Trabalho Prático 3 - Redes Neurais Artificiais

Hugo Araujo de Sousa

Computação Natural (2017/2)
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)

hugosousa@dcc.ufmg.br

Resumo. *Nesse trabalho são explorados conceitos relacionados a redes neurais, colocando-os em prática através da utilização da biblioteca Keras com Tensorflow, que nos permite abordar um problema de classificação.*

1. INTRODUÇÃO

Dentro da área de Computação Natural, o campo de Redes Neurais Artificiais tem como objetivo a criação de modelos computacionais inspirados pelo conhecimento que temos sobre como funciona o sistema nervoso, mais especificamente, na estrutura e função dos neurônios no cérebro [Brownlee 2011].

Dessa forma, uma Rede Neural Artificial é uma coleção de neurônios artificiais que são conectados a fim de se realizar alguma computação em padrões de entrada para gerar padrões de saída. Esses neurônios, então, se adaptam, modificando sua estrutura interna ao longo do tempo. Geralmente, essa modificação se dá através da atualização dos pesos das conexões entre os neurônios da rede. Esse processo define o aprendizado da rede neural, que então se torna, com o tempo, cada vez mais adaptada na tarefa de gerar o padrão de saída correto de acordo com a entrada.

No trabalho em questão, usaremos a biblioteca Keras [1] com Tensorflow [2], que juntas fornecem implementações de redes neurais já prontas para uso. A partir dessas duas bibliotecas, o problema a ser resolvido será o de classificação de um conjunto de dados específico.

Esse conjunto de dados reúne informações de 1429 proteínas, descritas por 8 atributos (números reais). Para cada uma dessas proteínas, a sua classe se refere à parte da célula em que a proteína se encontra. Ao todo, existem 7 classes possíveis, descritas na Tabela 1.

Portanto, a rede neural criada será alimentada com os dados dessa base e, ao longo do tempo, tentará aprender os padrões que determinam a saída da mesma, isto é, dado um conjunto de 8 atributos de uma proteína, a rede deve dizer qual é a posição que essa proteína ocupa na célula (representada pela classe, dentre as 7 descritas acima).

2. MODELAGEM

A modelagem do problema de classificação em questão, utilizando redes neurais implementadas através da biblioteca Keras com Tensorflow é mostrada nessa seção.

[1] https://keras.io/
[2] https://www.tensorflow.org/

Classe	Descrição
CYT	Citoplasma
MIT	Mitocôndria
ME1	Uma membrana específica da célula
ME2	Uma membrana específica da célula
ME3	Uma membrana específica da célula
EXC	Exterior da célula
NUC	Núcleo da célula

Tabela 1. Classes que descrevem a posição das proteínas em uma célula.

2.1. Arquitetura

Para modelar esse problema, o primeiro passo é determinar qual o tipo da rede neural a ser construída - existem vários tipos de arquitetura de redes neurais diferentes.

Para o trabalho em questão, optou-se por simplicidade em termos de representação, o que, por sua vez, acarreta em maior controle sobre a estrutura da rede e de seu funcionamento. Dessa forma, a arquitetura escolhida foi a *Multilayer Perceptron - MLP* (Perceptron de múltiplas camadas) -, que é uma versão da rede *Perceptron* generalizada para conter múltiplas camadas escondidas de neurônios. A Figura 1 mostra a estrutura da rede MLP.

