Problem Set 2, Problem 3

Kireet

November 12, 2015

Abstract

This doc contains the derivations needed to complete the coding of the recursive neural net problem. See Lecture Notes 4 for background.

Contents

1 Introduction

This document will discuss the Recurrent Neural Network (RNN) problem. RNNs are similar to feed forward nets, but they also take in as input the hidden layer values from the previous "time step":

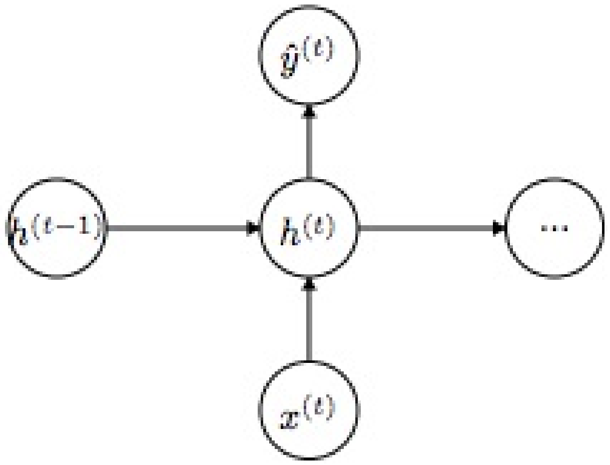

Figure 1: A snapshot of the RNN hidden layer transformation at time step t

In this problem, we are trying to build an RNN to do word prediction, each time step could be occurrence of a particular word. The RNN would output the prediction for the next word given the current input word and the accumulated context within the hidden layer.

2 Setup

Similar to the examples from the lecture notes, this problem deals with a single hidden layer RNN. The cost function is again the cross entropy loss function, however now we indicate the time step t at which the error occurred:

$$J^{(t)}(\theta) = -\sum_{j=1}^{|V|} y_j^{(t)} log(\hat{y}_j^{(t)})$$

The errors are summed across all time steps in the sequence to calculate the total loss for that sequence and then the errors for all sequences are summed to calculate the total loss for the data set.

The hidden layer at time step t is calculated as

$$h^{(t)} = \sigma\left(Hh^{(t-1)} + Lx^{(t)}\right)$$

where σ is the sigmoid function. The output layer at time step t is calculated as

$$\hat{y} = \rho\left(Uh^{(t)}\right)$$

where ρ is the softmax function. The $\hat{y}$ vector is the probabilities the next word, i.e.:

$$P(x_{t+1} = v_j | x_t, \ldots, x_1) = \hat{y}_j^{(t)}$$

$h^{(0)}$ is a fixed initialization vector. Here, $x^{(t)}$ is a one hot vector, and thus just "selects" a row from L. Thus our dimensions are:

$$H \in \mathbb{R}^{D_h \times D_h}, L \in \mathbb{R}^{D_h \times |V|}, U \in \mathbb{R}^{|V| \times D_h}$$

We can see L contains the representations of the input word, H is the hidden to hidden layer weights matrix, and U is the output weight matrix. $|V|$ is the size of the vocabulary and D_h is the size of the hidden layer.

3 Understanding Back Propagation Through Time (BPTT)

Some formulas were presented in the lectures without much justification. We'll go through them here. First we define the overall cost function for a sequence:

$$J = -\frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{|V|} y_j^{(t)} log(y_j^{(t)})$$

This is simply the sum of all the costs at each time step. if we want to calculate $\frac{\partial J}{\partial H}$, we can phrase this as

$$\frac{\partial J}{\partial H} = \sum_{t=1}^{T} \frac{\partial J^{(t)}}{\partial H}$$

Remember H is the same at each time step. So calculating $\frac{\partial J^{(t)}}{\partial H}$ is not straight forward. Let's take a look at this network "unrolled" through a few time steps:

