Deep Learning 1, Homework 2

Mathieu Bartels
11329521
UvA
mathieubartels@gmail.com

1 Vanilla RNN versus LSTM

1.1

$$\begin{aligned}
\frac{\partial L^{(t)}}{\partial W_{ph}} &= \frac{\partial L^{(t)}}{\partial \hat{y}_k^{(t)}} \frac{\partial \hat{y}_k^{(t)}}{\partial W_{ph}} \\
&= \frac{-y_k^{(t)}}{\hat{y}_k^{(t)}} * \hat{y}_k^{(t)}(I_{kj} - \hat{y}_j^{(t)}) * h^{(t)} \\
&= -y_k^{(t)}(I_{kj} - \hat{y}_j^{(t)}) * h^{(t)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L^{(t)}}{\partial W_{hh}} &= \sum_{s=0}^{t} \frac{\partial L}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial P_k^{(t)}} \frac{\partial P_k^{(t)}}{\partial h_k^{(t)}} \frac{\partial h_k^{(t)}}{\partial h_s^{(t)}} \frac{\partial h_s^{(t)}}{\partial W_{hh}} \\
\frac{\partial L}{\partial \hat{y}_k} &= \frac{-y_k^{(t)}}{\hat{y}_k^{(t)}} \\
\frac{\partial \hat{y}_k}{\partial P_k^{(t)}} &= -\hat{y}_k^{(t)}(I_{kj} - \hat{y}_j^{(t)}) \\
\frac{\partial P_k^{(t)}}{\partial h_k^{(t)}} &= W_{ph} \\
\frac{\partial h_k^{(t)}}{\partial h_s^{(t)}} &= \prod_{j=k+1}^{t} \frac{\partial h_j^{(t)}}{\partial h_{j-1}^{(t)}} \\
\frac{\partial h_s^{(t)}}{\partial W_{hh}} &= (1 - h^{2(t)}) * h^{(t-1)}
\end{aligned}$$

$$\frac{\partial L^{(t)}}{\partial W_{hh}} = \sum_{s=0}^{t} y_k^{(t)}(I_{kj} - \hat{y}_j^{(t)}) * W_{ph} * \prod_{j=k+1}^{t} \frac{\partial h_j^{(t)}}{\partial h_{j-1}^{(t)}} * (1 - h^{2(t)}) * h^{(t-1)}$$

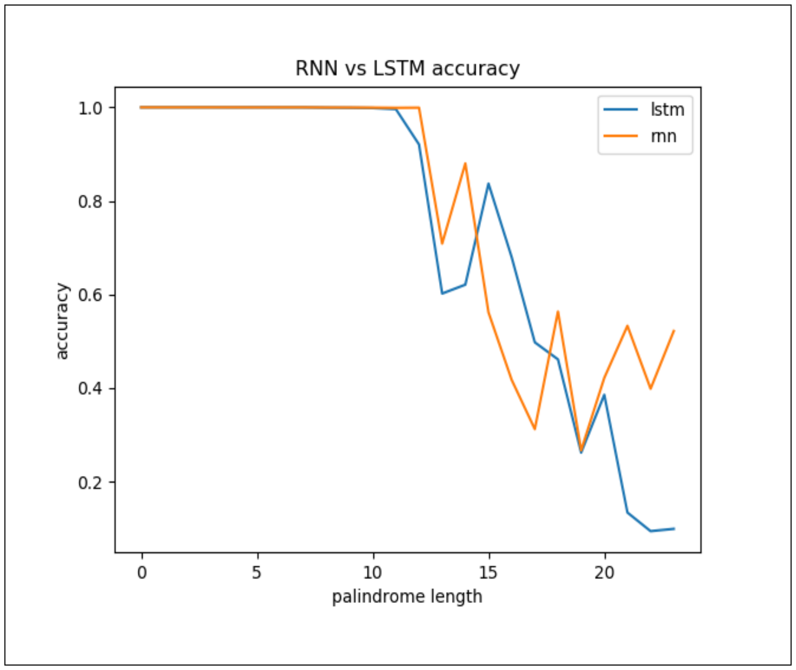

Figure 1: RNN and LSTM accuracy

1.2

1.3

Figure 3 shows the accuracy of the RNN and LSTM model. A test set of 4000 examples is used. A learning rate of 0.001 for palindromes length smaller than 10, 0.01 otherwise. The batch size is 1024. The accuracy is based on 10 different seeds.

