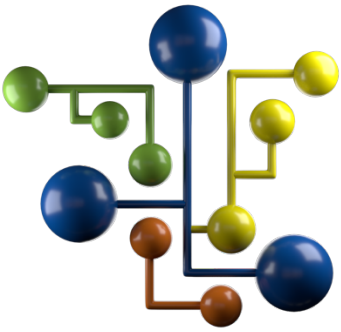


Data Science Academy

www.datascienceacademy.com.br

Processamento de Linguagem Natural

Gated Recurrent Units (GRUs)

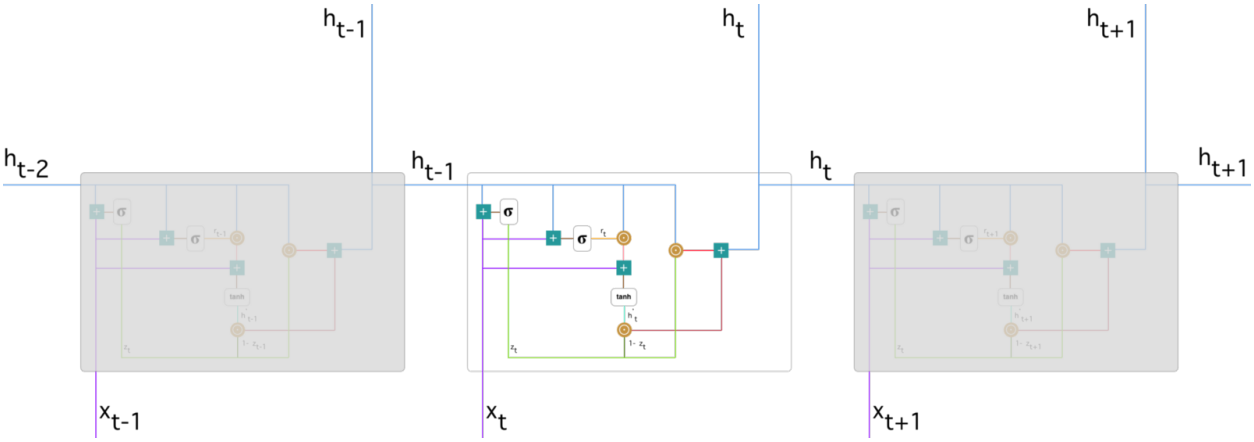
Vamos fornecer uma explicação bastante simples e compreensível de um tipo realmente fascinante de rede neural. Introduzido por Cho, et al. em 2014, o GRU (Gated Recurrent Unit), tem como objetivo resolver o problema da dissipação do gradiente, comum em uma rede neural recorrente padrão. O GRU também pode ser considerado uma variação do LSTM porque ambos são projetados de maneira semelhante e, em alguns casos, produzem resultados igualmente excelentes.

Como os Modelos GRUs funcionam?

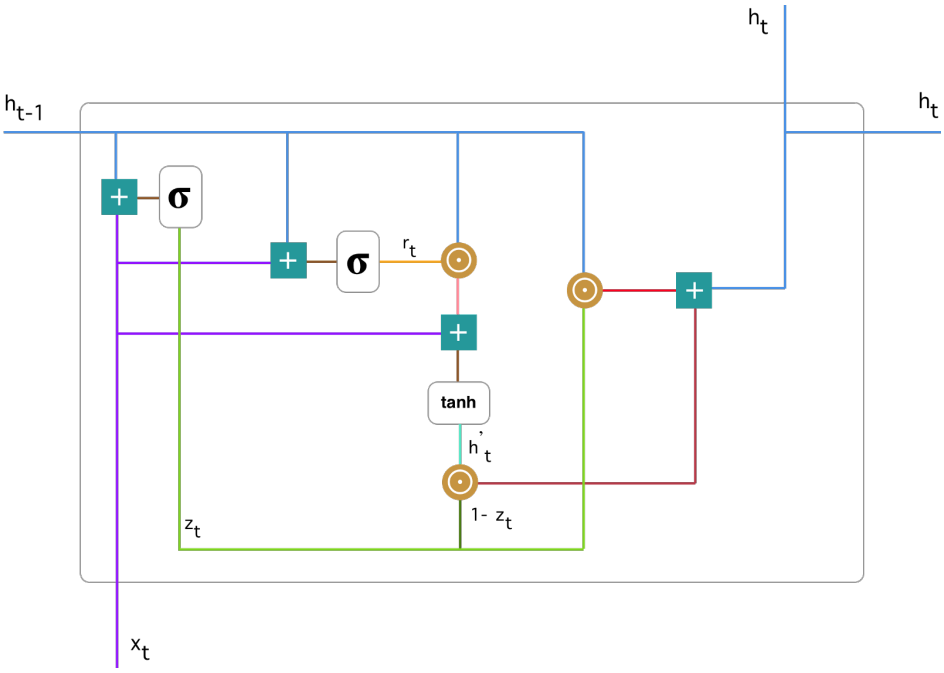
Como mencionado acima, os GRUs são uma versão melhorada da rede neural recorrente padrão. Mas o que os torna tão especiais e eficazes?

