METL – TP2
Recurrent Neural Networks for Language Modeling

Alexandre Kabbach
alexandre.kabbach@unige.ch

Paola Merlo
paola.merlo@unige.ch

05.03.2020

Evaluation: You are allowed an unlimited number of submissions in order to receive feedback. When you are satisfied with your work, you can ask for it to be graded. You can also ask for it to be graded upon a single submission, without receiving feedback. All your TPs must have been graded and must have received an average grade of at least 4/6 for you to register for the METL exam. Indicative deadline: March 18 2020 (this TP should take you two weeks).

1 Model

Let us consider a *language model*, which predicts, given a set of input words $x_1, \ldots, x_t$, the following word x_{t+1} by modeling:

$$p(x_{t+1} = v_j | x_t, \ldots, x_1) \tag{1}$$

where v_j is a word in the vocabulary V of size $|V|$.

In this TP, we will rely on the RNN language model introduced by Mikolov et al. (2010).[1] The model is a RNN language model which uses feedback information in the hidden layer to model the "history" $x_t, x_{t-1}, \ldots, x_1$. The model is formally defined, for $t = 1, \ldots, n-1$ by:

$$e^{(t)} = x^{(t)} L \tag{2}$$

$$h^{(t)} = sigmoid(h^{(t-1)} H + e^{(t)} I + b_1) \tag{3}$$

$$\hat{y}^{(t)} = softmax(h^{(t)} U + b_2) \tag{4}$$

$$p(x_{t+1} = v_j | x_t, \ldots, x_1) = \hat{y}_j^{(t)} \tag{5}$$

where $h^{(0)} = h_0 \in \mathbb{R}^{D_h}$ is some initialization vector for the hidden layer and $x^{(t)} L$ is the product of L with the one-hot row-vector $x^{(t)}$ representing the index of the current word. The parameters of the model are:

$$L \in \mathbb{R}^{|V| \times d} \quad H \in \mathbb{R}^{D_h \times D_h} \quad I \in \mathbb{R}^{d \times D_h} \quad b_1 \in \mathbb{R}^{D_h} \quad U \in \mathbb{R}^{D_h \times |V|} \quad b_2 \in \mathbb{R}^{|V|} \tag{6}$$

where L is the embedding matrix, I the input word representation matrix, H the hidden transformation matrix, and U the output word representation matrix. b_1 and b_2 are biases, d is the embedding dimension, $|V|$ the vocabulary size, and D_h the hidden layer dimension.

[1] http://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf

The ouptut vector $\hat{y}^{(t)} \in \mathbb{R}^{|V|}$ is a probability distribution over the vocabulary, and we optimize the unregularized cross-entropy loss via:

$$J^{(t)}(\theta) = CE(y^{(t)}, \hat{y}^{(t)}) = -\sum_{i=1}^{|V|} y_i^{(t)} \log \hat{y}_i{}^{(t)} \tag{7}$$

where $y^{(t)}$ is the one-hot vector corresponding to the target word (which is here equal to x_{t+1}). Usually, to evaluate model performance, people rely on the previously defined point-wise loss, and sum (or average) the cross-entropy loss across all examples in a sequence, and across all sequences in the dataset.

2 Notes on gradients and backprop

In the below questions it is asked to compute the gradient of a scalar (the cost function) with respect to a matrix (of weights). This is at the core of the backpropagation algorithm. Recall the backpropagation algorithm where the weights W and the bias b are updated with respect to a cost function J given a learning rate α:

$$W = W - \alpha \frac{\partial J}{\partial W} \tag{8}$$

$$b = b - \alpha \frac{\partial J}{\partial b} \tag{9}$$

The backpropagation algorithm therefore requires the scalar-by-matrix gradient $\frac{\partial J}{\partial W}$ to be of a size consistent with the size of W. Therefore, it is important to understand that the gradient matrix $\frac{\partial J}{\partial W}$ is actually an **abuse of notation** as it is understood as the following matrix, considering W to be a $n \times m$ matrix:

$$\frac{\partial J}{\partial W} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}} & \cdots & \frac{\partial J}{\partial W_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial W_{n1}} & \cdots & \frac{\partial J}{\partial W_{nm}} \end{bmatrix} \tag{10}$$

Now to ease the computation we will introduce a vector $z = xW$ so that:

$$\frac{\partial J}{\partial W} = \frac{\partial z}{\partial W} \frac{\partial J}{\partial z} \tag{11}$$

(Note that the chain rule for the denominator layout notation is in reverse order, and that the aforementioned equation is also somehow an abuse of notation).
We saw in TP1 that in the case where $J = CE(y, \hat{y})$ and $\hat{y} = softmax(hW_2 + b_2)$ and given $z = hW_2 + b_2$:

$$\frac{\partial J}{\partial z} = \frac{\partial CE}{\partial z} = \hat{y} - y \tag{12}$$

Note that if z is a vector of size $1 \times m$, $\hat{y}$ will also be a vector of size $1 \times m$ and so will be $\frac{\partial J}{\partial z}$.
All we need to do now is to compute $\frac{\partial z}{\partial W} = \frac{\partial xW}{\partial W}$, which we do for all element of w_{ij} with:

$$z_k = \sum_{l=1}^{n} x_l W_{lk} \tag{13}$$

$$\frac{\partial z_k}{\partial W_{ij}} = \sum_{l=1}^{n} x_l \frac{\partial W_{lk}}{\partial W_{ij}} \tag{14}$$

where z_k is the k^{th} column component of the row vector z and x is a $1 \times n$ row vector and W is a $n \times m$ matrix. Given that:

$$\frac{\partial W_{lk}}{\partial W_{ij}} = \begin{cases} 1 & \text{if } i = l \text{ and } j = k \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Let us detail the calculations:

$$z = [x_1 w_{11} + \ldots + x_n w_{n1}, \cdots, x_1 w_{1m} + \ldots + x_n w_{nm}] \tag{16}$$

$$\begin{aligned} z_1 &= x_1 w_{11} + \ldots + x_n w_{n1} = \sum_{l=1}^{n} x_l W_{l1} \\ &\cdots \\ z_m &= x_1 w_{1m} + \ldots + x_n w_{nm} = \sum_{l=1}^{n} x_l W_{lm} \end{aligned} \tag{17}$$

