UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

Mônica de Castro Henriques

Capacidade de Aproximação de Redes Residuais com Largura Mínima

Campinas
25/06/2025

Mônica de Castro Henriques

Capacidade de Aproximação de Redes Residuais com Largura Mínima

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina de Projeto Supervisionado I, sob a orientação do(a) Prof. Marcos Eduardo Valle.

Resumo

Uma rede residual (ou ResNet, do inglês *Residual Network*) é um tipo de rede neural profunda que incorpora conexões residuais (*skip connections*). Essas conexões permitem que o sinal de entrada seja transmitido diretamente para camadas mais profundas, contornando algumas camadas intermediárias.

As conexões residuais ajudam a combater o problema do desaparecimento do gradiente (*vanishing gradient problem*), permitindo que a rede aprenda a diferença (resíduo) entre a entrada e a saída esperada, em vez de precisar aprender a transformação completa.

Neste projeto, veremos que uma ResNet com apenas um neurônio por camada oculta possui capacidade de aproximação universal. Em outras palavras, mesmo com largura mínima, essas redes neurais podem aproximar qualquer função integrável arbitrariamente bem, desde que tenham profundidade suficiente.

Abstract

A residual network (or ResNet) is a type of deep neural network that incorporates residual (skip) connections. These connections allow the input signal to be transmitted directly to deeper layers, bypassing some intermediate layers.

Residual connections help address the vanishing gradient problem by enabling the network to learn the residual (difference) between the input and the expected output, rather than having to learn the full transformation.

In this project, we will see that a ResNet with only one neuron per hidden layer possesses universal approximation capability. In other words, even with minimal width, these neural networks can approximate any integrable function arbitrarily well, as long as they have sufficient depth.

Conteúdo

1 Introdução

Desde o surgimento dos primeiros neurônios artificiais em 1943, e posteriormente das redes neurais, diversos modelos foram desenvolvidos e aprimorados ao longo das décadas. O modelo *Multilyer Perceptron* (MLP), surgiu na década de 80 e, apesar de ser um modelo simples, ainda é muito utilizado tanto para problemas de classificação como de regressão [4]. O modelo, como o próprio nome sugere, consiste em um rede neural artificial de múltiplas camadas de neurônios dispostos de uma maneira hierárquica. Pode ser considerada larga no caso em que possui muitos neurônios em uma mesma camada ou profunda quando possui um grande número de camadas.

Em 2016, com a finalidade de solucionar um problema recorrente em diversos modelos de redes neurais, o desaparecimento do gradiente, foi desenvolvida a rede neural residual (ResNet) [2]. O desaparecimento do gradiente surge devido ao uso do *backpropagation* em redes muito profundas, uma vez que os gradientes dos pesos das camadas iniciais são multiplicados por diversos termos menores do que 1, fazendo com que se aproximem de zero. A estratégia adotada para contornar esse problema, foi criar um bloco residual que utiliza um atalho, o qual consiste em tomar a saída de uma camada como a saída da função de ativação somada à entrada da camada.

Tendo em vista o contexto do desenvolvimento das redes neurais e o sucesso que vêm tendo para resolver grandes problemas, alguns resultados foram cruciais. Um resultado muito importante foi proposto em 1989 por Geroge Cybenko, o chamado teorema da aproximação universal, voltado para uma rede MLP com apenas uma camada oculta. Neste projeto, veremos que uma rede ResNet com apenas um neurônio por camada também possui capacidade de aproximação universal. Ao longo do trabalho foram feitos experimentos computacionais a fim de tornar mais visuais as proposições. Utilizando a linguagem de programação `python` foram geradas redes MLP e ResNet as quais foram utilizadas para resolver alguns problemas de classificação e regressão.

2 Desenvolvimento

Redes Neurais é o nome dado a uma grande classe de modelos e técnicas de apendizado de máquinas (*machine learning*). Uma rede neural é uma estrutura composta

por duas ou mais unidades de processamento, chamadas neurônios. O processamento de uma rede neural é baseado no funcionamento do sistema nervoso humano, ou seja, os neurônios recebem um sinal, processam, mandam adiante até que se atinja a saída da rede.