Para resolver o problema da dissipação do gradiente de um RNN padrão, o GRU usa, assim chamado, update gate e reset gate. Basicamente, estes são dois vetores que decidem quais informações devem ser passadas para a saída. O fato especial sobre eles é que eles podem ser treinados para manter as informações de muito tempo atrás, sem dissipar através do tempo ou remover informações irrelevantes para a previsão.

Para explicar a matemática por trás desse processo, examinaremos uma única unidade a partir da seguinte rede neural recorrente:



Aqui está uma versão mais detalhada do GRU único:



Primeiro, vamos apresentar as notações:

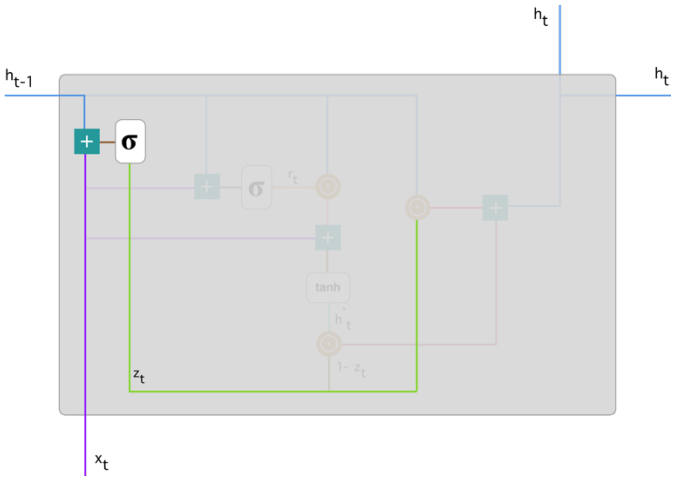
"plus" operation "sigmoid" function "Hadamard product" operation "tanh" function

1. Portão de Atualização (Update Gate)

Começamos com o cálculo do portão de atualização z_t para o passo de tempo t usando esta fórmula:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

Quando x_t é conectado à unidade de rede, ele é multiplicado pelo seu próprio peso W (z). O mesmo vale para h_ (t-1) que contém as informações para as unidades t-1 anteriores e é multiplicado pelo seu próprio peso U (z). Ambos os resultados são somados e uma função de ativação sigmóide é aplicada para *esmagar* o resultado entre 0 e 1. Seguindo o esquema acima, temos:



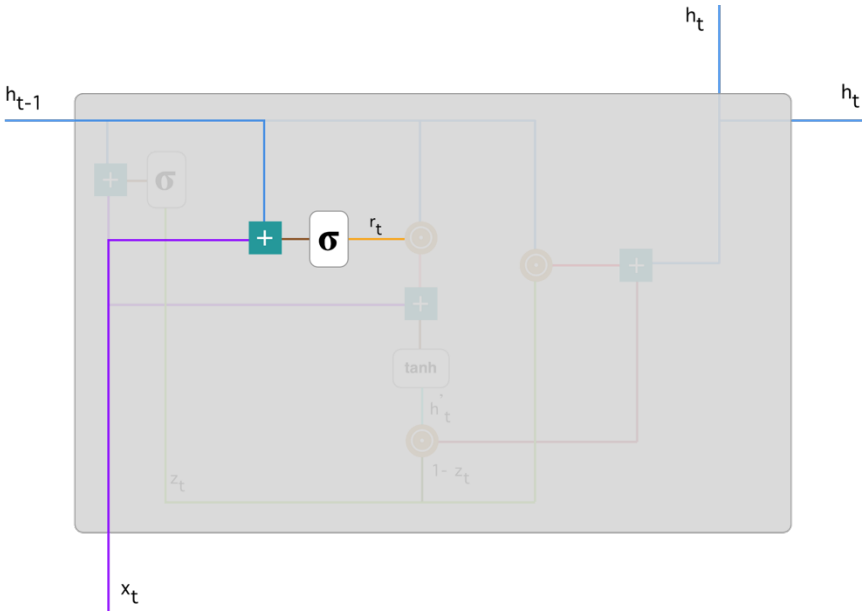
O gate de atualização ajuda o modelo a determinar quanto das informações anteriores (de etapas de tempo anteriores) precisa ser repassado para o futuro. Isso é realmente poderoso porque o modelo pode decidir copiar todas as informações do passado e eliminar o risco de desaparecer o gradiente. Vamos ver o uso do portão de atualização mais tarde. Por agora lembre-se da fórmula para z_t.