1.4 To improve optimization of neural networks, many variants of stochastic gradient descent have been proposed. Two popular optimizers are RMSProp and Adam and/or the use of momentum. In practice, these methods are able to converge faster and obtain better local minima. In your own words, write down the benefits of such methods in comparison to vanilla stochastic gradient descent. Your answer needs to touch upon the concepts of momentum and adaptive learning rate.

While optimizing a neural net, we update our weights in the direction of the gradient with respect to the loss function. However, we don't just add this gradient, we multiply it with a learning rate. This learning rate is a scalar that determines the magnitude of a single update. With SGD we have a constant update magnitude. We know that the first examples need a higher magnitude than later examples, so a adaptive learning rate would be needed. This learning rate has conditions where it changes as the number of batches grows. Adam and RMSProp have these adaptive learning rates. RMSProp boosts small gradients and suppresses small gradients. Adam is similar to RMSProp, however it uses momentum and correction bias. Momentum is the idea that previous derivatives have effect on the current gradient. So that when the gradient is at the beginning of a valley, it's boosted to the depth.

1.5 Question 1.5

1. The LSTM extends the vanilla RNN cell by adding four gating mechanisms. Those gating mechanisms are crucial for successfully training recurrent neural networks. The LSTM has an input modulation gate g(t),input gate i(t),forget gate f(t)and output gate o(t). For each of these gates, write down a brief explanation of their purpose; explicitly discuss the non-linearity they use and motivate why this is a good choice

g(t) = Is a non-linear module that uses the tanh and decides what new information should be added to the memory. The Tanh is used so negative values can be used
i(t) = This gate is combined with the g(t) module to decide about new information that should be added to memory. The sigmoid is used to map the value between 0, not important and 1, must keep.
f(t) = The forget gate decides what information can stay in memory based on the current input. The sigmoid is used to map the value between 0, not important and 1, must keep.
o(t) = Is the output gate modulates the output. The sigmoid is used to map the value between 0, not important and 1, must keep.

2. Given the LSTM cell as defined by the equations above and an input sample x where T denotes the sequence length and d is the feature dimensionality. Let n denote the number of units in the LSTM and m represents the batch size. Write down the formula for the total number of trainable parameters in the LSTM cell as defined above.
   $4(n^2 + dn + n)$

1.6

In figure 3 the accuracy of both models are shown we expect the lstm model to perform better because the gradients vanish less quickly. However this plot doesn't show this. I expect this is the case because neither of the two models are optimised. When tweaking hyper parameters and biases the lstm can outperform the the Vanilla rnn. What I have seen in my experiments is that the standard deviation of the accuracy of the lstm is a lot lower. This . means the learning of the lstm model happens slower