Ao longo das décadas, foram se desenvolvendo modelos mais complexos de redes, isso devido ao crescente nível de complexidade dos problemas a serem resolvidos. Alguns dos modelos mais conhecidos são a *Multilyer Perceptron* (MLP), a qual se tornou base para muitas outras. Além disso, quando tratamos de uma aprendizagem mais profunda, ou seja, redes com três ou mais camadas, temos a LeNet, VGG-Net e ResNet. Nesse projeto, desenvolveremos mais a fundo sobre os modelos MLP e ResNet.

2.1 Modelo de rede neural Multilyer Perceptron (MLP)

O modelo de rede neural *Multilayer Perceptron*, ou MLP como é mais conhecida, surgiu em 1986 como contribuição de Rumelhart e McClelland. Como o próprio nome sugere, consiste em diversos perceptrons organizados em camadas (*layers*) sucessivas.

Uma camada é um conjunto de m neurônios artificiais em paralelo. Do ponto de vista matemático, uma camada com m neurônios define uma função $L : \mathbb{R}^n \to \mathbb{R}^m$ dada pela equação

$$L(\mathbf{x}) = \varphi(\mathbf{W}\mathbf{x} - \mathbf{b}), \tag{1}$$

em que a função de ativação φ é aplicada a cada compomente, $\mathbf{W} \in \mathbb{R}^{m \times n}$ e $\mathbf{b} \in \mathbb{R}^m$ são a matriz dos pesos sinápticos e o vetor viés, respectivamente. Nesse projeto, utilizamos algumas funções de ativação tais como:

- Função ReLu:

$$\texttt{relu}(t) = max\{0, t\} \in [0, +\infty) \tag{2}$$

- Função Logistica (ou sigmoide):

$$\sigma(t) = \frac{1}{(1 + e^{-t})} \in (0, 1) \tag{3}$$

Uma rede neural de múltiplas camadas, é dada pela composição de camadas de neurônios L_k para $k = 1, 2, 3, \ldots$. Formalizando, se temos uma rede progressiva com K camadas, podemos definir uma função $N : \mathbb{R}^n \to \mathbb{R}^m$, a qual é dada pela composição

$$N = L_K \circ \cdots \circ L_2 \circ L_1, \tag{4}$$

em que $L_K : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}$ são camadas de neurônios para $k = 1, \ldots, K$, com $n = n_0$ e $m = n_K$. Chamamos a camada L_K de camada de saída, por ser a última da rede. As camadas anteriores são as intermediárias ou ocultas [1].

A fim de verificar o desempenho da MLP, escolhemos alguns exemplos de problemas. Dois problemas de classificação e dois de regressão. Desse modo, para um problema de classificação, utilizamos os seguintes hiperparâmetros: além das camadas de saída e entrada, utilizamos apenas uma camada oculta com cinco neurônios; há a presença de viés (*bias*) e para a camada oculta foi utilizada a função de ativação ReLu dada por (2), enquanto que para a camada de saída foi utilizada a função logística definida em (3). Para os problemas de regressão utilizamos uma arquitetura muito parecida, entretanto na camada oculta utilizamos até 50 neurônios (dependendo da complexidade da função a ser aproximada) e não há uma função de ativação para a camada de saída.

Um resultado importante no caso das redes MLP é teorema da Aproximação Universal, o qual foi desenvolvido e provado por diversos pesquisadores ao longo das décadas de 80 e 90, sendo um dos primeiros Cybenko em 1989 [5].

Teorema 1. *Seja $f : \mathbb{R} \to \mathbb{R}$ uma função contínua, $\mathbf{X} \subset \mathbb{R}^n$ um compacto e φ uma função contínua não polinomial. Dado $\epsilon > 0$, existe uma rede MLP N, com uma única camada oculta contendo M neurônios com φ como função de ativação, tal que*

$$|f(x) - N(x)| < \epsilon, \forall \mathbf{x} \in \mathbf{X}. \tag{5}$$

De modo geral, tendo uma função contínua f, é possível construir uma rede MLP com apenas uma camada oculta que aproxima tão bem quanto se queira f em um compacto.

