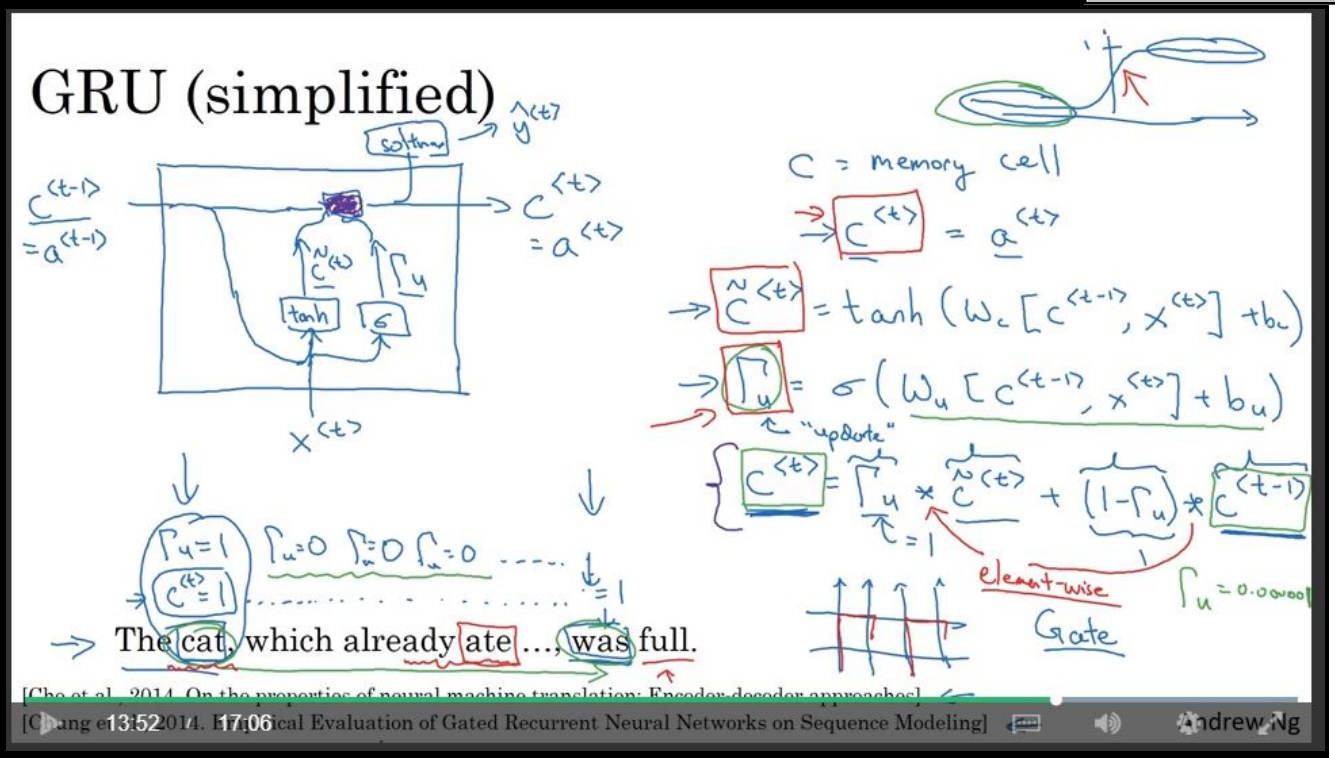

13:52 / 17:06

Gated Recurrent Unit (GRU)

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript

Downloads

Lecture Video mp4

Subtitles (English) WebVTT

0:00

You've seen how a basic RNN works. In this video, you learn about the Gated Recurrent Unit which is a modification to the RNN hidden layer that makes it much better capturing long range connections and helps a lot with the vanishing gradient problems. Let's take a look. You've already seen the formula for computing the activations at time t of RNN. It's the activation function applied to the parameter Wa times the activations in the previous time set, the current input and then plus ba. So I'm going to draw this as a picture. So the RNN unit, I'm going to draw as a picture, drawn as a box which inputs a of t-1, the activation for the last time-step. And also inputs x<t> and these two go together. And after some weights and after this type of linear calculation, if g is a tanh activation function, then after the tanh, it computes the output activation a. And the output activation a(t) might also be passed to say a softener unit or something that could then be used to output y<t>. So this is maybe a visualization of the RNN unit of the hidden layer of the RNN in terms of a picture. And I want to show you this picture because we're going to use a similar picture to explain the GRU or the Gated Recurrent Unit. Lots of the idea of GRU were due to these two papers respectively by Yu Young Chang, Kagawa, Gaza Hera, Chang Hung Chu and Jose Banjo. And I'm sometimes going to refer to this sentence which we'd seen in the last video to motivate that. Given a sentence like this, you might need to remember the cat was singular, to make sure you understand why that was rather than were. So the cat was for or the cats were for. So as we read in this sentence from left to right, the GRU unit is going to have a new variable called c, which stands for cell, for memory cell. And what the memory cell do is it will provide a bit of memory to remember, for example, whether cat was singular or plural, so that when it gets much further into the sentence it can still work under consideration whether the subject of the sentence was singular or plural. And so at time t the memory cell will have some value c of t. And what we see is that the GRU unit will actually output an activation value a of t that's equal to c of t. And for now I wanted to use different symbol c and a to denote the memory cell value and the output activation value, even though they are the same. I'm using this notation because when we talk about LSTMs, a little bit later, these will be two different values. But for now, for the GRU, c of t is equal to the output activation a of t. So these are the equations that govern the computations of a GRU unit. And every time-step, we're going to consider overwriting the memory cell with