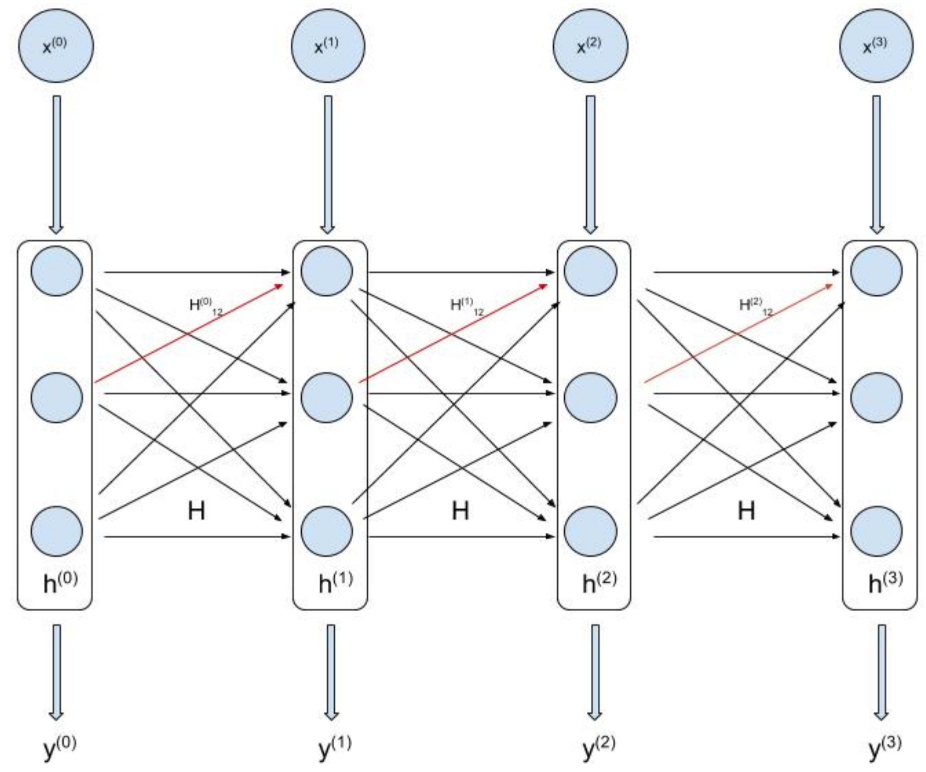

Figure 2: A RNN across 4 time steps

Here we see an RNN through 4 time steps. Each time step has an input $x^{(t)}$, the current state of the hidden layer $h^{(t)}$, and an output vector $y^{(t)}$ (really $\hat{y}^{(t)}$, but here it's y due to limited drawing tool). $h^{(0)}$ is the initialization vector and each subsequent $h^{(t)}$ depends on $h^{(t-1)}$ via the weights matrix H. Note H is the *same* weights matrix at each time step!

$J^{(3)}$ here is a function of $\hat{y}^{(3)}$. But $\hat{y}^{(3)}$is a function of all the previous hidden layers and H. The thing to do is think of things element-wise. Let's take a particular weight, H_{12} (highlighted as the red arrow). For clarity, the diagram has labeled each instance of H_{12} separately as $H_{12}^{(t)}$, but again remember it's the same weight at each time step. So to compute how J changes with H_{12}, we need to sum how it changes with each particular "instance" of H_{12} in the network:

$$\frac{\partial J^{(3)}}{\partial H_{12}} = \sum_{i=0}^{t-1} \frac{\partial J^{(3)}}{\partial H_{12}^{(i)}}$$
$$= \frac{\partial J^{(3)}}{\partial h^{(3)}} \cdot \frac{\partial h^{(3)}}{\partial H_{12}} + \frac{\partial J^{(3)}}{\partial h^{(2)}} \cdot \frac{\partial h^{(2)}}{\partial H_{12}} + \frac{\partial J^{(3)}}{\partial h^{(1)}} \cdot \frac{\partial h^{(1)}}{\partial H_{12}}$$

