Controle de um pêndulo invertido utilizando uma rede neural *feedforward*.

1st André Eduardo Alessi
Acadêmico do Curso de Engenharia de Computação
Universidade Tecnológica Federal do Paraná
Pato Branco, Brasil
andrealessi@alunos.utfpr.edu.br

2nd Vinícius Nakalski Nicolau
Acadêmico do Curso de Engenharia de Computação
Universidade Tecnológica Federal do Paraná
Patro Branco, Brasil
viniciusnicolau@alunos.utfpr.edu.br

***Resumo*—In this article, a theoretical survey of the main basics of Artificial Neural Networks (ANN) is made, as well as an application in Control Engineering, using an ANN to solve the inverted pendulum problem in an graphical environment made by GYM. The graph of the ANN was provided by the TensorFlow tools.**

***Index Terms*—Feedforward Neural Network, Inverted Pendulum, Control Engineering, Tensorflow, Gym.**

I. Introdução

A. Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) é um modelo computacional inspirado pela maneira em que o cérebro humano processa informações. Nos últimos anos, as redes neurais foram objeto de diversas pesquisas devido a grandes descobertas feitas na área de reconhecimento de voz, visão computacional e de processamento de textos [6].

Segundo [4], uma RNA é uma combinação paralela massiva de unidades simples de processamento que podem adquirir conhecimento de um ambiente através de um processo de aprendizagem e armazenamento do conhecimento e suas conexões. As redes neurais possuem diversas aplicações. Neste trabalho utilizou-se de uma para a solução de um problema clássico de engeharia de controle, o pêndulo invertido, e também será discutido básicamente a estrutura e o funcinamento de uma RNA.

B. TensorFlow

O TensorFlow é uma biblioteca de *software* em código aberto para computação numérica que usa gráficos de fluxo de dados. Foi desenvolvido por pesquisadores e engenheiros da Google Brain Team no departamento de pesquisas de inteligência de máquina do Google para realizar pesquisas sobre redes neurais profundas e aprendizado de máquina [13].

C. OpenAI - GYM

OpenAI é uma empresa de pesquisa sem fins lucrativos sobre inteligência artificial, criadora do Gym, um pacote de ferramentas para criar e comparar algoritmos de aprendizado por reforço. Este pacote consiste em uma coleção de problemas de teste (ambientes), que podem ser utilizados para trabalhar com algoritmos escritos pelos usuários [9].

D. O problema do pêndulo invertido

Segundo [8], o pêndulo invertido é um problema clássico na Engenharia de Controle, e existem implementações da robótica ao lançamento de foguetes. O Pêndulo Invertido é um sistema instável, mas é possivel atingir a estabilidade ao se cumprir certas condições. Muita pesquisa tem sido feita na área para descobrir novos métodos de controle para este sistema. Neste trabalho foi desenvolvido o sistema de controle para um pêndulo invertido com o grau de liberdade igual a um, o que significa que a base somente se movimenta no eixo X.

II. Referencial teórico

A. Neurônio Artificial

A unidade básica computacional de uma rede neural é o neurônio. Ele recebe uma entrada de outros neurônios ou de alguma fonte externa e computa uma saida. Cada entrada está associada com um peso w_i, no qual é atribuido um valor que determina sua importância sobre as outras entradas. O neurônio aplica uma função de ativação φ (definida abaixo) sobre a soma ponderada de suas entradas. A saida y_k de um neurônio pode ser definida como:

$$y_k = \varphi(\sum_{i=1}^{n} w_i * x_i + b_k) \tag{1}$$

Onde b_k é um valor não nulo que tem a função de sempre manter o neurônio ativo. Na figura 1, pode-se observar uma representação visual de um neurônio artificial.

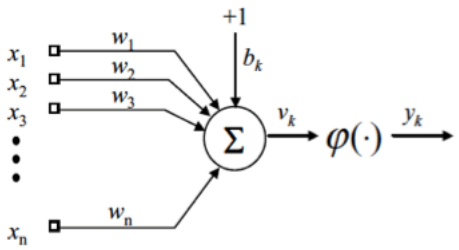

Figura 1. Neurônio Artificial. Fonte: [11].