Try deriving with respect to w_{11} and w_{12}, etc. and you should find that $\frac{\partial z_k}{\partial W_{ij}} = x_i$. Therefore, we can show that::

$$\forall (i,j) \in [1,n] \times [1,m] \quad \frac{\partial J}{\partial W_{ij}} = \frac{\partial z}{\partial W_{ij}} \frac{\partial J}{\partial z} = \sum_{k=1}^{m} \frac{\partial z_k}{\partial W_{ij}} \Big(\frac{\partial J}{\partial z}\Big)_k = x_i \Big(\frac{\partial J}{\partial z}\Big)_j \tag{18}$$

Recall that $\frac{\partial J}{\partial z}$ is a $1 \times m$ vector and that the chain rule for the denominator layout notation is in reverse order. We conclude that:

$$\frac{\partial J}{\partial W} = x^\top \frac{\partial J}{\partial z} \tag{19}$$

with the aforementioned abuse of notation (see eq. 10). You can double check consistency of dimensions to be sure ($x : 1 \times n$, $\frac{\partial J}{\partial z} : 1 \times m$ and therefore $\frac{\partial J}{\partial W} : n \times m$).

3 Questions

1. Conventionally, when reporting performance of a language model, people rely on the notion of *perplexity* PP defined as the inverse probability of the correct word according to the model distribution p:

$$PP^{(t)}(y^{(t)}, \hat{y}^{(t)}) = \frac{1}{p(x_{t+1}^{pred} = x_{t+1} | x_t, \ldots, x_1)} = \frac{1}{\sum_{j=1}^{|V|} y_j^{(t)} \cdot \hat{y}_j^{(t)}} \tag{20}$$

   (a) Demonstrate that you can derive perplexity from the cross-entropy loss (*Hint: remember that $y^{(t)}$ is one-hot*) and that minimizing the arithmetic mean cross-entropy loss will also minimize the geometric mean perplexity across a given dataset.

   (b) For a vocabulary of $|V|$ words, what would you expect perplexity to be if your model predictions were completely random? Compute the corresponding cross-entropy loss for $|V| = 2000$ and $|V| = 10000$.

2. (a) Compute the gradients for all the model parameters at a single point in time t:

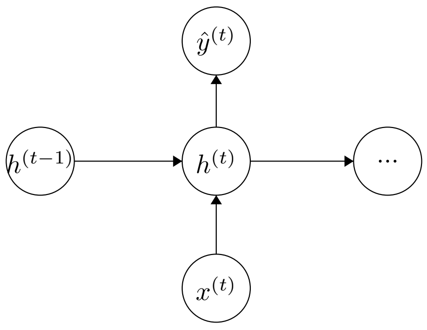
$$\frac{\partial J^{(t)}}{\partial U} \quad \frac{\partial J^{(t)}}{\partial b_2} \quad \frac{\partial J^{(t)}}{\partial L_{x^{(t)}}} \quad \left.\frac{\partial J^{(t)}}{\partial I}\right|_{(t)} \quad \left.\frac{\partial J^{(t)}}{\partial H}\right|_{(t)} \quad \left.\frac{\partial J^{(t)}}{\partial b_1}\right|_{(t)} \tag{21}$$

where $L_{x^{(t)}}$ is the column of L corresponding to the current word $x^{(t)}$ and $\big|_{(t)}$ denotes the gradient for the appearance of that parameter at time t. $h^{(t-1)}$ is taken to be fixed and you need not backpropagate to earlier timesteps just yet, you will do that in question 3.

(b) Compute the derivative with respect to the *previous* hidden layer value:

$$\frac{\partial J^{(t)}}{\partial h^{(t-1)}} \tag{22}$$

3. Consider the following sketch of the RNN at a single timestep:



(a) Draw the "unrolled" network for 3 timesteps

(b) Compute the backpropagation-through-time gradients:

$$\frac{\partial J^{(t)}}{\partial L_{x^{(t-1)}}} \quad \left.\frac{\partial J^{(t)}}{\partial H}\right|_{(t-1)} \quad \left.\frac{\partial J^{(t)}}{\partial I}\right|_{(t-1)} \quad \left.\frac{\partial J^{(t)}}{\partial b_1}\right|_{(t-1)} \tag{23}$$

Express the derivatives in terms of the error term:

$$\delta^{(t-1)} = \frac{\partial J^{(t)}}{\partial h^{(t-1)}} \tag{24}$$

Note that the true gradient with respect to a training example requires us to run backpropagation all the way back to $t = 0$. In practice, however, we generally truncate this and only backpropagate for a fixed number $\tau \approx 3$ to 5 timesteps.

4. Given $h^{(t-1)}$. Express in big-O notation in terms of the dimensions d, D_h and $|V|$ (recall Equation 6):

(a) the number of operations required to perform one step of forward propagation to compute $J^{(t)}(\theta)$

(b) the number of operations required to perform one step of backpropagation

(c) the number of operations required to perform τ steps of backpropagation

What is the "slow" step?