2. Portão de Reset (Reset Gate)

Essencialmente, esse gate é usado a partir do modelo para decidir quanto da informação passada deve ser esquecida. Para calcular, usamos:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

Essa fórmula é a mesma que a do gate de atualização. A diferença vem nos pesos e no uso do portão, que será visto daqui a pouco. O esquema abaixo mostra uma visão do portão de reset:



Como antes, conectamos h_ (t-1) - linha azul e x_t - linha púrpura, multiplique-os com seus pesos correspondentes, some os resultados e aplique a função sigmóide.

3. Conteúdo da Memória Atual

Vamos ver como exatamente os portões afetarão a saída final. Primeiro, começamos com o uso do portão de reset. Introduzimos um novo conteúdo de memória que usará o gate de reset para armazenar as informações relevantes do passado. É calculado da seguinte forma:

$$h_t^{'} = \tanh(Wx_t + r_t \odot Uh_{t-1})$$

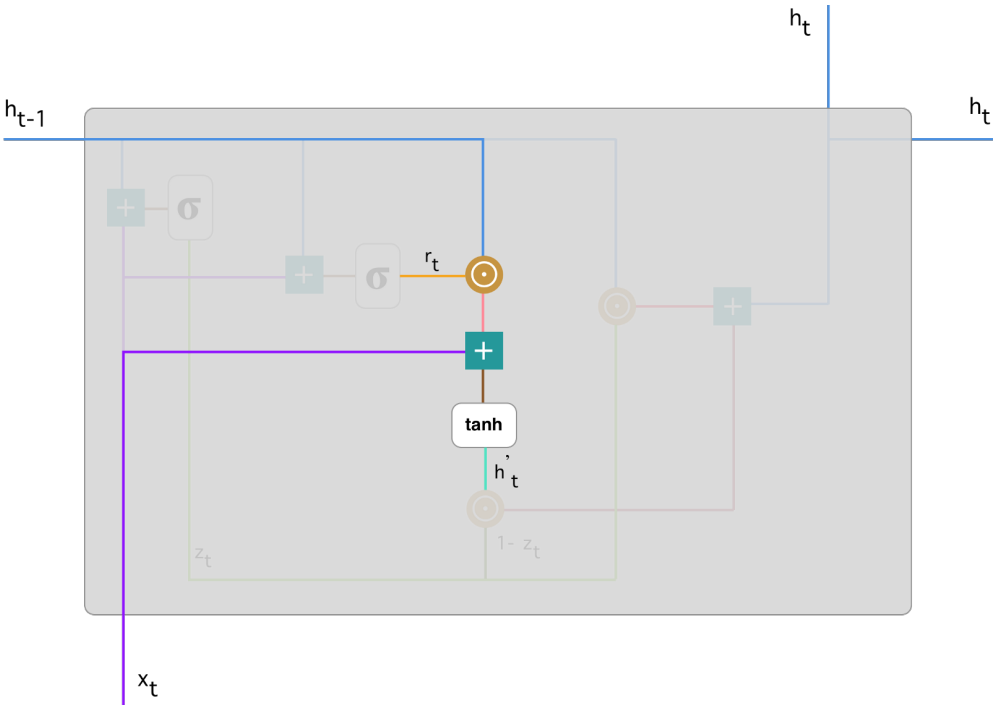
A. Multiplique a entrada x_t com um peso W e h_ (t-1) com um peso U.

B. Calcule o produto element-wise entre as portas de reset r_t e Uh_ (t-1). Isso determinará o que remover das etapas de tempo anteriores. Digamos que tenhamos um problema de análise de sentimentos para determinar a opinião de um livro com base em um comentário escrito por um leitor. O texto começa com “Este é um livro de fantasia que ilustra…” e depois de alguns parágrafos termina com “Eu não gostei muito do livro porque acho que ele captura muitos detalhes.” Para determinar o nível geral de satisfação do livro só precisamos da última parte da revisão. Nesse caso, à medida que a rede neural se aproxima do final do texto, ela aprenderá a atribuir um vetor r_t próximo de 0, eliminando o passado e concentrando-se apenas nas últimas sentenças.