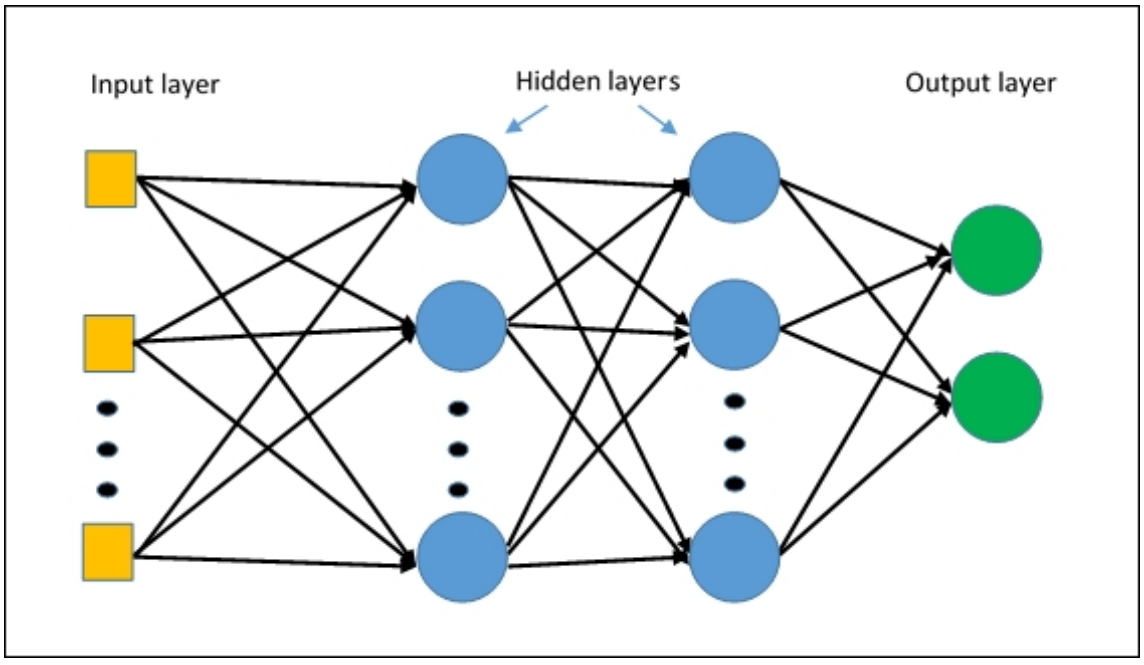

Figura 1. Estrutura da rede *Multilayer Perceptron*. Vemos que, além das camadas de entrada e saída, a rede também pode apresentar um número arbitrário de camadas escondidas de neurônios. O número de neurônios em cada camada também pode ser escolhido arbitrariamente.

O número de camadas escondidas e número de neurônios em cada camada utilizados no trabalho foram definidos experimentalmente, como mostrado na Seção 5.

De forma geral, temos que a saída que um neurônio produz consiste na soma ponderada (pelos pesos) de suas entradas, além de um valor de *bias*.

$$Saida = \sum(Pesos * Entradas) + Bias$$

Além disso, essa saída passa por uma função de ativação antes de servir como entrada para os neurônios da camada seguinte. Essa função de ativação determina se e quanto um neurônio deve contribuir na rede, de acordo com os valores que gera.

2.2. Inicialização de Pesos

Uma decisão importante é a de como inicializar os pesos na rede. Sabemos que se uma camada escondida tem um grande número de entradas, pequenas mudanças nos pesos podem causar grandes alterações nas saídas dessa camada. Uma forma de contornar isso é tornar os pesos iniciais da rede proporcionais ao número de entradas. Para esse trabalho, optou-se por inicializar os pesos proporcionalmente à raiz quadrada do número de entradas nas camadas. Na biblioteca Keras, pode-se fazer isso adicionando o argumento *kernel_initializer='lecun_uniform'* ao construtor de uma nova camada.

2.3. Funções de Ativação

2.4. Validação Cruzada

2.5. Codificação da Saída

3. IMPLEMENTAÇÃO

4. ESTRUTURA DO PROJETO E EXECUÇÃO

5. EXPERIMENTOS

6. CONCLUSÃO

7. REFERÊNCIAS

[Brownlee 2011] Brownlee, J. (2011). *Clever Algorithms: Nature-Inspired Programming Recipes*. Lulu.com, 1st edition.