08

Para saber mais

Keras, Tensorflow e outras bibliotecas de Deep Learning

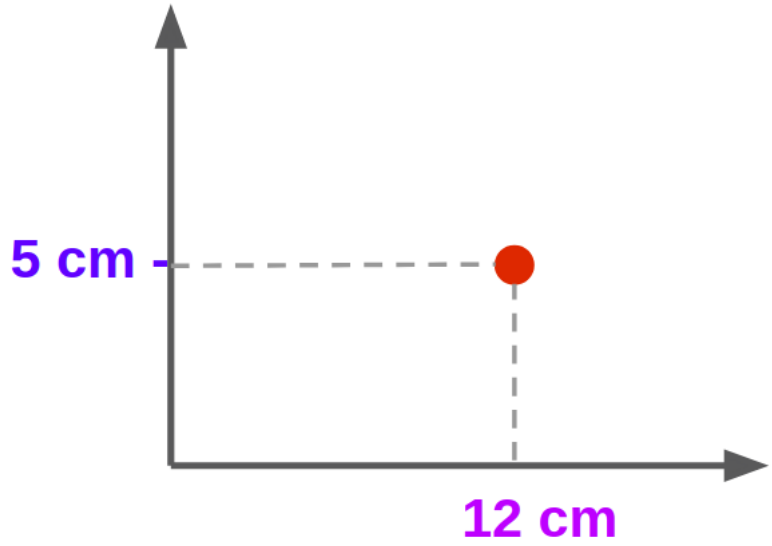
Estamos escrevendo a maior parte do nosso código usando uma API de alto nível chamada Keras (https://keras.io/).

Mas o Keras não roda direto em cima do Tensorflow (https://www.tensorflow.org/), ele foi construído em cima de uma outra biblioteca chamada Theano (http://www.deeplearning.net/software/theano/), como cita o seu criador (https://www.quora.com/What-are-some-important-engineering-and-design-decisions-you-made-in-creating-Keras) nesta resposta do Quora (site em inglês).

Dada a agilidade que o Keras proporciona para gerarmos e testarmos modelos, ele foi integrado ao Tensorflow, por isso que ao importarmos o Tensorflow, importamos também o Keras.

Mas, se o Keras está acima do Tensorflow, porque não usamos o Tensorflow direto? Um aspecto da resposta é porque a maioria das implementações de projetos de Deep Learning se baseia em duas coisas: vetores e tensores.

Se eu te pedir para me dizer quais as distâncias que juntas mostram onde o ponto vermelho está nesse gráfico,

