Exercício-Programa 2: Previsão da Fase de Infecção por COVID-19

MAC0425 - Inteligência Artificial

Juliana Namie Yamamoto 11314971

23 de junho de 2025

Resumo

Este trabalho desenvolve um modelo de rede neural para classificação do estágio de infecção por COVID-19 com base no repositório de exames de pacientes do Hospital das Clínicas (`https://repositoriodatasharingfapesp.uspdigital.usp.br/`). Utilizando dados de índices IgG, implementamos uma arquitetura Multilayer Perceptron (MLP) que alcançou 98.68% de acurácia na previsão de três categorias: não reagente (0), inconclusivo (0.5) e reagente (1).

1 Introdução

A pandemia de COVID-19 evidenciou a necessidade de métodos rápidos e precisos para classificação da fase infecciosa. Testes sorológicos geram índices contínuos cuja interpretação manual pode variar entre profissionais. Automatizar essa análise com redes neurais oferece vantagens em consistência e escalabilidade.

Os principais objetivos deste projeto é desenvolver um classificador automático que:

- Interprete valores de índice sorológico
- Classifique em três categorias (0=não reagente, 0.5=inconclusivo, 1=reagente)
- Alcance uma porcentagem de acurácia aceitável

Este relatório apresenta a metodologia e, em seguida uma discussão sobre os resultados obtidos pela rede neural.

2 Metodologia

Os dados utilizados neste estudo foram originalmente extraídos do sistema hospitalar do Hospital das Clínicas (HC-FMUSP) e organizados em uma tabela estruturada por meio de consultas em SQL. Essa etapa prévia permitiu a limpeza, filtragem e formatação dos dados relevantes — em especial os valores dos índices IgG e os respectivos rótulos de classificação sorológica — de forma a facilitar o carregamento posterior via `pandas` no ambiente Python.

2.1 Pré-processamento dos Dados

- **Amostra:** 759 pacientes (489 reagentes, 12 inconclusivos, 258 não reagentes)
- **Features:** Índice sorológico de pacientes que fizeram o exame "COVID-19 - PESQUISA DE ANTICORPOS IgG"
- **Split:** 80% treino, 20% teste

```python
import pandas as pd
from sklearn.model_selection import train_test_split

dataset = pd.read_csv("HC_TABELA_SEM.csv", sep='|')
dataset['RESULTADO'] = dataset['RESULTADO'].map({
    'Reagente': 1,
    'N o reagente': 0,
    'Inconclusivo': 0.5
})

x = dataset[['INDICE']].values
y = dataset['RESULTADO'].values

x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2
)
```

Listing 1: Leitura e processamento dos dados

Intanciando Custo e Otimizador:

```
device = torch.device("cuda" if torch.cuda.is_available()
    else "cpu")
model = COVID_predictor().to(device)
criterion = nn.BCELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=
    learning_rate)
```