B. Arquitetura de uma rede neural

Segundo [5], uma rede neural é composta por diversos elementos:

- **Camada de entrada:** Nenhum processo de computação é realizado nesta camada, ela somente passa a informação para o próximo nível da rede.
- **Camada oculta:** É onde ocorre computação intermediária, são feitos processamentos e transferência de pesos da camada de entrada para o próximo nível (camada oculta ou de saída).
- **Camada de saída:** É onde utiliza-se a função de ativação para mapear o formato de saída desejado.
- **Conexões e pesos:** A rede consiste de conexões transferindo a saída de um neurônio i para a entrada de outro neurônio j. Cada conexão possui um peso W_{ij}.
- **Função de ativação:** Define a saída de um nó dada uma entrada ou conjunto de entradas, ou seja, aplicando-se a soma das entradas ponderadas ela resulta em um unico valor que será atribuido ao neuronio. O principal objetivo da Função de transferência é introduzir não linearidades para a solução de um problema .Neste trabalho foram usados dois tipos distintos de funções de ativação. Nas camadas ocultas foi usado $ReLU$ que é uma função linear retificada, que na engenharia elétrica é conhecida como função rampa [10]. Já na camada de saida foi utilizado a *Softmax* que é uma função que comprime um vetor K- dimensional em um valor real, que é usado como resultado probabilistico da rede neural. [1]
- **Regra de aprendizado:** É uma regra ou um algoritmo que modifica os parâmetros de uma rede neural para uma dada entrada produzir uma saída favorável. Normalmente envolve a modificação de pesos e limites.

C. Gradiente descendente

Segundo [12], o algoritmo do gradiente descendente tem a finalidade de minimizar a função custo. A consequência disso é que o modelo terá o menor erro possível, aumentando sua acurácia.

D. Backpropagation

O *backpropagation* está entre as coisas mais importantes em uma rede neural artificial, seu objetivo é atualizar os valores dos pesos de cada neurônio a partir de um erro δ.

$$\delta = \frac{\partial C}{\partial net} \tag{2}$$

onde C é o valor do neurônio e net são todas as entradas ponderadas do neurônio.

Pode-se dizer que os algoritmos de *backpropagation* possuem as seguintes etapas:

- Inicialização dos pesos da rede neural.
- Propagar as entradas a frente através da rede neural.
- Calcular o δ dos neurônios da camada de saída.
- Realizar a retropropagação para calcular os δ para todos os neurônios das camadas ocultas.
- Os erros da saída e da entrada de um neurônio são multiplicados para achar o gradiente do peso.
- Assim subtrai-se do peso uma fração do gradiente do peso que é determinada pela taxa de aprendizado.

Todo o processo de *backpropagation* é complexo e foge do escopo deste trabalho, porém uma implementação completa pode ser encontrada em [14].

E. Métodos de análise

O desempenho de uma rede neural pode ser medido de várias formas. Algumas delas incluem *loss*, *accuracy* e histogramas de cada peso acionado. *Accuracy* é a porcentagem com que um indivíduo do conjunto de validação foi aprovado. *Loss*, diferentemente da *accuracy*, não é uma porcentagem, mas sim uma soma feita pelos erros de cada exemplo no conjunto de treinamento ou no conjunto de validação.

Com outras palavras: o conjunto de treinamento consiste nos indivíduos que irão treinar a rede, já o conjunto de validação trata-se dos indivíduos utilizados após o treinamento para realizar o ajuste dos micro-parâmetros da rede neural.

F. ADAM

A escolha entre os algoritmos de gradiente descendente é o fator que determina se serão obtidos bons resultados em alguns minutos, horas e até dias [2].