The important thing to see here is the first element of the sum is only considering the impact of $H_{12}^{(2)}$, $h^{(2)}$ is considered as fixed in that element. Similarly, the second term only considers $H_{12}^{(1)}$ and the final term only $H_{12}^{(0)}$. Now we can generalize the above formula for t time steps and with respect to each weight H_{ij} represented by the full matrix H:

$$\frac{\partial J^{(3)}}{\partial H} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial h^{(i)}} \cdot \frac{\partial h^{(i)}}{\partial H}$$

Applying the chain rule:

$$\frac{\partial J^{(3)}}{\partial H} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial h^{(i)}} \cdot \frac{\partial h^{(i)}}{\partial H}$$

The difficult thing here to calculate is $\frac{\partial h^{(t)}}{\partial h^{(i)}}$. Via the chain rule we can see that this is simply:

$$\prod_{k=i+1}^{t} \frac{\partial h^{(k)}}{\partial h^{(k-1)}}$$

To calculate $\frac{\partial h^{(k)}}{\partial h^{(k-1)}}$, we can perform some element wise calculations. Note $x^{(t)}$ as a subscript just means the index of the one hot element:

$$\begin{aligned}
\frac{\partial h_i^{(t)}}{\partial h_j^{(t-1)}} &= \frac{\partial}{\partial h_j^{(t-1)}} \sigma\left(Hh^{(t-1)} + Lx^{(t)}\right)_i \\
&= \frac{\partial}{\partial h_j^{(t-1)}} \sigma\left(\sum_k H_{ik} h_k^{(t-1)} + L_{x^{(t)}k}\right) \\
&= \sigma'\left(\sum_k H_{ik} h_k^{(t-1)} + L_{ik}\right) \cdot H_{ij}
\end{aligned}$$

Let's define the $\sigma'(...)$ term as σ_i' for notational convenience. Then in Jacobian matrix form:

$$\frac{\partial h^{(k)}}{\partial h^{(k-1)}} = \begin{bmatrix} \frac{\partial h_1^{(k)}}{\partial h_1^{(k-1)}} & \frac{\partial h_1^{(k)}}{\partial h_2^{(k-1)}} & \cdots & \frac{\partial h_1^{(k)}}{\partial h_{D_h}^{(k-1)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_{D_h}^{(k)}}{\partial h_1^{(k-1)}} & \frac{\partial h_{D_h}^{(k)}}{\partial h_2^{(k-1)}} & \cdots & \frac{\partial h_{D_h}^{(k)}}{\partial h_{D_h}^{(k-1)}} \end{bmatrix}$$

$$\frac{\partial h^{(k)}}{\partial h^{(k-1)}} = \begin{bmatrix} \sigma_1' H_{11} & \sigma_1' H_{12} & \sigma_1' H_{13} & \ldots & \sigma_1' H_{1D_h} \\ \sigma_2' H_{21} & \sigma_2' H_{22} & \sigma_2' H_{23} & \ldots & \sigma_2' H_{2D_h} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{D_h}' H_{D_h 1} & \sigma_{D_h}' H_{D_h 2} & \sigma_{D_h}' H_{D_h 3} & \ldots & \sigma_{D_h}' H_{D_h D_h} \end{bmatrix}$$

$$= diag(\sigma') \cdot H$$

The diag as the leading matrix in a multiplication simply weights the 2nd matrix's rows by the diagonal elements of the first. In later derivations, it will often be useful to phrase this derivative as the inverse Jacobian. In that case we have:

$$\frac{\partial h^{(k)}}{\partial h^{(k-1)}} = H^{\top} \cdot diag(\sigma')$$

4 Part a

Part (a) asks us to show that minimizing the cross entropy loss function from Section 2 will minimize perplexity, given as :

$$PP(y, \hat{y}) = \frac{1}{\sum_{j=1}^{|V|} y_j^{(t)} \cdot \hat{y}_j^{(t)}}$$

Given a correct class k, we have:

$$J(\theta) = -log(\hat{y}_k^{(t)}), PP(y, \hat{y}) = \frac{1}{\hat{y}_k^{(t)}}$$

It's clear that as $\hat{y}$ increases, both values monotonically decrease. Thus minimizing the cross entropy loss should minimize perplexity.

