

Problem Set 2, Problem 3

Kireet

October 28, 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	3
2	Setup	3
3	Understanding Back Propagation Through Time (BPTT)	4
4	Part a	7
5	Part b	7

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":

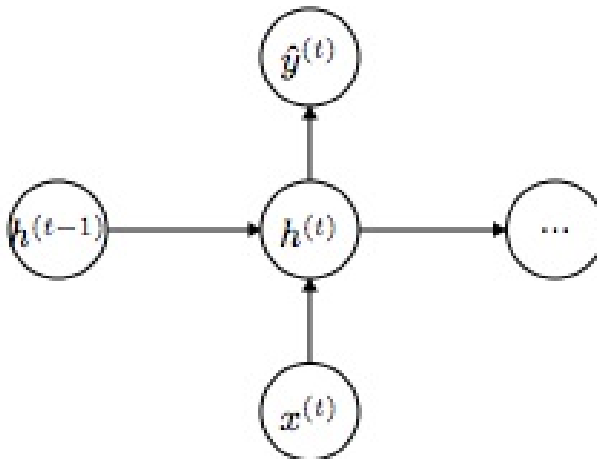


Figure 1: A snapshot of the RNN hidden layer transformation at time step t

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

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

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

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

$h^{(0)}$ 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:

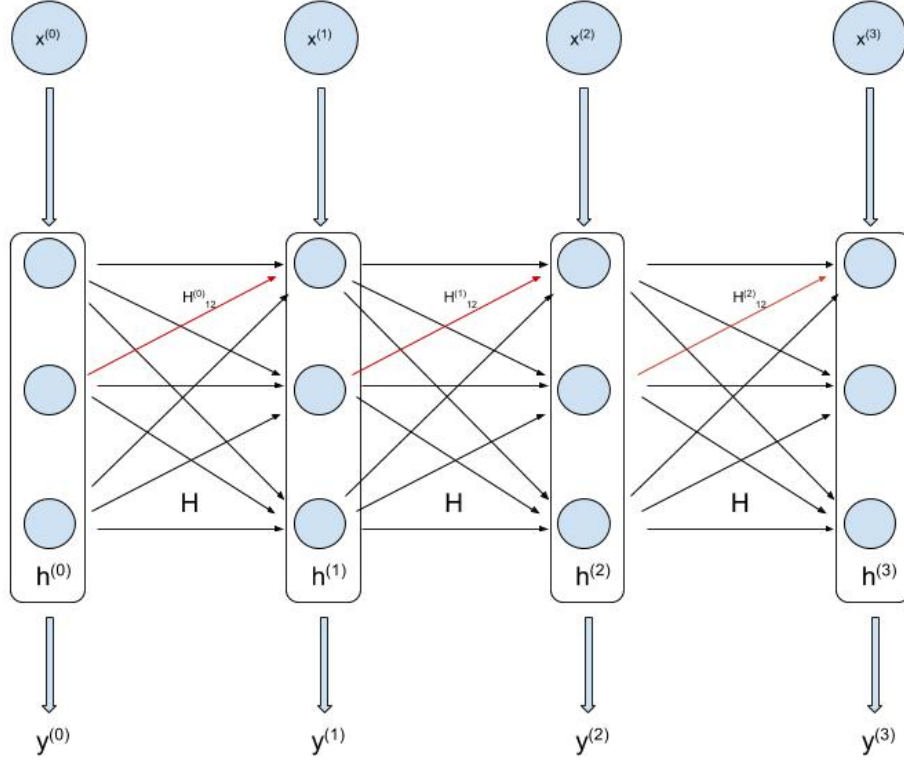


Figure 2: A RNN across 4 time steps

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:

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

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:

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

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

$$\begin{aligned} \frac{\partial h^{(k)}}{\partial h^{(k-1)}} &= \begin{bmatrix} \frac{\partial h_1^{(k)}}{\partial h_1^{(k-1)}} & \frac{\partial h_1^{(k)}}{\partial h_2^{(k-1)}} & \cdots & \frac{\partial h_1^{(k)}}{\partial h_{D_h}^{(k-1)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_{D_h}^{(k)}}{\partial h_1^{(k-1)}} & \frac{\partial h_{D_h}^{(k)}}{\partial h_2^{(k-1)}} & \cdots & \frac{\partial h_{D_h}^{(k)}}{\partial h_{D_h}^{(k-1)}} \end{bmatrix} \\ \frac{\partial h^{(k)}}{\partial h^{(k-1)}} &= \begin{bmatrix} \sigma'_1 H_{11} & \sigma'_1 H_{12} & \sigma'_1 H_{13} & \cdots & \sigma'_1 H_{1D_h} \\ \sigma'_2 H_{21} & \sigma'_2 H_{22} & \sigma'_2 H_{23} & \cdots & \sigma'_2 H_{2D_h} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma'_{D_h} H_{D_h 1} & \sigma'_{D_h} H_{D_h 2} & \sigma'_{D_h} H_{D_h 3} & \cdots & \sigma'_{D_h} H_{D_h D_h} \end{bmatrix} \end{aligned}$$

$$= \text{diag}(\sigma') \cdot H$$

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:

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

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:

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

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 .

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 $z^{(3,t)}$

$$\frac{\partial J^{(t)}}{\partial U} = \frac{\partial J^{(t)}}{\partial z^{(3,t)}} \cdot \frac{\partial z^{(3,t)}}{\partial U} \triangleq \delta^{(3,t)} \cdot \frac{\partial z^{(3,t)}}{\partial U}$$

Element wise for the 2nd term:

$$\frac{\partial z_i^{(3,t)}}{\partial U_{ij}} = \frac{\partial}{\partial U_{ij}} \sum_k U_{ik} h_k^{(t)}$$

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

In matrix form we have:

$$\nabla_U = \begin{bmatrix} \delta_1^{(3,t)} h_1^{(t)} & \delta_1^{(3,t)} h_2^{(t)} & \delta_1^{(3,t)} h_3^{(t)} & \dots & \delta_1^{(3,t)} h_{D_h}^{(t)} \\ \delta_2^{(3,t)} h_1^{(t)} & \delta_2^{(3,t)} h_2^{(t)} & \delta_2^{(3,t)} h_3^{(t)} & \dots & \delta_2^{(3,t)} h_{D_h}^{(t)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta_{|V|}^{(3,t)} h_1^{(t)} & \delta_{|V|}^{(3,t)} h_2^{(t)} & \delta_{|V|}^{(3,t)} h_3^{(t)} & \dots & \delta_{|V|}^{(3,t)} h_{D_h}^{(t)} \end{bmatrix}$$

Thus:

$$\nabla_U = (\delta^{(3,t)})(h^{(t)})^\top \quad (1)$$

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 .

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

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