

Redes Neurais Recorrentes

Lidando com todo tipo de sequência

Posted on 12/09/2017

Neste tutorial não vamos implementar nada e nenhum código será utilizado. Vou apenas explicar o que são as Redes Neurais Recorrentes de uma maneira intuitiva e matemática. Há várias formas de utilizar RNRs, o que significa que nenhum exemplo pode ser geral o suficiente para abarcar todos os casos. Assim, prefiro primeiro cobrir os aspectos teóricos desse modelo e depois dedicar tutoriais de programação lidando com as diversas aplicações.

Pré-requisitos

Vou pressupor que você tenha os conhecimentos especificados no tutorial sobre matemática e programação para aprendizado de máquina (<https://matheusfacure.github.io/2017/01/15/pre-req-ml/>), isto é, que sabe cálculo (derivadas), o básico de álgebra linear, de estatística e de programação. Eu também vou pressupor que você viu os tutoriais anteriores a esse. Meus tutoriais (<https://matheusfacure.github.io/tutorials/>) são ordenados de maneira lógica e sugiro fortemente que você se atenha à ordem deles para maior compreensão.

Conteúdo

1. Motivação
2. Intuição
3. Aplicações
4. Matemática
5. Referências

Motivação

Eu aposto que você consegue recitar o alfabeto sem maiores problemas. Agora tente recitá-lo de trás para frente e você verá que esta última tarefa é bem mais difícil. Vamos tentar outro experimento. Tente ler a seguinte frase:

Miséria nossa de legado o criatura nenhuma a transmiti não, filhos tive não.

Ela não faz muito sentido, mas se a colocarmos na ordem certa você facilmente reconhecerá o fim de Memórias Póstumas de Brás Cubas:

Não tive filhos, não transmiti a nenhuma criatura o legado de nossa miséria.

Os dois experimentos acima mostram que algumas informações se fazem presente não apenas pelo conteúdo que as compõe, mas também pela maneira como esse conteúdo está disposto. Em termos mais simples, podemos dizer que existem informações que são sequenciais por natureza. Palavras, frases e enredos são os exemplos mais típicos desse tipo de informação, mas não precisamos nos limitar a isso. Quando você vê uma cena em movimento seu cérebro antecipa o que vai acontecer, de forma que uma pausa ou um movimento revertido pode causar estranheza. De fato, estamos sempre prevendo o futuro, seja projetando a trajetória de um objeto que é jogado em sua direção, seja antecipando o sabor do almoço pelo aroma que vem da cozinha.

Dados sequenciais também são comuns em outros campos que não são tão relacionados com a percepção humana. Um exemplo são séries de tempo, como o preço de ações, a taxa de juros ou as exportações de um país. Outros exemplos são dados de transação bancária por cliente, características das escolas registradas anualmente no censo escolar (<http://censobasico.inep.gov.br/censobasico/#/>) ou compras por produto registradas em um supermercado diariamente.

A forma mais tradicional de lidar com esse tipo de dado é com **variáveis defasadas**. Por exemplo, se você tem dados diários de demanda de sorvete e temperatura, pode ser uma boa ideia usar essas duas variáveis defasadas 2 dias para tentar prever a demanda de sorvete no dia seguinte. Nesse caso a variável dependente seria $y = demanda_{t+1}$ e as variáveis independentes seriam $x = [demanda_t, demanda_{t-1}, temp_t, temp_{t-1}]$. No entanto, há alguns problemas com essa representação. Por exemplo, como fazer para decidir quantas defasagens é suficiente? Uma vez decidido isso, o que fazer quando algumas observações não têm nem sequer o número mínimo de defasagens? E o que fazer quando o tamanho das sequências é variável? Além disso, se cada observação no tempo tiver muitas variáveis, defasá-las gerará um vetor x tão grande que talvez não caiba na memória!