O algoritmo ADAM, proposto em [7], é uma evolução do clássico Gradiente Descendente Estocástico (GDE), e obteve grande utilização com a expansão das Redes Neurais Profundas (RNP). Diferente do GDE, o ADAM mantém uma *Learning Rate* fixa para todas as atualizações de peso que ocorrem durante o treinamento. O ADAM também combina as vantagens de outros dois algoritmos bem populares: o AdaGrad e RMSProp, que modificam exponencialmente a *Learning Rate* conforme o treinamento progride.

Segundo [1], "Empiricamente demonstramos que o ADAM funciona bem na prática e se compara favoravelmente aos outros metodos de otimização estocásticos", o que pode ser observado na Figura 2. Os fatores de configuração do ADAM são:

- **Alpha**: o tamanho do passo, ou seja, a proporção em que os pesos serão atualizados. Valores grandes resultam em um aprendizado mais rápido, mas também não garantem uma melhor otimização.
- **Beta1**: taxa de queda exponencial das estimativas de primeiro momento.
- **Beta2**: taxa de queda exponencial das estimativas de segundo momento, deve ter um valor próximo a 1 para problemas esparsos, como problemas de visão computacional.
- **Epsilon**: um número muito pequeno para evitar uma divisão por zero.

Segundo [1], para uma configuração boa utiliza-se:

- Alpha = 0.001
- Beta1 = 0.9
- Beta2 = 0.999
- Epsilon = 10E-8

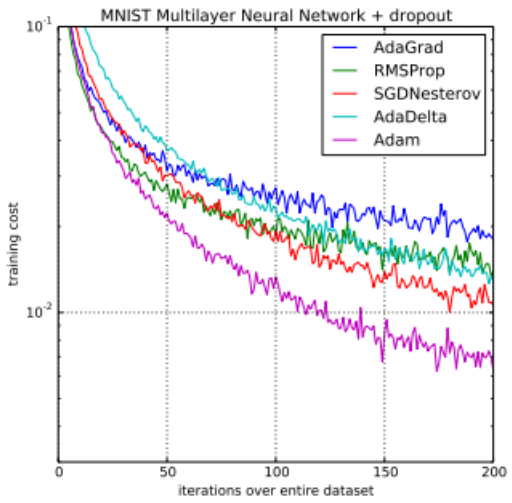

Figura 2. Comparativo entre algoritmos de gradiente descendente. Fonte: [2]

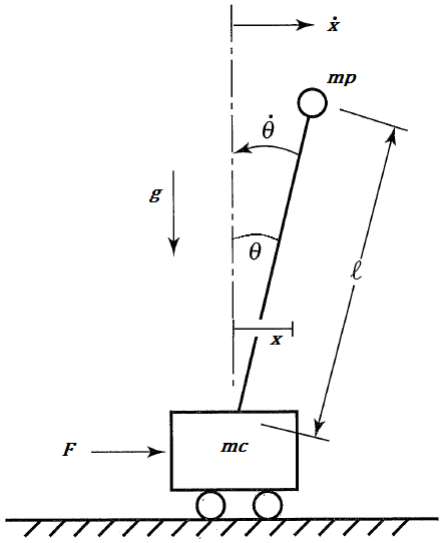

Figura 3. Pêndulo invertido. Fonte: [3]

G. Estrutura do pêndulo invertido

O pêndulo utilizado para este trabalho tem as seguintes características, g é a aceleração da gravidade, m_c é a massa do carrinho, m_p é a massa do pêndulo, l é o comprimento do pêndulo, F é a força aplicada na base móvel, τ é o período de amostragem, $temp$ é uma variável temporária para reduzir o tamanho das expressões, θ_{aacc} é a aceleração angular, X_{acc} é aceleração no eixo X, x é a posição no eixo X e $\dot{\theta}$ é a velocidade angular. As variáveis de estado são inicializadas em zero.