2.2 Rede Neural Residual (ResNet)

A rede neural residual (também conhecida como ResNet), foi desenvolvida com o propósito de solucionar o problema do desaparecimento do gradiente quando tratamos de redes muito profundas. O desaparececimento do gradiente (*gradient vanishing*) decorre do uso do *backpropagation* em redes muito profundas. Isso ocorre, pois os gradientes das camadas iniciais são produtos de diversos termos de magnitude menor que 1, ou seja, muito próximos de zeros, sendo assim os pesos das camadas iniciais deixam de sofrer ajustes significativos e, desse modo, o treinamento acaba por estagnar.

A ideia utilizada para resolver esse problema foi criar um atalho (*short cut* ou *skip conection*), o qual consiste em somar a entrada à saída de uma ou mais camadas, ou seja, $R(x) = F(x)+x$. A composição das camadas que faz esse atalho é chamado de bloco residual, o elemento base da ResNet. Sendo assim, a arquitetura básica de uma Resnet é formada por uma sequência de blocos residuais.

De acordo com [3], o bloco residual foi estruturado da seguinte forma:

Seja $R(x)$ uma função de $\mathbb{R}^n \to \mathbb{R}^n$, temos

$$R(\mathbf{x}) = \mathbf{x} + \mathtt{vrelu}(\mathbf{u}^T\mathbf{x} + b), \quad \mathbf{x},\mathbf{u},\mathbf{v} \in \mathbb{R}^n \quad e \quad b \in \mathbb{R}. \tag{6}$$

O bloco residual pode ser expresso em termos de camadas densas dadas por (1), com m sendo o número de neurônios, φ a função de ativação e b o viés. Desse modo denotamos a camada densa por,

$$D = D(m, \varphi, b) \tag{7}$$

Assim, temos

$$\begin{aligned} y &= D_u(1, \mathtt{relu}, b)(\mathbf{x}) \\ R(\mathbf{x}) &= \mathbf{x} + D(n, \mathtt{none}, 0)(y) \end{aligned} \tag{8}$$

Desse modo, de maneira análoga à rede MLP, a ResNet consiste em uma composição de blocos residuais. Formalizando, se temos uma ResNet progressiva com K

camadas, podemos definir uma função $N : \mathrm{R}^n \to \mathrm{R}^m$, a qual é dada pela composição

$$N = R_K \circ \cdots \circ R_2 \circ R_1, \tag{9}$$

em que $R_K : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}$ são blocos residuais para $k = 1, \ldots, K$, com $n = n_0$ e $m = n_K$.

A arquitetura da ResNet que utilizamos para resolver os exemplos propostos, sofre variação dependendo do tipo e complexidade do problema. Por exemplo, para os problemas de classificação foram utilizadas entre 5 e 15 camadas (a depender da complexidade), sem viés e com a função logística (3) como função de ativação na camada de saída. Já no caso dos problemas de regressão, utilizamos entre 50 e 100 camadas, sem viés e sem nenhuma função de ativação na camada de saída.

Assim como para a rede MLP, vem sendo estudando o teorema que garante a aproximação universal para uma rede ResNet [3]. Abaixo deixamos o enunciado do teorema.

Teorema 2. *Para qualquer $d \in \mathbb{N}$ a família da ResNet com um único neurônio por camada oculta e função de ativação ReLu pode aproximar qualquer função $f \in l_1(\mathbb{R}^d)$. Em outras palavras, para qualquer $\epsilon > 0$, existe uma ResNet N com um número finito de camadas tal que*

$$\int_{\mathbb{R}^d} |f(x) - N(x)| dx < \epsilon. \tag{10}$$

Em particular, dada uma função contínua por partes e com uma quantidade finita de descontinuidades, é possível construir uma rede ResNet, com finitas camadas ocultas, as quais possuem apenas um neurônio, que aproxima tão bem quanto se queira a função f.

