LEVANDO IA PARA PRODUÇÃO

UNIDADE II
FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

Elaboração

Paulo Vitor Pereira Cotta

Natasha Sophie Pereira

Produção

Equipe Técnica de Avaliação, Revisão Linguística e Editoração

SUMÁRIO

UNIDADE II

FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL 5

CAPÍTULO 1

FRAMEWORKS 5

REFERÊNCIAS 10

CAPÍTULO 1

FRAMEWORKS

O que são *frameworks*?

Basicamente, é um conjunto de funcionalidades que podem ser usadas pelo desenvolvedor. Com os *frameworks*, é desnecessário gastar tempo para reproduzir a mesma função em diferentes projetos, o que facilita na construção, auxiliando em um gerenciamento ágil de projetos. De acordo com seu conjunto, ele é uma estrutura base, uma plataforma de desenvolvimento, como uma espécie de conjunto de ferramentas que contém um conjunto de funcionalidades, guias, sistemas e componentes que agilizam o processo de desenvolvimento de soluções, auxiliando os desenvolvedores em seus trabalhos.

Eles contêm um conjunto de bibliotecas que permitem aos desenvolvedores trabalhar sobre elas para operações maiores. É o responsável por ajudar na solução criada, por assim dizer. Assim, para que seu aplicativo ou sistema *Web* faça com que a solução dê certo, é necessário escolher um bom *framework*, principalmente por gerar todo o fluxo de controle da aplicação e facilitadores para os desenvolvedores. Isso também é importante, pois os *frameworks* fazem com que você não tenha que se preocupar em ficar reescrevendo códigos, mas tenha que aprender a utilizá-lo, focando na resolução de problemas, ou seja, direcionando seus esforços para o objetivo final, agilizando o processo de construção.

Uma boa comparação que demonstra como é um *framework* é a da caixa de ferramentas. Porém, em vez de ferramentas como o martelo, tem-se funções prontas, bases para formulários de *login*, validação de campos e conexão com bancos de dados. Nesse sentido, como parte da tendência de buscar reduzir custos e aumentar a produtividade da equipe técnica, o uso de *frameworks* tem se tornado cada vez mais popular. Hoje, já existe uma grande variedade de *frameworks* disponíveis para as mais diversas linguagens, com comunidades que testam e criam diferentes funções.

TensorFlow

O *Tensorflow* é uma biblioteca *open source*, ou seja, de código aberto, utilizada em *Machine Learning* com possibilidade de aplicação a uma grande variedade de tarefas. É uma biblioteca para desenvolvimento e treinamento de Redes Neurais Artificiais voltadas para a detecção e reconhecimento de padrões e correlações. O *Tensorflow* busca realizar as tarefas de modo análogo ao comportamento do cérebro humano, ou seja, permitem aos sistemas computacionais aprender e raciocinar como um humano. De acordo com pesquisadores do Google Inc., essa ferramenta é utilizada tanto na pesquisa quanto para produção na Google e aos poucos vem, inclusive, substituindo o código proprietário da empresa, chamado DistBelief. O *Tensorflow* foi criado pela equipe Google Brain com finalidade de uso interno pela empresa, porém, em novembro de 2015, foi lançado para uso pela comunidade de desenvolvedores, sob a licença de código aberto Apache 2.0.

Hoje, uma utilização muito comum para o *Tensorflow* é o treinamento e execução de Redes Neurais Artificiais Profundas, também conhecidas como *Deep Learning*. Nesse seguimento, a biblioteca é amplamente utilizada na classificação de dígitos manuscritos, classificação e reconhecimento de imagens, incorporação de palavras em texto, montagem de Redes Neurais Recorrentes, modelos para tradução automática de texto, processamento de linguagem natural, tanto falada quanto escrita, entre outras diversas aplicações.

O *TensorFlow* permite aos desenvolvedores uma infinidade de possibilidades para trabalhar os dados, desde a utilização de grafos de fluxo de dados; também permite trabalhar com tensores, vetores, matrizes e estruturas de dados que se movem por um grafo (sequência de nós) de processamento, de modo que cada nó do grafo representa uma operação matemática, e cada conexão (aresta) entre os nós é uma matriz de dados ou um tensor multidimensional.

O *TensorFlow* pode ser executado na maioria das plataformas: uma máquina local, um *cluster* na nuvem, dispositivos iOS e Android, CPUs ou GPUs, via Javascript em um *site web*. Se você usa a própria nuvem do Google, é possível executar o *TensorFlow* no silício, que é um átomo de energia da GPU, personalizado da unidade de processamento *TensorFlow* (TPU) do Google para acelerar ainda mais a execução dos projetos de cálculos matemáticos usando grafos. Os modelos criados a partir de aplicações que se utilizam da biblioteca *TensorFlow*, ou seja, os resultados das aplicações que utilizam essa tecnologia, podem ser colocados para produção em grande parte dos dispositivos utilizados para exibir previsões.