Tabela I
VALORES DE PARÂMETROS DO AMBIENTE

Parâmetros do ambiente	Valores
Aceleração da gravidade (g)	9.8 m/s^2
Massa da base móvel (m_c)	1 Kg
Massa do pêndulo (m_p)	0.1 Kg
Comprimento do pêndulo (l)	1 m
Força aplicada (F)	20 N
Taxa de amostragem (τ)	0.01 s

$$m_t = m_c + m_p$$

$$temp = \frac{F + l\dot{\theta}^2 \sin\theta}{m_t}$$

$$\theta_{aacc} = \frac{m_t(g\sin\theta - temp\cos\theta)}{l(\frac{4}{3} - m_p\cos^2\theta)}$$

$$X_{acc} = temp - \frac{l\theta_{aacc}\cos\theta}{m_t}$$

$$x = x + \tau\dot{x}$$

$$\dot{x} = \dot{x} + \tau X_{acc}$$

$$\dot{\theta} = \dot{\theta} + \tau\theta_{aacc}$$

$$estado = [x, \dot{x}, \theta, \dot{\theta}]$$

Na Figura 3 temos a descrição das características do pêndulo de maneira visual.

III. MÉTODOS

A. Estrutura da rede neural

A rede neural utilizada possui sete camadas completamente conectadas, distribuidas da seguinte maneira:

- **Input:** Contendo 4 neurônios que recebem os quatro valores do vetor $estado$.
- **Primeira Camada:** Contém 256 neurônios.
- **Segunda Camada:** Contém 256 neurônios.
- **Terceira Camada:** Contém 512 neurônios.
- **Quarta Camada:** Contém 256 neurônios.
- **Quinta Camada:** Contém 128 neurônios.
- **Output:** Contém 2 neurônios que são a convergência da rede e determinam o lado que a força F irá ser aplicada sobre a base móvel.

Após cada uma das camadas ocultas, há uma função de *Dropout*, que tem como objetivo fazer a desativação de alguns neurônios durante o treinamento para evitar que a rede neural sempre resulte no mesmo valor, assim o valor de convergência será mais confiável. Após alguma experimentação, foi encontrado que o valor de 20% de desativação aleatória era o ideal.

Antes da camada output há um estimador que resultará no valor de *Loss* da rede neural.

O grafo que descreve a rede neural encontra-se na Figura 4.

B. Seleção dos indivíduos

No ambiente CartPole-V1 do Gym, a única váriavel controlável é o lado em que a força F será aplicada, sendo que '1' empurra a base móvel para a direta e '0' para a esquerda. O sistema de recompensa do ambiente funciona desta maneira: para cada ação válida para o pêndulo, ele soma um valor unitário na váriavel *reward*. Assim, foram gerados dez mil

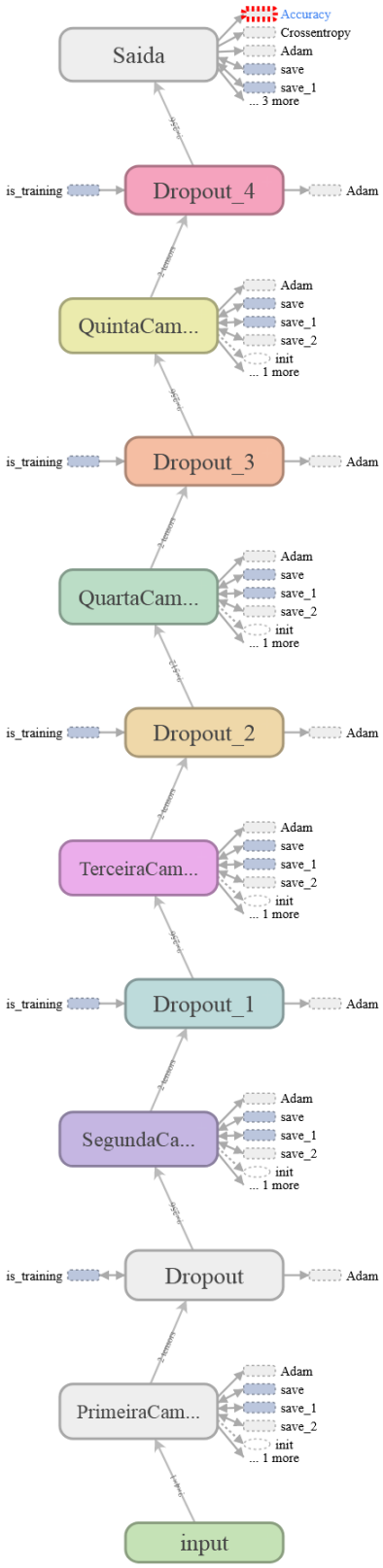

Figura 4. Grafo da Rede Neural. Fonte: Autoria própria.