3 Resultados

Como foi mencionado no início desse projeto, por meio da linguagem de programação `python` juntamente com bibliotecas constantemente utilizadas para problemas de *machine learning* tais como *tensorflow* e *keras*, geramos redes neurais tanto MLP quanto ResNet para resolver alguns problemas de classificação e regressão. Primeiramente, trabalhamos com problemas de classificação, utilizamos o tradicional problema

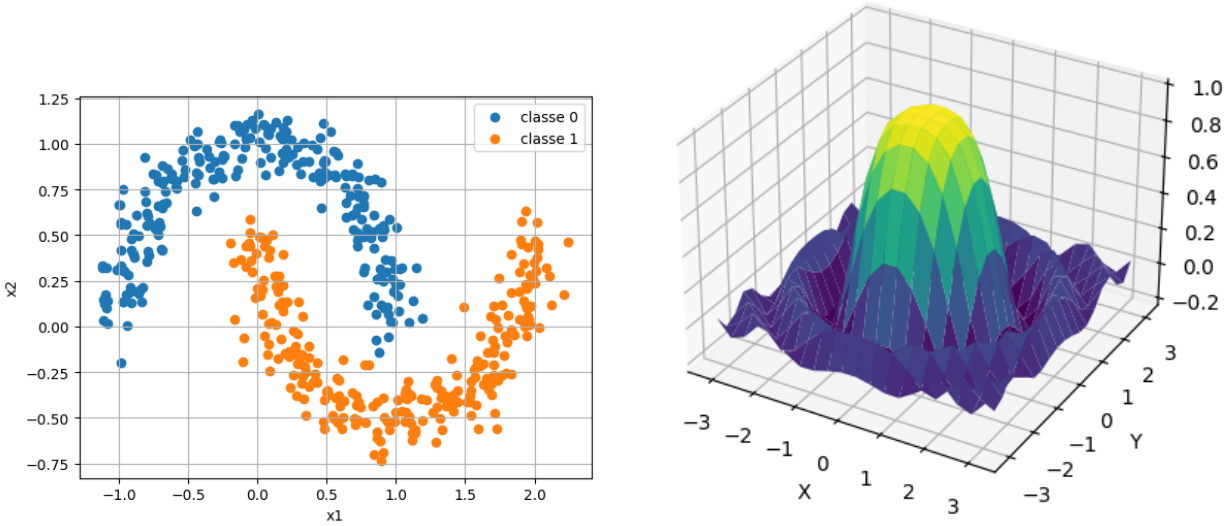

Figura 1: Representação gráfica dos problemas de classificação e regressão a serem aproximados.

de duas meia-luas. Depois, para o problema de regressão, tentamos aproximar a função $g(x, y) = \frac{sin(x^2+y^2)}{x^2+y^2}$, também conhecida como chapéu mexicano. A figura 1 mostra graficamente como é cada um desses problemas.

A fim de comparar o desempenho de cada uma das redes, levamos em conta a menor acurácia obtida no treinamento de cada rede ao longo de 100 épocas. Para a rede MLP a maior acurácia obtida foi de $0,9478$. Já para a ResNet, conseguimos uma acurácia máxima, ou seja, foi capaz de classificar todos os pontos do conjunto corretamente. Na figura 2, podemos ver graficamente o desempenho de cada um dos modelos.

Entretanto, a quantidade de parâmetros utilizados pela rede ResNet é muito superior à quantidade utilizada pela MLP. Desse modo, para fins de comparação, montamos um gráfico de Parâmetros vs Acurácia. Para montar esse gráfico, criamos dois laços que geram e treinam 10 redes MLP e ResNet, variando o número de neurônios e camadas ocultas. Para as redes MLP variamos o número de neurônios na única camada oculta entre 1 e 50. Já no caso da ResNet, variamos a quantidade de camadas ocultas dentre 50 e 100. Por fim, geramos dois vetores, um para guardar o número de parâmetros e outro para guardar o último valor de acurácia de cada treinamento. Na figura 3 podemos ver esse gráfico comparativo.