1.7

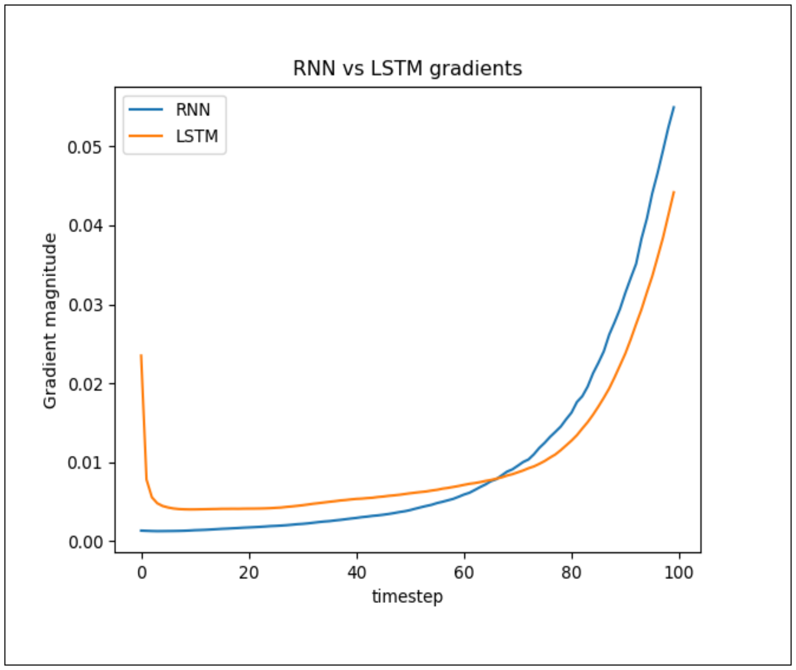

Figure 2: RNN and LSTM gradients

In Figure 2 The gradients are shown of both models. We see the RNN and lstm follow the same curve, the rnn has a bit higher gradients at the start, this is because of the normalisation that is happening. The LSTM has a spike at around timestep 0. That's perfect! The lstm is able to learn

2

1. For this experiment the Adam optimiser is used, the default learning rate and batchsize are used. 0.002 and 64 respectively. The text used for this experiment is the King James Bible. The lstm network has 128 hidden units and 2 lstm layers. I chose Adam over RMSProp because of the momentum and correction bias, these help for quicker convergence.

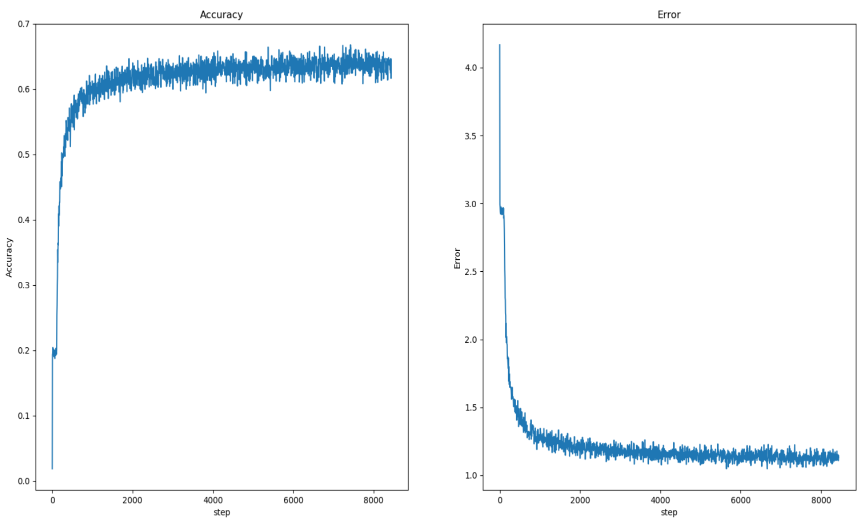

Figure 3: LSTM Error and Accuracy on the grimms fairy tails

2. At the start of the training process simple sentences are generated: "6 And the sea, and the sea, and", in this example we see a lot of repetition. When training a while longer we get sentences like "Mordecai the son of Joshua the", "Uzza the son of Abraham the son", "$ and the sons of Jerusalem the". In the last sentence we can see the short memory of the lstm. The long therm memory remembered that in the bible everybody is the son of. The short term memory chooses who is whose son. In the last example the lstm starts a new verse "1:13 And the LORD said unto the".
   Because it would be interesting to see how this verse finishes we can increase the generation length. However, making the sentence longer reveals the lstm made empty promises "1:12 And the LORD said unto the LORD said unto". But sometimes, by chance we can get some real longer sentences! "Now the son of Joab said unto them, If thou sh".
   Making the sentences shorter just leaves us with some words. "of the sons of A", "Now the LORD sai". So this is not so interesting.