Listing 2: Custo e Otimizador

2.2 Arquitetura da Rede Neural

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class COVID_predictor(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(1, 16)
        self.fc2 = nn.Linear(16, 16)
        self.output = nn.Linear(16, 1)
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = F.relu(self.fc2(x))
        return torch.sigmoid(self.output(x))
```

Listing 3: Classe da rede neural em PyTorch

A arquitetura da rede neural implementada segue o modelo MLP, utilizando a biblioteca `PyTorch`. A rede possui uma camada de entrada com um único neurônio correspondente ao índice sorológico, seguida por duas camadas ocultas com 16 neurônios cada e função de ativação ReLU. Após a primeira camada oculta, é aplicada uma camada de dropout com taxa de 20% para regularização e prevenção de overfitting. A camada de saída contém um único neurônio com ativação sigmoide, adequada à previsão de probabilidades para uma tarefa de classificação binária ou contínua em [0,1].

2.3 Experimentos

Os experimentos consistiram no treinamento do modelo por 100 épocas com aprendizado supervisionado, utilizando a função de perda Binary Cross Entropy (BCELoss) e o otimizador Adam com taxa de aprendizado de 0.01. A avaliação do desempenho do modelo foi realizada por meio das métricas de acurácia e valor da função de perda, monitoradas tanto no conjunto de treino quanto no de teste. Além disso, foram registradas as predições do modelo e comparadas com os rótulos verdadeiros para análise qualitativa dos resultados, especialmente em casos intermediários entre as classes.

3 Resultados

Durante o treinamento da rede neural, observou-se uma melhoria progressiva no desempenho do modelo, conforme indicado pelos valores de perda e acurácia ao longo das épocas. A tabela a seguir resume os resultados obtidos no conjunto de teste para 10 épocas de treinamento.

Epoch	**Test Loss**	**Test Accuracy**
1	0.3358	0.6184
2	0.3222	0.8618
3	0.3089	0.9408
4	0.2955	0.9737
5	0.2821	0.9737
6	0.2687	0.9737
7	0.2554	0.9868
8	0.2426	0.9868
9	0.2300	0.9868
10	0.2174	0.9868

Tabela 1: Evolução da perda e acurácia no conjunto de teste ao longo de 10 épocas.

Ao final da décima época, o modelo alcançou uma acurácia de 98,68% com uma perda de 0,2174 no conjunto de teste, indicando um desempenho consistente e estável após a convergência.

Observa-se que a função de perda diminui progressivamente ao longo das épocas, como é esperado em treinamentos bem-sucedidos. Com mais épocas,

é provável que a perda continue reduzindo, refletindo um aprendizado ainda mais refinado do modelo.

Para avaliar a capacidade preditiva do modelo de forma qualitativa, a tabela abaixo apresenta um subconjunto das predições realizadas, comparando os valores dos índices de entrada, as saídas da rede e os rótulos verdadeiros.

Índice	**Predição**	**Resultado**
0.20	0.00	0.00
3.30	1.00	1.00
0.20	0.00	0.00
12.60	1.00	1.00
0.30	0.00	0.00
8.30	1.00	1.00
10.40	1.00	1.00
0.20	0.00	0.00
10.00	1.00	1.00
0.20	0.00	0.00

Tabela 2: Exemplos de predições do modelo no conjunto de teste.

O modelo demonstrou acerto em todos os casos ilustrados, destacando sua eficácia na separação entre os índices classificados como reagentes e não reagentes. Os bons resultados obtidos sugerem que a rede foi capaz de aprender uma fronteira de decisão adequada mesmo com um único preditor contínuo.

4 Discussão

Os resultados apresentam a rede neural como uma proposta que é viável para prever a fase da infecção por COVID-19 a partir do índice IgG. Com alta acurácia no teste, o modelo simples demonstrou ser eficaz na classificação, principalmente entre casos claramente reagentes e não reagentes.

Na prática, essa rede pode automatizar a triagem sorológica, auxiliando profissionais de saúde em decisões rápidas, especialmente em contextos com grande volume de exames. Contudo, para uso profissional, é necessária uma validação adicional e a integração com outros indicadores clínicos. Mesmo assim, o modelo apresenta potencial como ferramenta auxiliar para agilizar diagnósticos e reduzir interpretações subjetivas.

5 Bibliografia

Referências

[1] B., Arunachalam (2023). *Binary Classification with TensorFlow Tutorial*. Disponível em: [https://www.freecodecamp.org/news/binary-classification-made-simple-with-tensorflow/]

[2] Ghulam Jillani, Muhammad (2024). *Building an ANN with PyTorch: A Deep Dive into Neural Network Training* . Disponível em: [https://jillanisofttech.medium.com/building-an-ann-with-pytorch-a-deep-dive-into-neural-network-training-a7fdaa047d81]

[3] GeeksForGeeks (2025). *How to implement neural networks in PyTorch?* . Disponível em: [https://www.geeksforgeeks.org/deep-learning/how-to-implement-neural-networks-in-pytorch/]