Para o problema de regressão, seguimos o mesmo caminho do problema an-

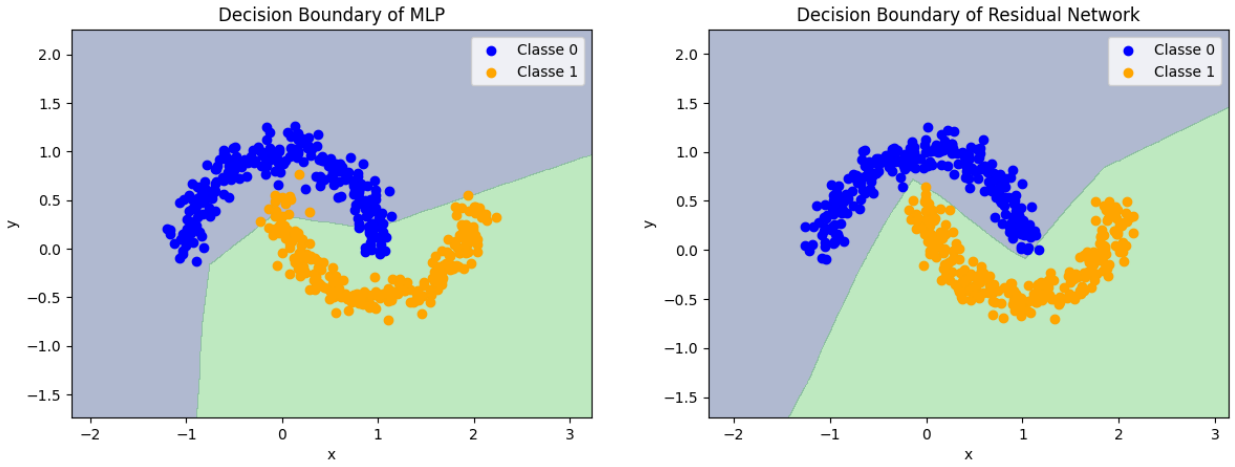

Figura 2: Representação gráfica dos desempenhos das redes MLP e ResNet respectivamente, para o problema de classificação.

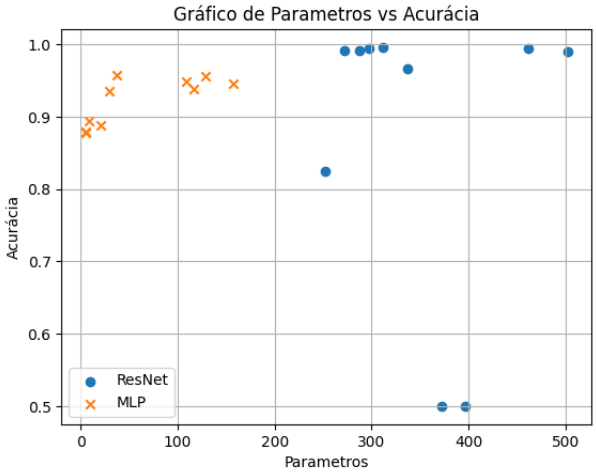

Figura 3: Gráfico comparativo Parâmetros vs Acurácia entre as redes MLP e ResNet no problema de classificação.

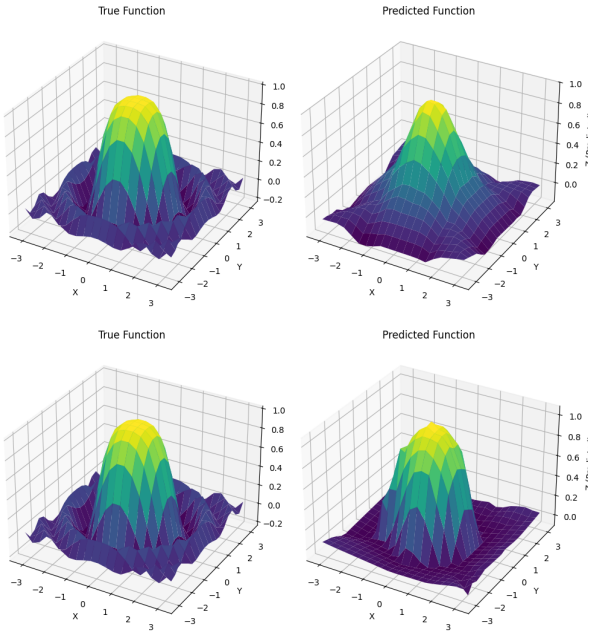

Figura 4: Aproximações da função g(x,y) pelas redes MLP e ResNet, respectivamente e comparação com a função original.

terior. Primeiramente comparamos cada um dos modelos individualmente e mostramos uma representação gráfica a fim de vermos o quanto as aproximações geradas pelas redes neurais se aproximam da função original. Como se trata de um problema mais complexo, foi necessário que treinássemos em 500 épocas. E, diferentemente do problema de classificação, não comparamos as acurácias, mas sim o erro quadrático médio (MSE). Na figura 4 podemos ver essa comparação gráfica.