5 Part b

Part b asks us to calculate the gradients a time step t, with the previous time steps considered fixed. As before, we'll proceed element wise and then convert the results to vector notation/operations. We'll again change some notation. U and H will be unchanged but we'll again use a and z to refer to layer outputs and inputs, respectively. $a^{(h,t)}$ and $z^{(h,t)}$ will refer to the output and input of the hidden layer at time t. Similarly, (x, t) and (o, t) superscript will refer to the input and output layer at time t.

5.1 Output Layer (U) Gradients

First we calculate $\frac{\partial J^{(t)}}{\partial U}$. As we've seen before, the derivative of the softmax and cross entropy loss combination simply results in a $\hat{y} - y$ vector. We term the input to the final output layer at time t as $z^{(o,t)}$

$$\frac{\partial J^{(t)}}{\partial U} = \frac{\partial J^{(t)}}{\partial z^{(o,t)}} \cdot \frac{\partial z^{(o,t)}}{\partial U} \triangleq \delta^{(o,t)} \cdot \frac{\partial z^{(o,t)}}{\partial U}$$

Element wise for the 2nd term:

$$\frac{\partial z_i^{(o,t)}}{\partial U_{ij}} = \frac{\partial}{\partial U_{ij}} \sum_k U_{ik} h_k^{(t)}$$

$$= h_j^{(t)}$$

In matrix form we have:

$$\nabla_U = \begin{bmatrix} \delta_1^{(o,t)} h_1^{(t)} & \delta_1^{(o,t)} h_2^{(t)} & \delta_1^{(o,t)} h_3^{(t)} & \dots & \delta_1^{(o,t)} h_{D_h}^{(t)} \\ \delta_2^{(o,t)} h_1^{(t)} & \delta_2^{(o,t)} h_2^{(t)} & \delta_2^{(o,t)} h_3^{(t)} & \dots & \delta_2^{(o,t)} h_{D_h}^{(t)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta_{|V|}^{(o,t)} h_1^{(t)} & \delta_{|V|}^{(o,t)} h_2^{(t)} & \delta_{|V|}^{(o,t)} h_3^{(t)} & \dots & \delta_{|V|}^{(o,t)} h_{D_h}^{(t)} \end{bmatrix}$$

Thus:

$$\nabla_U = (\delta^{(o,t)})(h^{(t)})^\top \tag{1}$$

We can check the dimensions and see we end up with $\nabla_U \in \mathbb{R}^{|V| \times D_h}$, which is correct as it matches the dimensions of U.

5.2 Hidden Layer Weights (H) Gradients

Next we calculate $\frac{\partial J^{(t)}}{\partial H}$, again element-wise. We define $z^{(h,t)}$ as the input to the hidden layer:

$$\frac{\partial J^{(t)}}{\partial H_{ij}} = \frac{\partial J^{(t)}}{\partial z_i^{(h,t)}} \cdot \frac{\partial z_i^{(h,t)}}{\partial H_{ij}}$$

First we derive the first term:

$$\frac{\partial J^{(t)}}{\partial z_i^{(h,t)}} = \frac{\partial J^{(t)}}{\partial a_i^{(h,t)}} \cdot \frac{\partial a_i^{(h,t)}}{\partial z_i^{(h,t)}}$$

$$= \frac{\partial J^{(t)}}{\partial a_i^{(h,t)}} \cdot \sigma'(z_i^{(h,t)})$$

$$= \sigma'(z_i^{(h,t)}) \cdot \left(\frac{\partial J^{(t)}}{\partial z_i^{(o,t)}} \cdot \frac{\partial z_i^{(o,t)}}{\partial a_i^{(h,t)}} \right)$$

$$= \sigma'(z_i^{(h,t)}) \cdot \sum_k \left[\delta_k^{(o,t)} \frac{\partial}{\partial a_i^{(h,t)}} \left(\sum_l U_{kl} a_l^{(h,t)} \right) \right]$$

$$\delta_i^{(h,t)} \triangleq \sigma'(z_i^{(h,t)}) \cdot \sum_k \delta_k^{(o,t)} U_{ki}$$

