i instead of $\vec{x}$ or x_j. The assignment uses x to actually refer to a one hot vector identifying a particular word in the vocabulary. We'll instead use w for this purpose.
- The first weights matrix is referred to as W instead of $W^{(1)}$.
- The second weights matrix is referred to as U and is transposed. So U^T is our $W^{(2)}$.
- Each layer has a bias input, $b^{(2)}$ and $b^{(3)}$. This is not pictured in the figures above but is standard practice. The assignment set uses 1 and 2 as the indices into layer 2 and 3 (the assignment refers to this architecture as a 2 layer network).
- The final output is referred to by $\hat{y}$ instead of $a^{(3)}$. We'll use $\hat{y}$ at times for convenience.

We will use our standard notation instead of the problem set notation. Thus:

$$w \in \mathbb{R}^{|V|} \quad \vec{x}, z^{(1)}, a^{(1)} \in \mathbb{R}^{150} \quad W^{(1)} \in \mathbb{R}^{100\times 150}$$
$$z^{(2)}, a^{(2)}, b^{(2)} \in \mathbb{R}^{100} \quad W^{(2)} \in \mathbb{R}^{5\times 100}$$
$$z^{(3)}, a^{(3)}, b^{(3)} \in \mathbb{R}^{5}$$

The softmax function is given as:

$$\rho(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}$$

The tanh function is given as:

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

where $\sigma()$ is the sigmoid function:

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

The derivative of the tanh function can be written as a function of itself or of the sigmoid:

$$\tau'(x) = 1 - \tau(x)^2 = 4\sigma(2x)(1 - \sigma(2x))$$

Our cost function is the cross entropy loss function:

$$J(\theta) = -\sum_{k=1}^{5} y_k log(\hat{y_k})$$

where y is the one hot vector identifying the correct output class.

We are asked to compute The partial derivative of J with respect to $W^{(2)}$, $b^{(3)}$, $W^{(1)}$, $b^{(2)}$, and $\vec{x}$.

3 Cross Entropy Loss/Softmax Derivative

First we need to compute the derivative of the loss function with respect to the softmax function input, here $z^{(3)}$. It's convenient to combine these as the final result is quite simple:

$$J(\theta) = -\sum_{k=1}^{5} y_k log(\hat{y_k})$$

for a given correct class i:

$$J(\theta) = -log(\hat{y_i})$$

$$J(\theta) = -log(\frac{exp(z_i^{(3)})}{\sum_{j=1}^{5} exp(z_j^{(3)})})$$

$$J(\theta) = -z_i^{(3)} + log(\sum_{j=1}^{5} exp(z_j^{(3)}))$$

$$\frac{\partial J}{\partial z_t^{(3)}} = -\frac{\partial z_i^{(3)}}{\partial z_t^{(3)}} + \frac{1}{\sum_{j=1}^{5} exp(z_j^{(3)})} exp(z_t^{(3)})$$

$$= -\frac{\partial z_i^{(3)}}{\partial z_t^{(3)}} + \rho(z_t^{(3)})$$

We can see the first expression will be -1 when $i = t$, and zero otherwise. Combined with the facts that the output $\hat{y}_t$ is also $\rho(z_t^{(3)})$ and y is the one hot vector with a one for the correct class and zero otherwise, we can rewrite this derivative in vectorized form as a function of the feed forward output:

$$\delta^{(3)} \triangleq \frac{\partial J}{\partial z^{(3)}} = \hat{y} - y \tag{1}$$

We define this as the "error" at layer 3 and refer to it as $\delta^{(3)}$. In general, the error at a particular layer will be the partial derivative of the cost function with respect to the input value of that layer.

4 Layer 3 Bias

$$\frac{\partial J}{\partial b^{(3)}} = \frac{\partial J}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial b^{(3)}} \in \mathbb{R}^5$$

The first term we know from equation 1 to be $\delta^{(3)}$. We need to calculate the last partial and vectorize it to complete the derivation. Note the dimensions, since $b^{(3)} \in \mathbb{R}^5$, the cost function derivative with respect to $b^{(3)}$ should be as well. An important observation is that the j^{th} bias only affects the j^{th} output node. Thus we can simplify this to a simpler element wise derivative:

$$\begin{aligned}\frac{\partial z^{(3)}}{\partial b_i^{(3)}} = \frac{\partial z_i^{(3)}}{\partial b_i^{(3)}} &= \frac{\partial}{\partial b_i^{(3)}} \left(\sum_{j=1}^{100} W_{ij}^{(2)} a_j^{(2)} + b_i^{(3)} \right) \\ &= 1\end{aligned}$$