a value c tilde of t. So this is going to be a candidate for replacing c of t. And we're going to compute this using an activation function tanh of Wc. And so that's the parameter to make sure it's Wc and we'll plus this parameter matrix, the previous value of the memory cell, the activation value as well as the current input value x<t>, and then plus the bias. So c tilde of t is going to be a candidate for replacing c<t>. And then the key, really the important idea of the GRU it will be that we have a gate. So the gate, I'm going to call gamma u. This is the capital Greek alphabet gamma subscript u, and u stands for update gate, and this will be a value between zero and one. And to develop your intuition about how GRUs work, think of gamma u, this gate value, as being always zero or one. Although in practice, your compute it with a sigmoid function applied to this. So remember that the sigmoid function looks like this. And so it's value is always between zero and one. And for most of the possible ranges of the input, the sigmoid function is either very, very close to zero or very, very close to one. So for intuition, think of gamma as being either zero or one most of the time. And this alphabet u stands for- I chose the alphabet gamma for this because if you look at a gate fence, looks a bit like this I guess, then there are a lot of gammas in this fence. So that's why gamma u, we're going to use to denote the gate. Also Greek alphabet G, right. G for gate. So G for gamma and G for gate. And then next, the key part of the GRU is this equation which is that we have come up with a candidate where we're thinking of updating c using c tilde, and then the gate will decide whether or not we actually update it. And so the way to think about it is maybe this memory cell c is going to be set to either zero or one depending on whether the word you are considering, really the subject of the sentence is singular or plural. So because it's singular, let's say that we set this to one. And if it was plural, maybe we would set this to zero, and then the GRU unit would memorize the value of the c<t> all the way until here, where this is still equal to one and so that tells it, oh, it's singular so use the choice was. And the job of the gate, of gamma u, is to decide when do you update these values. In particular, when you see the phrase, the cat, you know they you're talking about a new concept the especially subject of the sentence cat. So that would be a good time to update this bit and then maybe when you're done using it, the cat blah blah blah was full, then you know, okay, I don't need to memorize anymore, I can just forget that. So the specific equation we'll use for the GRU is the following. Which is that the actual value of c<t> will be equal to this gate times the candidate value plus one minus the gate times the old value, c<t>

minus one. So you notice that if the gate, if this update value, this equal to one, then it's saying set the new value of c<t> equal to this candidate value. So that's like over here, set gate equal to one so go ahead and update that bit. And then for all of these values in the middle, you should have the gate equals zero. So this is saying don't update it, don't update it, don't update it, just hang onto the old value. Because if gamma u is equal to zero, then this would be zero, and this would be one. And so it's just setting c<t> equal to the old value, even as you scan the sentence from left to right. So when the gate is equal to zero, we're saying don't update it, don't update it, just hang on to the value and don't forget what this value was. And so that way even when you get all the way down here, hopefully you've just been setting c<t> equals c<t> minus one all along. And it still memorizes, the cat was singular. So let me also draw a picture to denote the GRU unit. And by the way, when you look in online blog posts and textbooks and tutorials these types of pictures are quite popular for explaining GRUs as well as we'll see later, LSTM units. I personally find the equations easier to understand in a pictures. So if the picture doesn't make sense. Don't worry about it, but I'll just draw in case helps some of you. So a GRU unit inputs c<t> minus one, for the previous time-step and just happens to be equal to 80 minus one. So take that as input and then it also takes as input x<t>, then these two things get combined together. And with some appropriate weighting and some tanh, this gives you c tilde t which is a candidate for placing c<t>, and then with a different set of parameters and through a sigmoid activation function, this gives you gamma u, which is the update gate. And then finally, all of these things combine together through another operation. And I won't write out the formula, but this box here which wish I shaded in purple represents this equation which we had down there. So that's what this purple operation represents. And it takes as input the gate value, the candidate new value, or there is this gate value again and the old value for c<t>, right. So it takes as input this, this and this and together they generate the new value for the memory cell. And so that's c<t> equals a. And if you wish you could also use this process to soft max or something to make some prediction for y<t>. So that is the GRU unit or at least a slightly simplified version of it. And what is remarkably good at is through the gates deciding that when you're scanning the sentence from left to right say, that's a good time to update one particular memory cell and then to not change, not change it until you get to the point where you really need it to use this memory cell that is set even earlier in the sentence. And because the sigmoid value, now, because the gate