3. τ Is a scaling parameter applied before the softmax, at first it seems like this τ has no effect, it seems like a linear scaling. But after calculating some examples myself I found the effect of the τ to be very intuitive. Let's take a three letter vocabulary, a,b,c. Let's say the output of the linear layer looks as following, [3,4,1]. After applying the softmax, [0.26,0.70,0.04]. Now lets apply a large temperature, $\tau = 100$. The output becomes [300,400,100] and the softmax becomes [0, 1, 0], when sampling this distribution it will just be greedy sampling. with a low $\tau = 0.1$ the output is [0.3,0.4,0.1] and the softmax is [0.34, 0.38. 0.28] which looks more like random sampling. So high τ boosts spikes, low τ looks like equal distribution
   The sentences generated are different per τ used. Using a small $\tau = 0.5$ produces gibberish "40 Tyad ye, Talas,admlch Bilbia". However we can still recognise some words it tried to write. using a $\tau = 1$ "And he dwellah, who was in the " is generated. These are words that where not seen in the greedy example. Probably because the randomness selects letters that

will initiate new words. With $\tau = 2$ still see new words, but the sentences still make sense, this seems like the perfect value "Egypt, and the sight of the sid".

2.1

When training my lstm on the bible the question arose what the bible would think of me, the result is "Mathieu he shall sing for himself in their cates, and they". I notice that because 'Mathieu' does not appear in the text the lstm doesn't know it's a name, so I had to search for a place where by accident it happens to threat it this way. The sentence turned out to be funny, and not have any of the words I have seen before.

3

3.1 Describe how this layer exploits the structural information in the graph data, and how a GCN layer can be seen as performing message passing over the graph

a) $H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)}$ This layer exploits structural information by encoding structural information in the A matrix. This matrix encodes the structural information and is multiplied with the previous layer. We can see the passing of information because the A only has values with where there are edges. This is multiplied with the previous layer, so only the information is passed, where there are edges.

b) There are drawbacks to GCNs. Mention one, and explain how we can overcome this.
The memory requirement can be to big for a GPU. If a graph doesn't fit on the gpu with full-batch gradient descent. However, it can still be trained on cpu or mini-batch GDS can be used

3.2

a) Give the adjacency matrix A for the graph as specified in Equation 18

$$\begin{bmatrix} A & B & C & D & E & F \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

b) Information is passed in the shortest route possible, this is 3 steps, [C-D-F-E]

3.3 take a look at the publicly available literature and name a few real-world applications in which GNNs could be applied.

Graph neural nets can be used in all fields where relational information needs to be encoded. Gives[1] a great overview of possible application. One of the examples is Text classification, or image classification.

3.4

a) Consider using RNN-based models to work with sequence representations, and GNNs to work with graph representations of some data. Discuss what the benefits are of choosing either one of the two approaches indifferent situations. For example, in what tasks (or datasets) would you see one of the two outperform the other? What representation is more expressive for what kind of data?

RNN works well with information that has a single connection, the sequence connection, while Graph-based NN can encode multiple connections. These two models can be used on the same dataset, but probably one will outperform the other depending on the task at hand. A GNN is great at

[1]https://arxiv.org/pdf/1812.08434.pdf

finding relationships between inputs. For example take molecule structures. These are very much relation based. With a RNN it's hard to predict the outcome of adding different molecules, but with GNN relations between parts of molecules can be combined. A RNN models predictions over time. While GNN is used for relations between objects

b) Think about how GNNs and RNNs models could be used in a combined model, and for what tasks this model could be used. Feel free to mention examples from literature to answer this question.

If one needs to get information of graphs over time, a GNN and RNN can be combined to make predictions about graphs over time.