Thus $\frac{\partial z^{(3)}}{\partial b^{(3)}} = \vec{1}$ which leads to

$$\nabla_{b^{(3)}} = \delta^{(3)} \tag{2}$$

5 Layer 2 Weights

This derivation is similar to the layer 3 bias derivation, but just a little bit more complex:

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial J}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial W^{(2)}} \in \mathbb{R}^{5 \times 100}$$

The key thing to note in this derivation is $W_{ij}^{(2)}$ only affects $z_i^{(3)}$ and thus $a_i^{(3)}$. This is easiest understood by looking at the figures in the introduction and noting how values flow forward through the network. Therefore:

$$\begin{aligned}\frac{\partial z^{(3)}}{\partial W_{ij}^{(2)}} = \frac{\partial z_i^{(3)}}{\partial W_{ij}^{(2)}} &= \frac{\partial}{\partial W_{ij}^{(2)}} \left(\sum_{k=1}^{100} W_{ik}^{(2)} a_k^{(2)} + b_i^{(3)} \right) \\ &= a_j^{(2)}\end{aligned}$$

Combining things, we have:

$$\frac{\partial J}{\partial W_{ij}^{(2)}} = \delta_i^{(3)} \cdot a_j^{(2)}$$

In matrix form this looks like

$$\nabla_{W^{(2)}} = \begin{bmatrix} \delta_1^{(3)} a_1^{(2)} & \delta_1^{(3)} a_2^{(2)} & \delta_1^{(3)} a_3^{(2)} & \dots & \delta_1^{(3)} a_{100}^{(2)} \\ \delta_2^{(3)} a_1^{(2)} & \delta_2^{(3)} a_2^{(2)} & \delta_2^{(3)} a_3^{(2)} & \dots & \delta_2^{(3)} a_{100}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta_5^{(3)} a_1^{(2)} & \delta_5^{(3)} a_2^{(2)} & \delta_5^{(3)} a_3^{(2)} & \dots & \delta_5^{(3)} a_{100}^{(2)} \end{bmatrix}$$

$$\nabla_{W^{(2)}} = \delta^{(3)}(a^{(2)})^T \tag{3}$$

We can double check dimensions:

$$\nabla_{W^{(2)}} \in \mathbb{R}^{5\times 100}, \quad \delta^{(3)} \in \mathbb{R}^5, \quad a^{(2)} \in \mathbb{R}^{100}$$
$$\delta^{(3)}(a^{(2)})^T \Rightarrow [5\times 1] * [1\times 100] \in \mathbb{R}^{5\times 100}$$

6 Layer 1 Weights

The Layer 1 weights derivation is similar to the Layer 2 derivation, but with an additional layer involved. Similar to the previous derivation, we use the fact that $W_{ij}^{(1)}$ only affects $z_i^{(2)}$:

$$\begin{aligned}\frac{\partial J}{\partial W_{ij}^{(1)}} &= \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial W_{ij}^{(1)}} \\ &= \sum_k \frac{\partial J}{\partial z_k^{(2)}} \cdot \frac{\partial z_k^{(2)}}{\partial W_{ij}^{(1)}} \\ &= \frac{\partial J}{\partial z_i^{(2)}} \cdot \frac{\partial z_i^{(2)}}{\partial W_{ij}^{(1)}}\end{aligned}$$

First we derive the first term. Note that $z_i^{(2)}$ impacts all of layer 3, so we can't use any simplifying assumption here.

$$\begin{aligned}\frac{\partial J}{\partial z_i^{(2)}} &= \frac{\partial J}{\partial a_i^{(2)}} \cdot \tau'(z_i^{(2)}) \\ &= \tau'(z_i^{(2)}) \sum_k \frac{\partial J}{\partial z_k^{(3)}} \cdot \frac{\partial z_k^{(3)}}{\partial a_i^{(2)}} \\ &= \tau'(z_i^{(2)}) \sum_k \delta_k^{(3)} \cdot \frac{\partial z_k^{(3)}}{\partial a_i^{(2)}} \\ &= \tau'(z_i^{(2)}) \sum_k \delta_k^{(3)} \cdot \frac{\partial}{\partial a_i^{(2)}} \left(\sum_l W_{kl}^{(2)} a_k^{(2)} + b_k^{(3)} \right) \\ \delta_i^{(2)} \triangleq \frac{\partial J}{\partial z_i^{(2)}} &= \tau'(z_i^{(2)}) \sum_k \delta_k^{(3)} \cdot W_{ki}^{(2)}\end{aligned}$$

