MAC0425/5739 - Inteligência Artificial

# Relatório do Exercício-Programa 4: Previsão de Estágio da Infecção por COVID-19

Andrew Ijano Lopes
Nº USP: 10297797

**Resumo:** A COVID-19 é uma doença viral, que pode causar quadros respiratórios agudos graves, e possui três fases detectáveis através de exames para PCR (proteína C-reativa) e anticorpos IgM e IgG. Entretanto, esses testes podem ter correlação com outros exames laboratoriais. Por isso, o objetivo desse projeto é analisar bases de dados públicos sobre exames de pacientes e construir redes neurais que, dado um conjunto de exames, preveja o resultado de PCR, IgM e IgG. Efetuou-se o pré-processamento dos exames do Hospital Israelita Albert Einstein alimentando uma rede neural Feedforward de duas camadas ocultas. Através de um processo de Validação Cruzada com $k$-folds, as redes obtiveram uma acurácia de previsão de 68% para PCR e 59% para IgM e IgG. Com um desempenho 18% acima da *baseline* para o PCR, esse é um procedimento que precisa de mais atenção antes de ser aplicado.
**Palavras-chave:** COVID-19, redes neurais.

São Paulo, 20 de Julho de 2020

# 1 Introdução

A COVID-19 é uma doença causada pelo vírus SARS-CoV-2 e que podem levar, em alguns casos, a um quadro respiratório agudo grave. A infecção por esse vírus possui três fases detectáveis. Na primeira fase o paciente está infectado mas a reação de seu sistema imunológico pode ainda não ter-se iniciada, o que é detectado pelo exame PCR com resultado positivo. Na segunda fase, o corpo do paciente já começou a reagir e a produzir anticorpos do tipo IgM, o que é detectado por um exame que verifica a presença deste tipo de anticorpo. Na terceira fase, o paciente está curado da infecção e o corpo já produziu anticorpos do tipo IgG que conferem imunidade ao paciente para uma nova infecção, o que é detectado pela presença desse tipo de anticorpo.

Com o objetivo de contribuir para o desenvolvimento de pesquisas de combate a essa doença, foram disponibilizados dados públicos e anônimos sobre três hospitais da cidade de São Paulo que tratam esse tipo de infecção.(FAPESP, 2020) Nessa base de dados, existem exames de diversas naturezas e, dentre eles, existem exames de PCR e para detecção de anticorpos IgM e IgG.

Por isso, o objetivo desse trabalho é analisar a base de dados sobre os pacientes de um dos hospitais e construir redes neurais que, dado um conjunto de exames de um paciente em uma determinada data, preveja o resultado dos exames PCR e detecção de IgM e IgG. Dessa forma, seria possível prever o estado da infecção de um paciente.

Neste relatório, são descritos os métodos para o desenvolvimento do projeto, composto pelo pré-processamento dos dados, a arquitetura da rede neural e a descrição dos experimentos realizados; em seguida são apresentados os resultados obtidos e finaliza-se com uma discussão da viabilidade do modelo obtido.

# 2 Metodologia

Nesta seção, serão descritos os métodos de pré-processamento dos dados, a arquitetura da rede neural utilizada e a descrição dos experimentos realizados.

## 2.1 Pré-processamento dos dados

Para o desenvolvimento deste projeto, foram utilizados apenas dados de exames do Hospital Israelita Albert Einstein, dentre os três hospitais do banco de

dados compartilhado. Dos dados desse hospital, utilizou-se apenas a tabela de exames, identificada como `einstein_full_dataset_exames.csv`.

O início do pré-processamento, referente ao *script* salvo em `PRE/preprocess_einstein.py`, consistiu da remoção de linhas duplicadas, frutos, possivelmente, de erros de cadastro. Em seguida, os valores das colunas `de_analito`, `de_resultado` e `de_valor_referencia` foram convertidas para *string*s em minúsculo, para facilitar a sua manipulação.

Depois, foram feitas substituíções nos valores da base. Primeiro, os analitos referentes aos três exames de COVID-19 alvos desse projeto foram renomeados para `pcr-result`, `igm-result` e `igg-result`. Além disso, termos como "não reagente", "não detectado", "ausente" e "ausentes" foram substituídos por 0 e seus antônimos por 1. Ainda, valores categóricos sequenciais como "+", "++" e "+++"' foram substituídos por 1, 2 e 3, respectivamente.