is quite easy to set to zero right. So long as this quantity is a large negative value, then up to numerical around off the uptake gate will be essentially zero. Very, very, very close to zero. So when that's the case, then this updated equation and subsetting c<t> equals c<t> minus one. And so this is very good at maintaining the value for the cell. And because gamma can be so close to zero, can be 0.000001 or even smaller than that, it doesn't suffer from much of a vanishing gradient problem. Because when you say gamma so close to zero this becomes essentially c<t> equals c<t> minus one and the value of c<t> is maintained pretty much exactly even across many many many many time-steps. So this can help significantly with the vanishing gradient problem and therefore allow a neural network to go on even very long range dependencies, such as a cat and was related even if they're separated by a lot of words in the middle. Now I just want to talk over some more details of how you implement this. In the equations I've written, c<t> can be a vector. So if you have 100 dimensional or hidden activation value then c<t> can be a 100 dimensional say. And so c tilde t would also be the same dimension, and gamma would also be the same dimension as the other things on drawing boxes. And in that case, these asterisks are actually element wise multiplication. So here if gamma u, if the gate is 100 dimensional vector, what it is really a 100 dimensional vector of bits, the value is mostly zero and one. That tells you of this 100 dimensional memory cell which are the bits you want to update. And, of course, in practice gamma won't be exactly zero or one. Sometimes it takes values in the middle as well but it is convenient for intuition to think of it as mostly taking on values that are exactly zero, pretty much exactly zero or pretty much exactly one. And what these element wise multiplications do is it just element wise tells the GRU unit which other bits in your- It just tells your GRU which are the dimensions of your memory cell vector to update at every time-step. So you can choose to keep some bits constant while updating other bits. So, for example, maybe you use one bit to remember the singular or plural cat and maybe use some other bits to realize that you're talking about food. And so because you're talk about eating and talk about food, then you'd expect to talk about whether the cat is four letter, right. You can use different bits and change only a subset of the bits every point in time. You now understand the most important ideas of the GRU. What I'm presenting in this slide is actually a slightly simplified GRU unit. Let me describe the full GRU unit. So to do that, let me copy the three main equations. This one, this one and this one to the next slide. So here they are. And for the full GRU unit, I'm sure to make

one change to this which is, for the first equation which was calculating the candidate new value for the memory cell, I'm going just to add one term. Let me pushed that a little bit to the right, and I'm going to add one more gate. So this is another gate gamma r. You can think of r as standing for relevance. So this gate gamma r tells you how relevant is c<t> minus one to computing the next candidate for c<t>. And this gate gamma r is computed pretty much as you'd expect with a new parameter matrix Wr, and then the same things as input x<t> plus br. So as you can imagine there are multiple ways to design these types of neural networks. And why do we have gamma r? Why not use a simpler version from the previous slides? So it turns out that over many years researchers have experimented with many, many different possible versions of how to design these units, to try to have longer range connections, to try to have more the longer range effects and also address vanishing gradient problems. And the GRU is one of the most commonly used versions that researchers have converged to and found as robust and useful for many different problems. If you wish you could try to invent new versions of these units if you want, but the GRU is a standard one, that's just common used. Although you can imagine that researchers have tried other versions that are similar but not exactly the same as what I'm writing down here as well. And the other common version is called an LSTM which stands for Long Short Term Memory which we'll talk about in the next video. But GRUs and LSTMs are two specific instantiations of this set of ideas that are most commonly used. Just one note on notation. I tried to define a consistent notation to make these ideas easier to understand. If you look at the academic literature, you sometimes see people- If you look at the academic literature sometimes you see people using alternative notation to be x tilde, u, r and h to refer to these quantities as well. But I try to use a more consistent notation between GRUs and LSTMs as well as using a more consistent notation gamma to refer to the gates, so hopefully make these ideas easier to understand. So that's it for the GRU, for the Gate Recurrent Unit. This is one of the ideas in RNN that has enabled them to become much better at capturing very long range dependencies has made RNN much more effective. Next, as I briefly mentioned, the other most commonly used variation of this class of idea is something called the LSTM unit, Long Short Term Memory unit. Let's take a look at that in the next video.