indivíduos aleatórios que performavam ações até atingirem 500 passos de simulação ou fracassarem quando ultrapassarem os limites da tela ou atingissem o ângulo limite de 20°. Sendo assim, para todos os indivíduos em que $reward \geq 130$ seriam selecionados para o treinamento da rede neural.

C. Treinamento

Após o filtro de seleção, dos dez mil indivíduos, somente 230 elementos foram selecionados.

Para o treinamento, foram realizados seis épocas, resultando em 1465 passos de treino. O tempo total de treinamento foi de 1 minuto e 26 segundos. Os resultados de *Loss* e *Accuracy* podem ser encontrados nas figuras 5 e 6, respectivamente. A curva laranja desvanecida no fundo dos dois gráficos representa os dados brutos. Já a curva em laranja forte representa os mesmos dados com uma suavização de 97%.

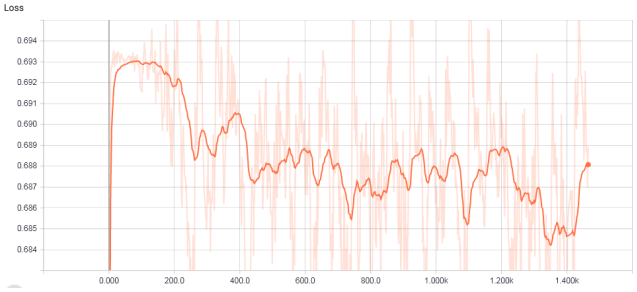

Figura 5. Curva de *Loss* em função do passo de treinamento. Fonte: Autoria própria.

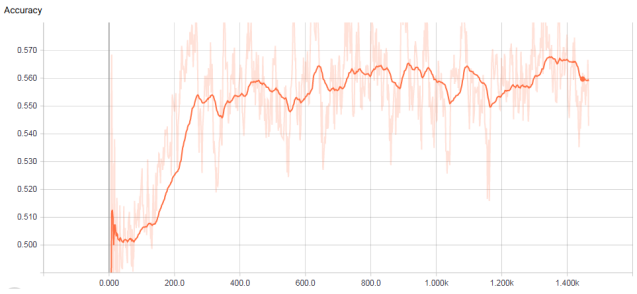

Figura 6. Curva de *Accuracy* em função do passo de treinamento. Fonte: Autoria própria.

Para visualização dos dados de distribuição e histogramas, um arquivo de log completo para ser usado em conjunto com o Tensorboard será disponibilizado. Todos os códigos utilizados neste trabalho para futura reprodução estão disponíveis em *https://github.com/Nakalski/Pendulum-Control*. Foi utilizando um processador Dual Core Intel i7-3517u e 4 GB de memória RAM para realizar o treinamento.

IV. Resultados e Discussões

Após a sessão de treinamento, os dados referentes aos pesos que interligam as camadas são carregados e a simulação incia com um movimento aleatório. Os movimentos seguintes são a predição da rede neural, sempre mantendo o pêndulo estável.

Para verificar a ação de controle, o usuário pode realizar uma perturbação externa utilizando-se das teclas 'A' e 'D' para acionar a força F tanto para a esquerda como para a direita.

Na Figura 7 há uma renderização das acões que foram realizadas para compensar uma perturbação para a esquerda, comprovando que a rede neural possui a habilidade de compensar o sistema e mantê-lo estável.

Contudo, se a perturbação durar muitos instantes, o sistema é levado para a instabilidade, não conseguindo se recuperar, resultando no término da simulação.