Note that while the form of $\delta_j^{(2)}$ differs from $\delta_i^{(3)}$, their meanings are similar. Namely, each represents the derivative of the cost function with respect to the input to the layer. Now the second term:

$$\frac{\partial z_i^{(2)}}{\partial W_{ij}^{(1)}} = \frac{\partial}{\partial W_{ij}^{(1)}} \left(\sum_k W_{ik}^{(1)} x_k + b_i^{(2)} \right)$$

$$= x_j$$

Combining these we have:

$$\frac{\partial J}{\partial W_{ij}^{(1)}} = \delta_i^{(2)} \cdot x_j$$

Notice this is the same form as the layer 2 derivations, the derivative is some back-propagated error at the output layer multiplied by the incoming value at the input layer! It's instructive to think about the components of which $\delta^{(2)}$ is composed: $\tau'(z^{(2)})$, $W^{(2)}$, and $\delta^{(3)}$ which is itself composed of $\rho'(z^{(3)})$ and the cross entropy loss function. So we can see that all the "downstream" operations play a part in the back-propagating the error "upstream".

To vectorize operations, we can first compute $\delta^{(2)}$:

$$\delta^{(2)} = \begin{bmatrix} \tau'(z_1^{(2)}) \cdot (\delta_1^{(3)} W_{11}^{(2)} + \delta_2^{(3)} W_{21}^{(2)} + \delta_3^{(3)} W_{31}^{(2)} + ...) \\ \tau'(z_2^{(2)}) \cdot (\delta_1^{(3)} W_{12}^{(2)} + \delta_2^{(3)} W_{22}^{(2)} + \delta_2^{(3)} W_{32}^{(2)} + ...) \\ \vdots \end{bmatrix}$$

$$\delta^{(2)} = \tau'(z^{(2)}) \circ \left((W^{(2)})^T \delta^{(3)} \right) \tag{4}$$

We can again verify dimensions:

$$\delta^{(2)} \in \mathbb{R}^{100}, (W^{(2)})^T \in \mathbb{R}^{100 \times 5}, \delta^{(3)} \in \mathbb{R}^5 \Rightarrow (W^{(2)})^T \delta^{(3)} \in \mathbb{R}^{100 \times 1}$$

$\circ$ is the element wise product of 2 vectors or matrices. This operator requires their dimensions to be the same and produces a vector of the same dimensions. Thus we can see that the dimensions are correct and $\delta^{(2)} \in \mathbb{R}^{100}$ – there should be an error signal value for each hidden layer node.

Using this vector, we can vectorize the $W^{(1)}$ calculation:

$$\nabla_{W^{(1)}} = \begin{bmatrix} \delta_1^{(2)} x_1 & \delta_1^{(2)} x_2 & \delta_1^{(2)} x_3 & \dots & \delta_1^{(2)} x_{150} \\ \delta_2^{(2)} x_1 & \delta_2^{(2)} x_2 & \delta_2^{(2)} x_3 & \dots & \delta_2^{(2)} x_{150} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta_{100}^{(2)} x_1 & \delta_{100}^{(2)} x_2 & \delta_5^{(100)} x_3 & \dots & \delta_{100}^{(2)} x_{150} \end{bmatrix}$$

$$\nabla_{W^{(1)}} = \delta^{(2)} (\vec{x})^T \tag{5}$$

Again let's verify dimensions to ensure the weights derivative matrix dimensions match those of the weights matrix:

$$\delta^{(2)} \in \mathbb{R}^{100}, (\vec{x})^T \in \mathbb{R}^{1 \times 150} \Rightarrow \nabla_W^{(1)} \in \mathbb{R}^{100 \times 150}$$

These weight matrix derivatives / error vector calculation patterns generalize for any number hidden layers. See the lecture notes for more details.