Não seria ótimo então um modelo que pudesse operar em sequências independentemente do seu tamanho e das suas variações? Melhor ainda seria se esse modelo não aumentasse de tamanho com o aumento das sequências. É para cumprir essas necessidades que podemos utilizar **Redes Neurais Recorrentes**. RNRs são Turing Completas

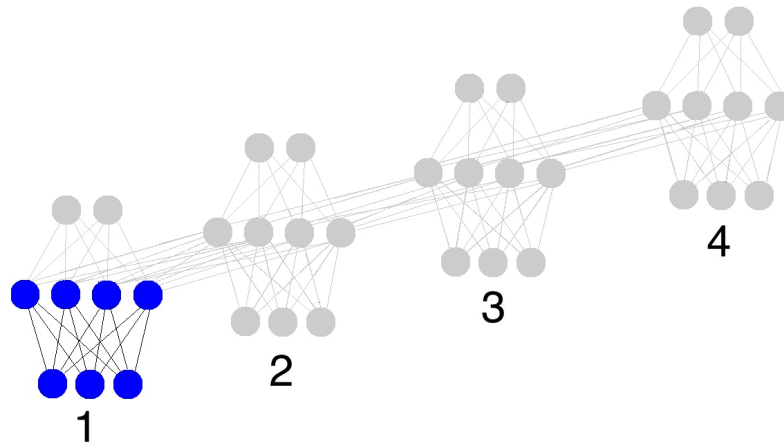
(https://pt.wikipedia.org/wiki/Turing_completude), o que significa que seu poder é tão grande que

elas têm a capacidade teórica de representar qualquer programa passível de ser feito em um computador. Esse tipo de modelo já existe há um bom tempo, mas só muito recentemente conseguimos alavancar seu enorme potencial para finalidades práticas.

Já **aviso** que esse é um dos modelos que mais tive dificuldade em entender, então não se assuste caso não o entenda de primeira. Se te parecer um conteúdo meio nebuloso, no final do post darei algumas dicas que podem lhe ajudar a aprender. Por enquanto, vem comigo!

Intuição

Considere uma rede neural com apenas uma camada oculta. Nesse caso, há apenas uma diferença fundamental entre as redes neurais recorrentes e as redes neurais clássicas que vimos até agora (<https://matheusfacure.github.io/2017/05/15/deep-ff-ann/>): a camada oculta das redes neurais *feedforward* totalmente conectada recebe sinais apenas da camada de entrada, isto é, dos dados brutos, processa esses sinais e os passa à camada de saída; a camada oculta da rede neural recorrente, por sua vez, recebe sinais tanto da camada de entrada quanto **da camada oculta na iteração de tempo anterior**. Em certo sentido, é como se a camada oculta da rede neural recorrente funcionasse como uma memória: a cada período no tempo, a rede neural não só armazena no seu estado oculto informações dos dados observados **naquele período no tempo**, como também recupera informações do estado oculto anterior que, num cenário ótimo, armazenou toda a informação relevante que aconteceu no passado. Mais ainda, o estado oculto **nesse mesmo período de tempo** pode fornecer informações para a camada de saída, caso seja o momento de realizar uma previsão, e também passa informações para o **estado oculto do próximo período de tempo**. De maneira mais resumida, o estado oculto da RNR recebe informações das variáveis independentes X_t , assim como de seu próprio resultado de processamento H_{t-1} , só que de um período de tempo atrás. Note que **período de tempo** aqui não representa o tempo do relógio, mas um tempo lógico que denota **ordem** numa sequência. Por exemplo, cada palavra em uma frase denota um período de tempo de processamento da RNR. Talvez seja melhor mostrar do que explicar:



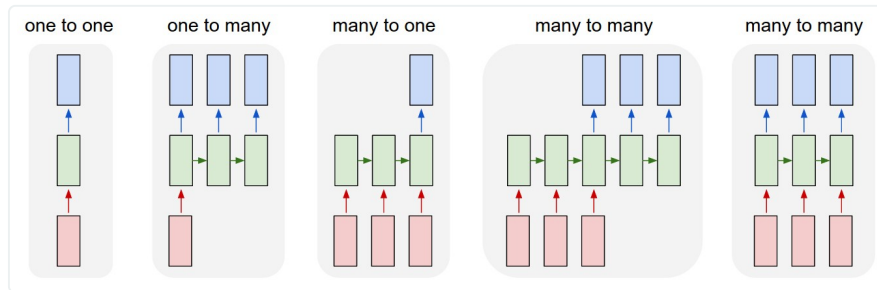
Processo de recorrência na rede neural, maravilhosamente explicado neste post por Trask (<https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/>)

No GIF acima vemos o processamento que ocorre numa rede neural recorrente em quatro períodos de tempo. No primeiro, o estado oculto é puramente definido pelos dados, isto é, pela camada de entrada. Já o segundo estado, é uma mistura do primeiro estado oculto e dos dados observados no segundo período de tempo. O terceiro estado oculto contém uma mistura dos primeiro e segundo, além de conter informações dos dados observados no terceiro período de tempo. Finalmente, o quarto estado oculto contém um pouco da informação do que aconteceu em todos os períodos de tempo.

É importante notar que, na prática, a RNR escolhe o que armazenar e o que esquecer. Nos segundo e terceiro períodos de tempo, a RNR representada acima julgou que era importante usar metade do seu estado oculto para armazenar o que foi observado naqueles momentos. Já no quarto período, a RNR julgou que só um pouco da informação capturada valia a pena armazenar, utilizando apenas um quarto do seu estado oculto para isso.

Aplicações

Você deve ter notado que o GIF acima não está mostrando informação saindo dos estados ocultos. Isso porque a forma como passamos a informação do estado oculto adiante pode variar drasticamente. Redes neurais recorrentes foram feitas para processar sequências, mas tanto a forma como essas sequências são lidas para a rede, quanto a forma como são geradas por ela, são ambas escolhas de arquitetura que dependerão do tipo de aplicação do modelo. Para entender melhor sobre isso, considere a imagem abaixo.



Fonte: *The Unreasonable Effectiveness of Recurrent Neural Networks*
 (<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>).

No caso **one to one** temos a clássica rede neural *feedforward*. Em aplicações **one to many**, lemos os dados uma única vez para o estado oculto e então passamos esse estado oculto adiante por vários períodos de tempo, nos quais também lemos informações contidas nele. Um exemplo de aplicação desse tipo é quando queremos criar um modelo para descrever imagens (*image-captioning* (<https://research.googleblog.com/2016/09/show-and-tell-image-captioning-open.html>)). Nesse caso, lemos a imagem uma vez para o estado oculto, mas lemos desse estado oculto várias vezes, uma vez para cada palavra gerada na descrição da imagem.

No caso **many to one**, lemos dos dados várias vezes, mas produzimos uma previsão apenas após ler toda a sequência. Um bom exemplo desse tipo de aplicação é classificação de texto, como análise de sentimentos ([/2017/05/26/deep-nlp-sentiment/](https://medium.com/@matheusf/2017/05/26/deep-nlp-sentiment/)), onde lemos o texto todo antes de realizar uma previsão.

Por fim, existem dois casos de aplicações **many to many**. Na primeira delas, nós lemos os dados por alguns períodos de tempo antes de começar a soltar previsões. Em outras palavras, há uma defasagem entre a leitura da sequência de entrada e a produção da sequência de saída. Um exemplo de aplicações desse tipo são casos de tradução por máquinas (https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/recurrent_neural_networks/machine-translation-using-rnn.html) ou transcrição de áudio, onde a RNR lê a sequência por algum tempo antes de começar a gerar a sequência de saída, seja ela as palavras em outra língua ou as palavras correspondentes a um áudio. O último caso de **many to many** é quando temos sequências na entrada e na saída da rede, mas cada entrada corresponde a uma saída no mesmo período de tempo. Esse tipo de estrutura de dados aparece em séries de tempo e dados em painel, nas quais queremos realizar uma previsão para o período de tempo seguinte, dado o que ocorreu neste período de tempo e nos períodos anteriores.