O *TensorFlow* traz muitos benefícios para o desenvolvedor e facilidade para a criação de redes neurais, sendo que o maior benefício que oferece para o desenvolvimento

de ML é a abstração e o processo de cálculos. Em vez de lidar com os detalhes da implementação de algoritmos, ou de descobrir formas adequadas de entradas e a saída de uma função à entrada de outra, o desenvolvedor pode se concentrar na lógica e no processo de escolha dos hiperparâmetros gerais da aplicação. Nesse caso, o TensorFlow cuida dos detalhes nos bastidores.

O TensorFlow oferece facilitadores adicionais para desenvolvedores que precisam depurar e obter introspecção na criação de ML e DL. O modo de execução ávido permite executar avaliações e modificar, de maneira transparente, cada operação do grafo, em vez de considerá-lo como um objeto único, que necessitaria de uma análise completa de uma vez. A biblioteca ainda possui uma suíte de visualização, conhecida como *TensorBoard*, que possibilita ao desenvolvedor a criação e inspeção do perfil que indica a forma como os grafos são executados. Isso pode ser feito por meio de um painel interativo, que pode ser acessado pela web ou aplicativos móveis.

Embora a implementação de código aberto original não tivesse recursos distribuídos após a liberação, o que foi um problema inicial, a partir da versão 0.8.0, a funcionalidade de execução distribuída ficou disponível como parte da biblioteca *TensorFlow*, o que ajudou muito. Embora essa API distribuída seja um pouco pesada, é incrivelmente poderosa. A maioria das outras bibliotecas de aprendizagem de máquina não possui esses recursos, e é importante observar que a compatibilidade nativa, hoje com CUDA, da NVidia com determinados gerenciadores de *cluster* está sendo trabalhada.

O mecanismo de execução distribuído do *TensorFlow* abstrai e facilita os muitos dispositivos suportados e fornece um núcleo de alto desempenho implementado em linguagem C/C++ para a plataforma TensorFlow.

Além disso, o TensorFlow oferece a possibilidade de desenvolvimento em Python, C++, GO, Java, entre outros. As *Layers* fornecem uma interface mais simples para camadas comumente usadas em modelos de aprendizagem profunda. E encontramos ainda as APIs de nível superior, o que facilita, incluindo Keras[1] e a *Estimator API*, que facilita o treinamento e a avaliação de modelos distribuídos. E muitos modelos que estão semiprontos ou prontos podem ser usados.

O *TensorFlow* não é fácil de aprender, possui uma curva alta e tem um nível de complexidade relativamente alto, mas, ao mesmo tempo, é flexível e poderoso o que deixa o desenvolvedor com o poder, sendo atualmente o principal *software* para desenvolvimento em *Deep Learning* e aplicações de Inteligência Artificial.

1 https://keras.io/.

Keras

Keras é uma biblioteca para rede neural de alto nível escrita em Python e roda como *frontend* no *TensorFlow* ou *Theano* (já descontinuado). É possível substituir uma rede neural por outra utilizando Keras em suas implementações. Ela foi desenvolvida para facilitar experimentações rápidas e facilita no processo de construção, já que sua interface é mais fácil de ser utilizada, isto é, sem que você tenha que dominar cada um dos *backgrounds* de maneira rápida e eficiente.

A biblioteca é aplicável conforme:

- desenvolvimento rápido e fácil, com total modularidade, minimalismo e extensibilidade;
- suporte a redes convolucionais e recorrentes, o que está feito, incluindo combinação de ambas;
- suporte a esquemas de conectividade arbitrária (de N para N) em seus treinos;
- executa em CPU ou GPU.

> A força da *Deep Learning* vem basicamente de um único algoritmo muito importante: *Stochastic Gradient Descent* (SGD). (GOODFELLOW, Ian; 2015).

O Keras é um *framework* de *Deep Learning* muito simples e fácil de ser usado.

PyTorch

Outra biblioteca *open source* de *Machine Learning* para linguagem de programação Python é o PyTorch. Baseada no Torch, essa biblioteca é muito utilizada no desenvolvimento de aplicativos para processamento de linguagem natural. Ele é desenvolvido principalmente pelo grupo de pesquisa de inteligência artificial do Facebook e o *software* de linguagem de programação probabilística.

Um exemplo bem conhecido do que podemos fazer com o PyTorch é o projeto *Unpaired Image-to-Image Translation* usando CCAN, construído pelos cientistas da computação da Universidade de Berkeley. Esse projeto aprende e é treinado com a entrada de imagens e mapeia a saída usando um conjunto de treinamento com imagens alinhadas (são imagens gerais).