7 Layer 2 Bias

The layer 2 bias derivation becomes trival now that we have the error vector $\delta^{(2)}$:

$$\frac{\partial J}{\partial b_i^{(2)}} = \frac{\partial J}{\partial z_i^{(2)}} \cdot \frac{\partial z_i^{(2)}}{\partial b_i^{(2)}}$$

$$= \delta_i^{(2)} \cdot \frac{\partial}{\partial b_i^{(2)}} \sum_k W_{ik}^{(1)} x_k + b_i^{(2)} = \delta_i^{(2)}$$

$$\nabla_{b^{(2)}} = \delta^{(2)} \tag{6}$$

8 Input ($\vec{x}$)

The input derivation follows a similar pattern to the previous derivations. We can reuse previously computed expressions to simplify the work. Note that each x_j impacts all of the hidden node layers:

$$\frac{\partial J}{\partial x_j} = \sum_i \frac{\partial J}{\partial z_i^{(2)}} \cdot \frac{\partial z_i^{(2)}}{\partial x_j}$$

$$= \sum_i \delta_i^{(2)} \cdot \frac{\partial}{\partial x_j} \left(\sum_k W_{ik}^{(1)} x_k + b_i^{(2)} \right)$$

$$= \sum_i \delta_i^{(2)} \cdot W_{ij}^{(1)}$$

Now we can vectorize:

$$\nabla_{\vec{x}} = \begin{bmatrix} \delta_1^{(2)} W_{11}^{(1)} + \delta_2^{(2)} W_{21}^{(1)} + \delta_3^{(2)} W_{31}^{(1)} + ... \\ \delta_1^{(2)} W_{12}^{(1)} + \delta_2^{(2)} W_{22}^{(1)} + \delta_2^{(2)} W_{32}^{(1)} + ... \\ \vdots \end{bmatrix}$$

$$\nabla_{\vec{x}} = (W^{(1)})^T \delta^{(2)} \tag{7}$$

Again, let's check dimensions

$$(W^{(1)})^T \in \mathbb{R}^{150 \times 100}, \delta^{(2)} \in \mathbb{R}^{100} \Rightarrow \nabla_{\vec{x}} \in \mathbb{R}^{150}$$

This form is very similar to the $\delta^{(2)}$ calculation, the difference being there is no non-linear transformation at the input, so there is no corresponding τ' element-wise product, or if one thinks of the transformation being the identity function, there would be a $\vec{1}$ element-wise product.

9 Regularization

Part (b) of the assignment asks us to extend the above derivations when using the regularization:

$$J_{full}(\theta) = J(\theta) + J_{reg}(\theta)$$

where:

$$J_{reg}(\theta) = \frac{\lambda}{2}\left[\sum_{i,j} W_{ij}^{(2)} + \sum_{k,l} W_{kl}^{(1)}\right]$$

We can immediately see that this change will only affect $\nabla_{W^{(2)}}$ and $\nabla_{W^{(1)}}$ as derivatives with respect to other terms will be zero, e.g.

$$\frac{\partial J_{full}(\theta)}{\partial b^{(2)}} = \frac{\partial J(\theta)}{\partial b^{(2)}} + \frac{\partial J_{reg}(\theta)}{\partial b^{(2)}}$$

$$= \frac{\partial J(\theta)}{\partial b^{(2)}} + \frac{\partial}{\partial b^{(2)}}\left(\frac{\lambda}{2}\left[\sum_{i,j} W_{ij}^{(2)} + \sum_{k,l} W_{kl}^{(1)}\right]\right)$$

$$= \frac{\partial J(\theta)}{\partial b^{(2)}} = \delta^{(2)}$$

We can also see from the above steps that the previous derivations can be left unchanged, we simply need to add the regularization derivative. Starting with the layer 2 weights:

$$\frac{\partial J_{reg}}{\partial W_{ij}^{(2)}} = \frac{\partial}{\partial W_{ij}^{(2)}}\left(\frac{\lambda}{2}\left[\sum_{i,j} W_{ij}^{(2)} + \sum_{k,l} W_{kl}^{(1)}\right]\right)$$

$$= \lambda W_{ij}^{(2)}$$

Thus $\nabla_{W^{(2)}} J_{reg} = \lambda W^{(2)}$ and by identical steps $\nabla_{W^{(1)}} J_{reg} = \lambda W^{(1)}$. Thus we can extend equation 3:

$$\nabla_{W^{(2)}} J_{full} = \nabla_{W^{(2)}} J + \nabla_{W^{(2)}} J_{reg}$$
$$\nabla_{W^{(2)}} J_{full} = \delta^{(3)}(a^{(2)})^T + \lambda W^{(2)} \tag{8}$$

and equation 5:

$$\nabla_{W^{(1)}} J_{full} = \nabla_{W^{(1)}} J + \nabla_{W^{(1)}} J_{reg}$$
$$\nabla_{W^{(1)}} J_{full} = \delta^{(2)}(\vec{x})^T + \lambda W^{(1)} \tag{9}$$

Again, we could verify dimensions, but as we previously verified the $\nabla_{W^{(\cdot)}}$ are the same as their source matrices and matrix addition is element-wise, this would be trivial.