The second term derivation:

$$\frac{\partial z_i^{(h,t)}}{\partial H_{ij}} = \frac{\partial}{\partial H_{ij}} \left(\sum_k H_{ik} a_k^{(h,t-1)} + \sum_l L_{il} x_l^{(t)} \right)$$

$$= a_j^{(h,t-1)}$$

Combining these, we have:

$$\frac{\partial J^{(t)}}{\partial H_{ij}} = \delta_i^{(h,t)} \cdot a_j^{(h,t-i)}$$

Similar to the layer 1 weights in the previous feed forward network problem, the vectorization of this is:

$$\delta^{(h,t)} = \sigma'(z^{(h,t)}) \circ (U^\top \delta^{(o,t)}) \tag{2}$$

$$\nabla_H = \delta^{(h,t)}(a^{(h,t-1)})^\top) \tag{3}$$

We can check the dimensions, $U^\top \in \mathbb{R}^{D_h \times |V|}$ and $delta^{(o,t)} \in \mathbb{R}^{|V|}$ means that their product $\in \mathbb{R}^{D_h}$. This is correct, we should have an error term for each hidden layer unit. For equation 3, we see that $\delta^{(h,t)} \in \mathbb{R}^{D_h}$ from equation 2 and $a^{(h,t-1)})^\top \in \mathbb{R}^{1\times D_h}$ which means $\nabla_H \in \mathbb{R}^{D_h \times D_h}$. This matches the dimensions of H.

5.3 Input ($L_{x^{(t)}}$) Gradients

$x^{(t)}$ is simply a one hot vector that identifies the input word at time t. Thus multiplying this vector by L simply extracts a word vector $L_{x^{(t)}} \in \mathbb{R}^{D_h}$. Unlike the feed forward network, we can see that there is no weights matrix used to combine this word vector into each hidden unit. Instead the i^{th} hidden unit's sigmoid function simply receives the i^{th} value from the word vector. Now we start our element wise derivation, using this fact:

$$\frac{\partial J^{(t)}}{\partial L_{x^{(t)}i}} = \sum_k \frac{\partial J^{(t)}}{\partial z_k^{(t)}} \cdot \frac{\partial z_k^{(t)}}{\partial L_{x^{(t)}i}}$$

$$= \frac{\partial J^{(t)}}{\partial z_i^{(t)}} \cdot \frac{\partial z_i^{(t)}}{\partial L_{x^{(t)}i}}$$

$$= \delta_i^{(h,t)} \cdot \frac{\partial}{\partial L_{x^{(t)}i}} \left(\sum_k H_{ik} a_k^{(h,t-1)} + L_{x^{(t)}i} \right)$$

$$= \delta_i^{(h,t)}$$

The vectorization is simply:

$$\nabla_{L_{x^{(t)}}} = \delta^{(h,t)} \tag{4}$$

5.4 Previous Time Step ($h^{(t-1)}$) Gradients

Finally we are asked to compute the gradients with respect to the previous time step. In our notation, this is $a^{(h,t-1)}$.

$$\frac{\partial J^{(t)}}{\partial a_j^{(h,t-1)}} = \sum_i \frac{\partial J^{(t)}}{\partial z_i^{(h,t)}} \cdot \frac{\partial z_i^{(h,t)}}{\partial a_j^{(h,t-1)}}$$

$$= \sum_i \delta_i^{(h,t)} \cdot \frac{\partial}{\partial a_j^{(h,t-1)}} \left(\sum_k H_{ik} a_k^{(h,t-1)} + L_{x^{(t)}i} \right)$$

$$= \sum_i \delta_i^{(h,t)} \cdot H_{ij}$$

Before vectorizing, let's examine this formula. We see that for the error gradient at a hidden layer node at time $t-1$ is simply a weighted sum of the errors at time step t, with the weight being the appropriate value in the weights matrix. That is the weight from the $t-1$ node to all the nodes at time t. This is essentially the same logic as the feed forward network derivations.