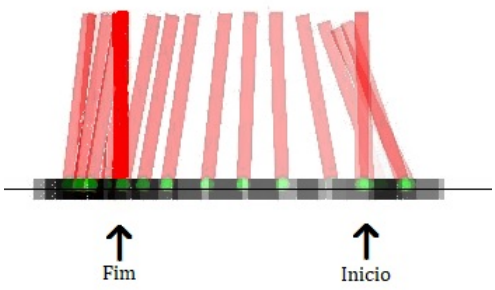

Figura 7. Renderização do ambiente com predição de acão pela rede neural. Fonte: Autoria própria.

V. Conclusões

Neste trabalho foi apresentado um resumo da teoria de redes neurais artificiais e uma alternativa para um problema clássico da Engenharia de Controle, o problema do pêndulo invetido, utilizando redes neurais *feedforward*.

Notou-se que os resultados foram satisfatórios já que o pêndulo conseguiu se manter estável para pertubações razoáveis. Suas limitações são as mesmas das técnicas clássicas de controle: há um limite do quanto se pode pertubar o sistema sem que este se torne instável.

A maior vantagem da utilização da abordagem por redes neurais se dá por não ser necessário toda a modelagem rigorosa do controle clássico, apenas uma simples abstração do problema. Além disso, a comunidade de pesquisadores de redes neurais é muito forte e ativa, tendo grande disseminação de conhecimento e *templates* para se trabalhar.

Como sugestão para trabalhos futuros, sugere-se utilizar os moldes deste trabalho para desenvolver um controle por redes neurais do problema do pêndulo invertido duplo, além de outros problemas clássicos de controle.

Referências

[1] Bishop, Christopher M. (2006) Pattern Recognition and Machine Learning.

[2] Brownlee, J. (2017) *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning.* **machinelearningmastery.com** Disponível em: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. Acesso em: 19 nov. 2018.

[3] Bryson, A.E., Dynamic Optimization, Addison-Wesley, 1999.

[4] Haykin, S. Neural Networks: A Comprehensive Foundation, Prentice Hall, New Jersey, 1999.

[5] Fumo, D. (2017) *A Gentle Introduction To Neural Networks Series - Part 1* **towardsdatascience.com** Disponível em: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc. Acesso em: 03 dez. 2018.

[6] Karn, U. "*A Quick Introduction to Neural Networks*." **/ujjwalkarn.me** Disponível em: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/. Acesso em: 08 dez. 2018.

[7] Kingma, D. Ba, J. (2014) "*ADAM: A Method for Stochastic Optimization*". *(3^{rd}* International Conference on Learning Representations).

[8] Kurdekar, V. Borkar, S. (2013) "***Inverted Pendulum Control: A Brief Overview***" *International Journal of Modern Engineering Research* 13:2924-2927

[9] OpenAI. *Getting Started with Gym*. **gym.openai.com** Disponível em: https://gym.openai.com/docs/. Acesso em: 26 nov. 2018.

[10] Ramachandran, Prajit; Barret, Zoph; Quoc, V. Le. (2017) Searching for Activation Functions.

[11] Sambatti, S. B. M. Furtado, H. Anochi, J. Luz, E. F. P. Campos Velho, H. F. (2012) "*Automatic configuration for neural network with application to the data assimilation*". *IMSE: Integral Methods in Science and Engineering*.

[12] Suryansh S. "*Gradient Descent: All You Need to Know*". **https://hackernoon.com**. Disponível em: https://hackernoon.com/gradient-descent-aynk-7cbe95a778da. Acesso em: 09 dez. 2018.

[13] TensorFlow. *An open source machine learning library for research and production.* **tensorflow.org** Disponível em: https://www.tensorflow.org/?hl=pt-br. Acesso em: 26 nov. 2018.

[14] Živković, N. "*Backpropagation Algorithm in Artificial Neural Networks*" **rubikscode.net**. Disponível em: https://rubikscode.net/2018/01/22/backpropagation-algorithm-in-artificial-neural-networks/. Acesso em: 19 nov. 2018.