Da mesma maneira, geramos 10 redes consecutivas para cada um dos modelos, variando apenas a quantidade de parâmetros e guardado em um vetor tanto os parâmetros quanto o último valor de MSE de cada treinamento. Na figura 5 podemos ver o gráfico com essa comparação.

4 Conclusão

De modo geral, com base nos resultados apresentados, podemos perceber que a rede MLP possui uma capacidade de aproximação muito alta, conseguindo resultados muito bons tanto para problemas de classificação quanto para regressão. Entretanto, a ResNet consegue obter resultados ainda melhores, alcançando uma acurácia máxima no

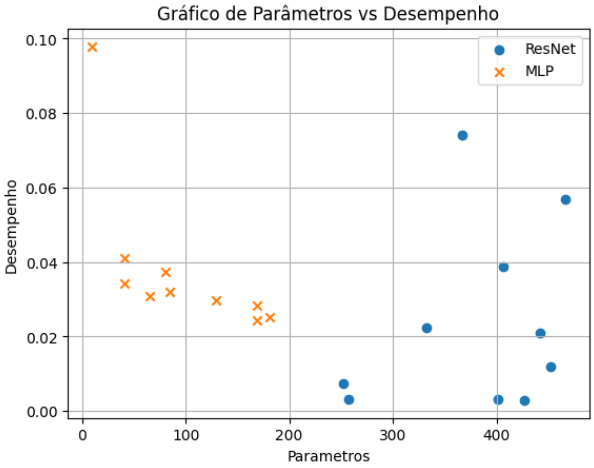

Figura 5: Gráfico comparativo parâmetro vs desempenho entre as redes MLP e ResNet no problema de regressão.

problema de classificação, além de uma aproximação muito próxima da função original no problema de regressão.

Com base na análise dos gráficos comparativos, pudemos perceber que, para o problema de classificação, apesar da ResNet ter alcançado uma acurácia máxima, precisou de um número de parâmetros bem superior ao número da MLP, a qual não ficou muito atrás em questão do valor da acurácia. Esse resultado muda quando vamos para o problema de regressão, no qual, apesar da ResNet continuar precisando de uma quantidade de parâmetros muito maior que a da MLP, seu desempenho também acaba por ser muito superior, conseguindo aproximações muito mais próximas às funções originais.

Sendo assim, alguns questionamentos e ideias poderiam ser desenvolvidas em projetos futuros. Como exemplo, se conseguimos construir uma rede com menos parâmetros, mas que obtivesse um alto desempenho, ou seja um baixo valor para o MSE. Isso poderia ser feito, "misturando"as duas redes, diminuindo a quantidade de camadas ocultas e em cada uma dessas camadas, aumentar a quantidade de neurônios. Desse modo, possivelmente poderíamos encontrar uma rede intermediária com desempenho tão bom quanto o da ResNet.

Referências

[1] Aurélien Géron. *Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow*. O'Reilly Media, São Paulo, 1ª edition, 2019.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[3] Hongzhou Lin and Stefanie Jegelka. Resnet with one-neuron hidden layers is a universal approximator. *arXiv preprint arXiv:1806.10909*, 2018. URL `https://arxiv.org/abs/1806.10909`.

[4] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing: Explorations in the Microstrutures of Cognition, volume 1.* Press, Cambridge, MA, 1986.

[5] Marcos Eduardo Valle, Wington L. Vital, and Guilherme Vieira. Universal approximation theorem for vector- and hypercomplex-valued neural networks. *Elsevier*, 2024.