C. Resumir os resultados dos passos 1 e 2.

D. Aplique a função de ativação não linear tanh.

Você pode ver claramente as etapas aqui:



Fazemos uma multiplicação de elementos de h_ (t-1) - linha azul e r_t - linha laranja e, em seguida, somamos o resultado - linha rosa com a entrada x_t - linha roxa. Finalmente, tanh é usado para produzir h'_t - linha verde brilhante.

4. Memória Final no Momento Atual

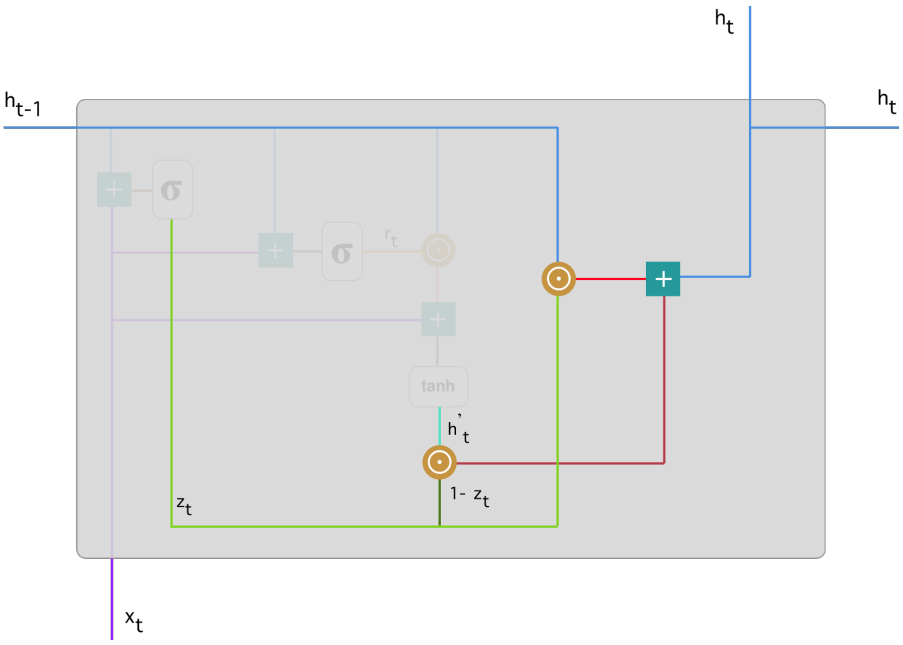
Como último passo, a rede precisa calcular h_t - vector que contém informações para a unidade atual e as transfere para a rede. Para fazer isso, o portão de atualização é necessário. Ele determina o que coletar do conteúdo atual da memória - h'_t e os passos anteriores - h_ (t-1). Isso é feito da seguinte maneira:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t^{'}$$

A. Aplique a multiplicação por elementos aos portões de atualização z_t e h_ (t-1).
B. Aplique multiplicação por elementos a (1-z) e h_t.
C. Soma os resultados dos passos 1 e 2.

Vamos trazer o exemplo sobre a resenha do livro. Desta vez, a informação mais relevante é posicionada no início do texto. O modelo pode aprender a definir o vetor z_t próximo de 1 e manter a maioria das informações anteriores. Como z_t será próximo de 1 neste intervalo de tempo, 1-z_t será próximo de 0, o que ignorará grande parte do conteúdo atual (neste caso, a última parte da revisão que explica o gráfico do livro) que é irrelevante para a nosso predição.

Aqui está uma ilustração que enfatiza a equação acima:



A seguir, você pode ver como a linha z_t - verde é usada para calcular 1-z_t que, combinada com h'_t - linha verde brilhante, produz um resultado na linha vermelha escura. z_t também é usado com h_ (t-1) - linha azul em uma multiplicação por elementos. Finalmente, h_t - linha azul é um resultado da soma das saídas correspondentes às linhas vermelhas brilhantes e escuras.

Agora, você pode ver como as GRUs podem armazenar e filtrar informações usando suas portas de atualização e reset. Isso elimina o problema da dissipação do gradiente, já que o modelo não está dissipando a nova entrada toda vez, mas mantém as informações relevantes e as transfere para as próximas etapas de tempo da rede. Se cuidadosamente treinados, eles podem executar extremamente bem até mesmo em cenários complexos.

Esse modelo de aprendizagem profunda de última geração é chamado GRU.

Referências:

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation
https://arxiv.org/pdf/1406.1078v3.pdf