Matemática

Se você prefere entender modelos com fórmulas em vez de explicações demoradas, aqui vão algumas equações que resumem tudo o que eu disse acima.

$$\mathbf{h}_t = \phi(\mathbf{b}_h + \mathbf{x}\mathbf{W}_x + \mathbf{h}_{t-1}\mathbf{W}_h)$$

$$\hat{\mathbf{y}}_t = \mathbf{b}_o + \mathbf{h}_t\mathbf{W}_o$$

Note que a diferença fundamental entre as equações acima e as das redes neurais clássicas é que adicionamos informação do estado oculto do período anterior: $\mathbf{h}_{t-1}\mathbf{W}_h$. Mais ainda, Repare que os parâmetros que fazem a transição da informação entre os estados ocultos de diferentes períodos são **sempre os mesmos**. Isso mostra que redes neurais recorrentes **compartilham parâmetros através do tempo**. Na prática, definimos a recorrência acima como um **loop** no código, o que significa que, novamente, o modelo não passa de algumas multiplicações de matrizes, seguidas de funções de ativação. O que também significa que não há nenhum mistério na forma como ocorre a otimização por GDE. Em alguns lugares, você pode encontrar referências ao nome *Backpropagation Through Time* para treinar RNRs, mas isso nada mais é do que o algoritmo de *Backpropagation* (/2017/03/10/backprop/) que já vimos.

Se ainda não acredita, tente escrever as equações da RNR de forma desenrolada no tempo. Por exemplo, vamos considerar uma RNR com uma camada oculta, processando quatro períodos de tempo e realizando uma previsão de uma variável contínua (problema de regressão) apenas após observar os 4 períodos (caso *many to one*):

$$\mathbf{H}_0 = \phi(\mathbf{b}_h + \mathbf{X}\mathbf{W}_x)$$

$$\mathbf{H}_1 = \phi(\mathbf{b}_h + \mathbf{X}\mathbf{W}_x + \mathbf{H}_0\mathbf{W}_h)$$

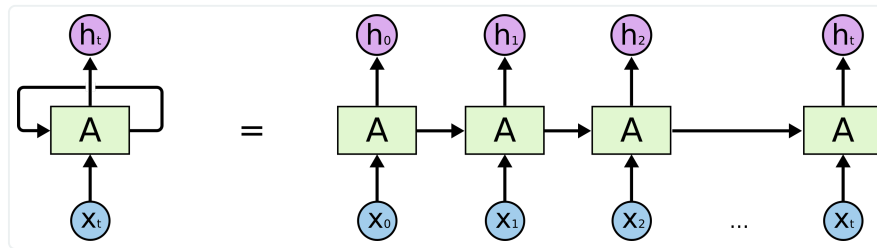
$$\mathbf{H}_2 = \phi(\mathbf{b}_h + \mathbf{X}\mathbf{W}_x + \mathbf{H}_1\mathbf{W}_h)$$

$$\mathbf{H}_3 = \phi(\mathbf{b}_h + \mathbf{X}\mathbf{W}_x + \mathbf{H}_0\mathbf{W}_h)$$

$$\hat{\mathbf{y}} = \mathbf{b}_o + \mathbf{H}_3\mathbf{W}_o$$

$$\mathcal{L} = \frac{1}{n} \sum_{i=0}^n (\hat{y} - y)^2$$

O gradiente então pode ser calculado partindo da função custo, indo de trás para frente. Lembre-se de que não precisamos nos preocupar com isso, já que os *frameworks* de *Deep Learning*, como o TensorFlow, tomarão conta disso para nós computando as derivadas automaticamente. Também é importante notar que, muitas vezes, você verá RNR representadas como uma conexão recorrente, saindo do estado oculto e apontando de volta para ele, como esquerda da imagem abaixo. Essa representação é enganosa, pois dá a impressão que o estado oculto não muda. Assim, peço que você sempre **pense em RNR como uma rede neural muito profunda, que se desenrola no tempo**, como na direita da imagem abaixo.