A versão do PyTorch atual teve algum sucesso e fornece flexibilidade para os pesquisadores de IA, desenvolvimento e desempenho em escala de produção, mas teve alguns desafios. De acordo com Jia (2015):

> Dado seu forte acoplamento com Python, frequentemente precisamos traduzir códigos de pesquisas, em scripts de treinamento ou modelos treinados, para a representação no modo de grafos do Caffe2 para executar em escala na produção. O executor com base em grafos do Caffe2 permite que o desenvolvedor obtenha a vantagem das otimizações que estão no estado da arte com transformações de grafos, reuso eficiente da memória, e uma integração ajustada com a interface do hardware. (JIA, 2015)

Alternativamente, o PyTorch 1.0 absorve e integra as capacidades do Caffe2 e ONNX, que são dois outros *frameworks*, e combina com a habilidade do PyTorch para fornecer um meio desacoplado e coeso entre a prototipação de pesquisa e a publicação em produção. O que facilita para levar os sistemas para produção com esse *framework*.

REFERÊNCIAS

"Google's AlphaGo AI wins three-match series against the world's best Go player". TechCrunch. , 25 Maio maio 2017.

ACKLEY, D. H.; HINTON, G. E.; SEJNOWSKI, T. J. ***A learning algorithm for boltzmann machines.*** Cognitive science, Elsevier, 1985.

AIZENBERG, Igor Aizenberg,; AIZENBERG, Naum N. Aizenberg,; JOOS, P.L. ***Vandewalle. Multi-Valued and Universal Binary Neurons:*** *Theory, Learning and Applications*. Springer Science & Business Media, 2000.

BAKER, J.; DENG, Li; GLASS, Jim; KHUDANPUR, S.; LEE, C.-H.; MORGAN, N.; O'SHAUGHNESSY, D. **"*Research Developments and Directions in Speech Recognition and Understanding*",** 2009.

Balázs Csanád Csáji. ***Approximation with Artificial Neural Networks.;*** Faculty of Sciences; Eötvös Loránd University, Hungary, 2011.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. ***Representation Learning:*** *A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.*

BENGIO, Yoshua; LECUN, Yann; HINTON, Geoffrey. ***"Deep Learning"***. Nature, 2015.

BOREN, W. L.;, HUNTER, T. B.;, BJELLAND, J. C.;, HUNT, K. R. ***Comparison of breast consistency at palpation with breast density at mammography.*** Invest Radiol, 1990.

DENG, L.; YU, D. *"**Deep Learning: Methods and Applications**"* **(PDF).** *Foundations and Trends in Signal Processing*, 2014.

HINTON, G. E.; SRIVASTAVA, N.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. R. ***"Improving neural networks by preventing co-adaptation of feature detectors"***, 2012.

HINTON, Geoffrey E.; DAYAN, Peter; FREY, Brendan J.; NEAL, Radford. ***"The wake-sleep algorithm for unsupervised neural networks"*,** 1995.

HORNIK, Kurt. *"**Approximation Capabilities of Multilayer Feedforward Networks**"*. Neural Networks, 1991.

KAPUR, Lenny. ***Neural Networks & The Backpropagation Algorithm, Explained.*** 1th ed. Spring, 2010.

KRIZHEVSKY; SUTSKEVER; HINTON. ***Methods research image.*** 2012.

LECUN *et al., "**Backpropagation Applied to Handwritten Zip Code Recognition**,"*, Neural Computation, 1989.

LU, Z., .; PU, H., .; WANG, F., .; HU, Z., .; & WANG, L. ***The Expressive Power of Neural Networks:*** *A View from the Width*, 2017.

LUI, Bing *et al***. *Lifelong Machine Learning:*** *Second Edition*. 2018.

MORGAN, Nelson; BOURLARD, Hervé; RENALS, Steve; COHEN, Michael; FRANCO, Horacio. ***"Hybrid neural network/hidden markov model systems for continuous speech recognition"***. *International Journal of Pattern Recognition and Artificial Intelligence*, 1998.

NASCIMENTO, Rodrigo. **Afinal, o que é Big Data?**. Disponível em: <http://marketingpordados.com/analise-de-dados/o-que-e-big-data-%F0%9F%A4%96/>. Publicado em: 2017. Acesso em: 17 de jun. de 2019.

OLSHAUSEN, B. A. ***"Emergence of simple-cell receptive field properties by learning a sparse code for natural images"***. Nature, 1996.

SCHMIDHUBER, J. ***"Deep Learning in Neural Networks: An Overview".*** Neural Networks, 2015.

SHAVLIK, J. W., and ***Dietterich, T. G. Readings in Machine Learning.*** San Mateo, CA: Morgan Kaufmann Publishers, 1990.

WAIBEL, A.; HANAZAWA, T.; HINTON, G.; SHIKANO, K.; LANG, K. J. ***"Phoneme recognition using time-delay neural networks"***, 1989.

WENG, J. WENG, ; AHUJA, N. AHUJA AND; HUANG, T. S. *HUANG,* ***"Learning recognition and segmentation using the Cresceptron",*** 1997.