Isso foi seguido pela remoção de linhas com analitos irrelevantes. Por exemplo, linhas com o analito "*leucócitos*" foram removidas pois o mesmo resultado pode ser obtido pelo seu valor numérico nas linhas com o analito "*leucócitos #*".

Com isso, foi criado uma nova tabela, onde cada linha representa os resultados de todos os exames sobre um analito que um paciente realizou numa determinada data. Essa tabela possui 103 colunas, cujas três primeiras correspondem aos resultados dos exames para PCR, IgM e IgG, e as seguintes correspondem ao resultado dos 100 analitos restantes mais comuns na base. Os valores numéricos das células são convertidos para *float* e os valores categóricos, não convertidos para números anteriormente, são substituídos por 0 caso sejam iguais ao valor de referência do exame e 1, caso contrário.

| PCR | IgM | IgG | $\text{analito}_1$ | $\text{analito}_2$ | $\cdots$ | $\text{analito}_{100}$ |
|---|---|---|---|---|---|---|

Figura 1: Cabeçalho da nova tabela gerada

Dessa tabela, são removidas todas as linhas que não possuem exames de PCR, IgM nem IgG. E são removidas também todas as linhas que possuem menos que quatro exames.

Assim, essa é a tabela pré-processada salva em `PRE/einstein.out.csv`.

Já a segunda parte do pré-processamento foi feita de forma individual para o treinamento das redes neurais, que farão previsão de um dos exames (PCR, IgM ou IgG). Nesse pré-processamento, todas as linhas que não possuam o exame específico são removidas e os campos restantes que estiverem vazios são substituídos pela mediana dos valores da coluna. Em seguida, a tabela é embaralhada nas linhas e dividida no vetor *target* $y$, que é a coluna do respectivo exame, e a matriz de *features* $X$, que são as 100 últimas colunas

da tabela. A matriz de *features* $X$ é, então, normalizada pelas colunas de acordo com:

$$\frac{valor - minColuna}{maxColuna - minColuna + 1}$$

O conjunto de dados $X$ e $y$ serão usados, posteriormente, para o treinamento da rede neural.

## 2.2 Arquitetura da rede neural

A rede neural deste projeto, construída com a biblioteca PyTorch (Paszke et al., 2017), foi projetada para prever o resultado de um exame (PCR, IgM ou IgG) dados os resultados dos exames nos 100 analitos selecionados na seção anterior. Ela consiste em uma rede Feedforward com duas camadas escondidas, de tamanhos 5 e 3, respectivamente, e com tamanho de entrada 100. As duas camadas escondidas utilizam a função de ativação ReLU, enquanto a camada de saída utiliza uma Sigmoid. No treinamento, a função de perda usada foi a *BCELoss*, que mede a *Binary Cross Entropy* entre o *target* e *output*, e para otimização usou-se o algoritmo Adam (Kingma and Ba, 2014).

O treinamento do modelo final é feito com todo *dataset*, cujos vetores *target* $y$ e matrizes $X$ de *features* são dados por:

| Rede Neural | tamanho de $X$ | tamanho de $y$ |
|---|---|---|
| PCR | (9634, 100) | 9634 |
| IgM | (1264, 100) | 1264 |
| IgG | (1264, 100) | 1264 |

Figura 2: Tamanhos do *dataset* por rede neural

As instruções para execução do código estão descritas em `NN/README.md`.

## 2.3 Descrição dos experimentos

Para cada uma das redes neurais, o treinamento consiste numa repetição de 1000 épocas. Em cada época, é realizado uma *forward propagation*, calculando o custo pela função de custo; em seguida, os gradientes do modelo são zerados, é efetuada a *backpropagation* e o otimizador atualiza seus parêmetros.

Para validar o modelo, foi realizado um processo de validação cruzada com $k$-folds, para $k = 10$. Nesse processo, realizado com auxílio do método

*KFold* do *scikit-learn* (Pedregosa et al., 2011), o *dataset* $(X, y)$ é divido em 10 partes. Em cada uma das 10 iterações do algoritmo, uma das partes se torna o conjunto de testes $(X\_test, y\_test)$ e o restante, o conjunto de treinamento $(X\_train, y\_train)$. Além disso, para balencear as classes, cada um desses conjuntos passa também por um processo de *Oversampling*, auxiliado pelo método *RandomOverSampler* da biblioteca *imbalanced-learn* (Lemaître et al., 2017), que duplica aleatoriamente amostras da classe menos frequente.