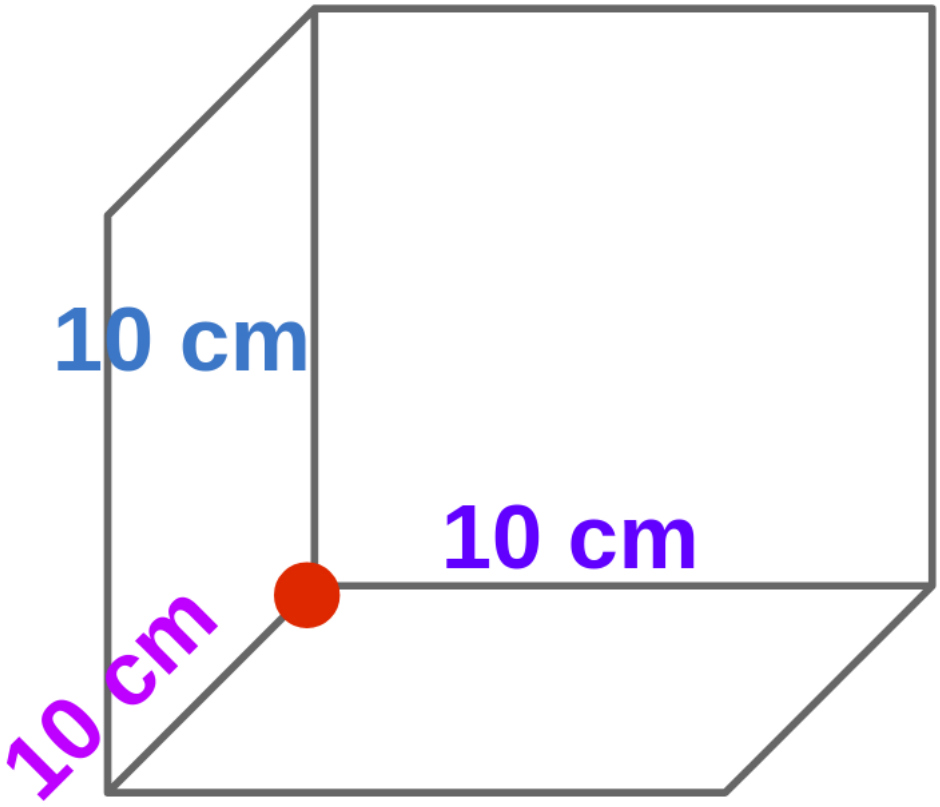


você irá me dizer 5 cm, 12 cm, certo? E aí podemos agrupar esses números para facilitar, como

(5, 12)

Esse agrupamento que descreve onde o ponto está, ou as suas coordenadas, é um vetor. E sim, o conceito de vetor que temos aqui é um conceito um pouco diferente daquele que aprendemos quando estudamos as disciplinas de Física ou Matemática.

E, se olharmos para esse vetor (5, 12), temos 2 números que descrevem um ponto, mas podíamos ter 3 números, como nesse vértice do cubo.



E aí para descrevê-lo teríamos

$$vetor = (10, 10, 10)$$

E esse vetor que usamos para descrever o ponto do cubo pode ser chamado também de tensor.

Então tensor e vetor são a mesma coisa? Não, tensor é uma generalização que serve também para vetor, então todo vetor é um tensor, mas nem sempre podemos chamar um tensor de vetor.

Repare que no gráfico temos 2 coordenadas para descrever o ponto, então temos 2 eixos e duas dimensões. Já, quando descrevemos o vértice do cubo temos 3 números, então 3 eixos e 3 dimensões.

Um vetor vai ter duas dimensões, mas um tensor pode ter quantas dimensões quisermos.

Agora, imagine declarar e configurar um tensor de 70.000 dimensões. Começa a ficar complicado. E imagine implementar cada algoritmo que vai rodar nesse tensor, vai levar um pouco mais de tempo. Estamos extrapolando nesse exemplo, muitas dessas coisas são facilitadas pelo próprio Tensorflow, mas podemos usar o Keras para abstrairmos e só dizermos o que a gente quer que o nosso modelo faça, deixando o Tensorflow se virar para fazer isso lá no baixo nível.

Mas, não há apenas o Keras como API de Deep Learning, há também muitas outras como:

- Caffe (http://caffe.berkeleyvision.org/) do grupo de pesquisa da universidade de Berkeley,
- Torch (http://torch.ch/) uma implementação usando Lua,
- PyTorch (https://pytorch.org/) uma variação do Torch para Python,
- CNTK (https://www.microsoft.com/en-us/cognitive-toolkit/) da Microsoft,

- DL4J (https://deeplearning4j.org/) uma implementação usando Java,
- MXNet (https://mxnet.incubator.apache.org/) da Apache.

Cada uma delas tem as suas particularidades, é sempre legal experimentar e ver aquela com a qual se sente mais confortável.

Além disso, tudo depende do objetivo do projeto. Se está desenvolvendo e testando um modelo, prototipando, é interessante usar uma API de alto nível para ganhar velocidade e descer de nível ao ter algo mais concreto para implementar à medida que for precisando obter mais controle sobre o modelo.

Lembrando que, quanto mais a API for próxima da linguagem natural, menos controle você vai ter do que acontece por debaixo dos panos, e quanto mais próxima ela for do baixo nível ou dos tensores (um teste pode ser fazer uma implementação usando Tensorflow puro) mais controle você vai ter do modelo que for desenvolver.