Now in matrix form:

$$\nabla_{a^{(h,t-1)}} = \begin{bmatrix} \delta_1^{(h,t)} H_{11} + \delta_2^{(h,t)} H_{21} + \delta_3^{(h,t)} H_{31} + \cdots + \delta_{D_h}^{(h,t)} H_{D_h 1} \\ \delta_1^{(h,t)} H_{12} + \delta_2^{(h,t)} H_{22} + \delta_3^{(h,t)} H_{32} + \cdots + \delta_{D_h}^{(h,t)} H_{D_h 2} \\ \vdots \\ \delta_1^{(h,t)} H_{1D_h} + \delta_2^{(h,t)} H_{2D_h} + \delta_3^{(h,t)} H_{3D_h} + \cdots + \delta_{D_h}^{(h,t)} H_{D_h D_h} \end{bmatrix}$$

$$\nabla_{a^{(h,t-1)}} = H^\top \delta^{(h,t)} \tag{5}$$

6 Part c

Part c asks us to calculate the $t-1$ derivatives for $L_{x^{(t-1)}}$ and H.

6.1 Input ($L_{x^{(t-1)}}$) Gradients

To calculate $\frac{\partial J^{(t)}}{\partial L_{x^{(t-1)}}}$, we can use the results from the previous part:

$$\frac{\partial J^{(t)}}{\partial L_{x^{(t-1)}}} = \frac{\partial J^{(t)}}{\partial a^{(h,t-1)}} \cdot \frac{\partial a^{(h,t-1)}}{\partial z^{(h,t-1)}} \cdot \frac{\partial z^{(h,t-1)}}{\partial L_{x^{(t-1)}}}$$

$$= \frac{\partial J^{(t)}}{\partial a^{(h,t-1)}} \cdot \frac{\partial a^{(h,t-1)}}{\partial z^{(h,t-1)}} \cdot \frac{\partial}{\partial L_{x^{(t-1)}}} \left(H a^{(t-1)} + L_{x^{(t-1)}} \right)$$

$$\nabla_{L_{x^{(h,t-1)}}} = \sigma'(z^{(h,t-1)}) \circ H^\top \delta^{(h,t)} \tag{6}$$

Note this is also $\frac{\partial J^{(t)}}{\partial z^{(h,t-1)}}$ and thus is our $\delta^{(h,t-1)}$:

$$\delta^{(h,t-1)} = \sigma'(z^{(h,t-1)}) \circ H^\top \delta^{(h,t)} \tag{7}$$

6.2 Hidden Layer Weights (H) Gradients

We again attempt to use previously calculated values to calculate the hidden layer weights matrix gradients at time step $t-1$:

$$\left.\frac{\partial J^{(t)}}{\partial H}\right|_{t-1} = \frac{\partial J^{(t)}}{\partial z^{(h,t-1)}} \cdot \frac{\partial z^{(h,t-1)}}{\partial H}$$

The first term we know to be $\delta^{(h,t-1)}$. The second term derivation is essentially identical to the latter steps of subsection 5.2, resulting in $(a^{(h,t-2)})^\top$. Thus we have:

$$\nabla_H\big|_{t-1} = \delta^{(h,t-1)} \cdot (a^{(h,t-2)})^\top \tag{8}$$

Using these equation, we can backprop arbitrarily back in time by continually calculating the δ error values at each time step.