Em cada uma dessas iterações, o modelo é trainado com o conjunto de treinamento. No fim do treinamento, o modelo realiza uma previsão $y_{pred}$ no conjunto de testes $(X_{test}, y_{test})$e, com isso, sua acurácia é calculada por

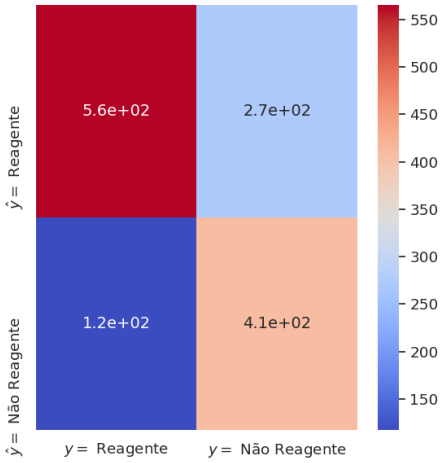
$$\frac{\texttt{número de casos que } y_{pred\,i} = y_{test\,i}}{\texttt{tamanho de } y_{test}}$$

# 3 Resultados

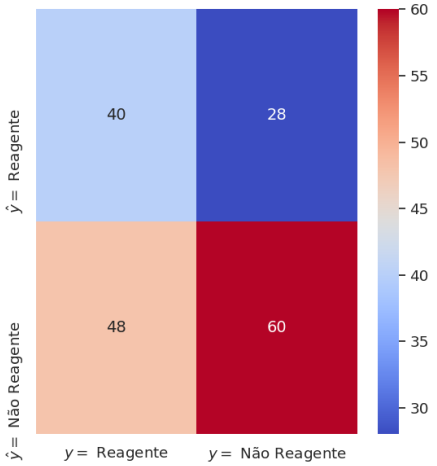
Através do processo de validação cruzada por *k-folds*, mencionado anteriormente, obtemos os seguintes resultados:

| | Acurácia | | |
|---|---|---|---|
| Repetição | PCR | IgM | IgG |
| 1 | 0.644 | 0.568 | 0.639 |
| 2 | 0.715 | 0.542 | 0.604 |
| 3 | 0.701 | 0.600 | 0.500 |
| 4 | 0.500 | 0.570 | 0.613 |
| 5 | 0.738 | 0.653 | 0.670 |
| 6 | 0.704 | 0.542 | 0.549 |
| 7 | 0.723 | 0.649 | 0.500 |
| 8 | 0.682 | 0.640 | 0.678 |
| 9 | 0.736 | 0.625 | 0.610 |
| 10 | 0.716 | 0.566 | 0.577 |
| Medidas | PCR | IgM | IgG |
| Média | 0.686 | 0.595 | 0.594 |
| Desv. Pad. | 0.071 | 0.044 | 0.063 |
| Maior | 0.738 | 0.653 | 0.678 |
| Menor | 0.500 | 0.542 | 0.500 |

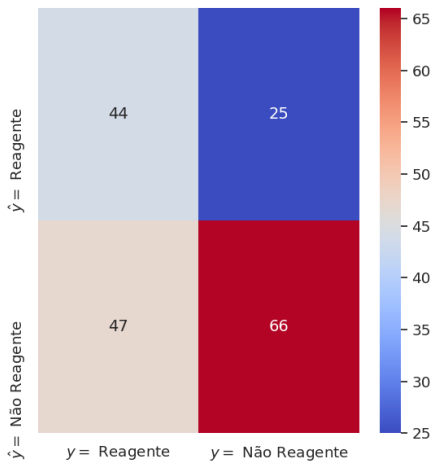
Figura 3: Resultados de acurácia no processo de validação cruzada




(a) PCR




(b) IgM




(c) IgG

Figura 4: Matrizes de confusão por rede

# 4 Discussão

Devido ao balanceamento realizado dos dados, o *baseline* para uma previsão é de 50%. Por isso, uma acurácia de 68% para PCR e 59% para IgM e IgG já mostra algum grau de aprendizado. Entretanto, ainda é um desempenho muito baixo para a rede ser usada na prática.

Além disso, é necessário um estudo mais aprofundado sobre os métodos utilizados para identificar possíveis viéses durante o treinamento e teste, que poderiam gerar falsas conclusões sobre o desempenho do algoritmo.

# Referências


FAPESP (2020). COVID-19 data sharing/BR. https://repositoriodatasharingfapesp.uspdigital.usp.br/. Accessado em 19/07/2020.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lemaître, G., Nogueira, F., and Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.