Fonte: *Understanding LSTM Networks* (<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>).

Referências

No momento em que escrevo, Redes Neurais Recorrentes são um dos tópicos mais efervescentes de Inteligência Artificial e Aprendizado de Máquina. Assim, é de se esperar que exista muito conteúdo explicando como elas funcionam. Caso você ainda tenha dúvidas depois de ler meu post, sugiro procurar informações sobre esse mesmo assunto em fontes diversas, assim poderá pegar de um autor o que outro deixou passar em sua explicação. Apenas tente evitar posts que misturem Redes Neurais Recorrentes com LSTM (*Long Short-Term Memory*). Essas últimas são uma complicação a parte que ainda não vimos. Mas, caso esteja muito muito curioso, sugiro este post (<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>) do Christopher Olah.

O melhor lugar para aprender sobre Redes Neurais Recorrentes é a aula 3, *Language Modelling and RNRs Part 1*, (<https://github.com/oxford-cs-deepnlp-2017/lectures>) do curso *Deep Learning for Natural Language Processing*, da universidade de Oxford.

O segundo melhor lugar para aprender sobre RNRs são os vídeos (<https://www.youtube.com/playlist?list=PLnnr1O8OWc6YM16tj9pdhBZOS9tDktNrx>) de Geoffrey Hinton, para o Coursera. Assista os primeiros 4 desta *playlist*, mas não se preocupe ainda em entender LSTM.

Se você prefere aprender com livros, sugiro este capítulo (<http://www.deeplearningbook.org/contents/rnn.html>) do livro *Deep Learning* (Goodfellow et al, 2016).

Se você prefere aprender com blogs, este post (<http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>), por Denny Britz, fornece uma apresentação bastante intuitiva e completa sobre RNRs. O post *The Unreasonable Effectiveness of Recurrent Neural Networks* (<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>), por Andrej Karpathy, é talvez o post mais famoso da internet sobre Redes Neurais Recorrentes. Eu recomendo muito que o leia para ver um pouco do enorme poder desse tipo de modelo.

TAMBÉM EM MATHEUSFACURE

TensorFlow Detalhado

4 anos atrás • 3 comentários

Explorando Aprendizado de Máquina nas Ciências Humanas e Sociais

Matemática e Programação para ...

4 anos atrás • 11 comentários

Explorando Aprendizado de Máquina nas Ciências Humanas e Sociais

RNRs para Séries de Tempo

3 anos atrás • 10 comentários

Explorando Aprendizado de Máquina nas Ciências Humanas e Sociais

Análise Sentim

4 anos at

Explorar Máquina Humana

2 Comentários

matheusfacure

Disqus' Privacy Policy

1 Entrar

Recomendar 2

Tweet

Compartilhar


Ordenar por Mais votados

Participe da discussão...

FAZER LOGIN COM

OU REGISTRE-SE NO DISQUS


Nome



Giovani Marin • 3 anos atrás

Muito bom, obrigado por compartilhar seu conhecimento.

Responder • Compartilhar



Luiz Andrade • 3 anos atrás

Olá Matheus,

Parabéns pelo post. Muito legal. Apenas um comentário, acho que a expressão de H3 no exemplo contém um pequeno engano. O índice do estado no lado direito da equação deveria ser relativo a H2, não?

Abs,

Responder • Compartilhar

Inscreva-se Adicione o Disqus no seu siteAdicionar DisqusAdicionar Do Not Sell My Data

in
(http://linkedin.com/in/matheus-
facure-
7b0099117)(https://www.instagram.com/matheusfacure)01@gmail.com