Previsão de estágio da infecção por COVID-19

MAC0425 Inteligencia artificial - EP04

Carolina Senra Marques - 10737101
Felipe Castro de Noronha - 10737032

Resumo

Neste trabalho exploramos o uso de redes neurais para a previsão de resultados de exames acerca da COVID-19. Atualmente, três principais exames são usados para detectar o virus no corpo de um paciente, são eles: PCR, que detecta a presença do material genético do virus no organismo; igM, que detecta se o organismo teve contato com o virus recentemente; igG, que detecta se o organismo teve contato com o virus previamente [1]. Utilizando 196 exames como *features* treinamos 3 redes neurais que prevem o resultado de cada um dos exames acima.

Introdução

A seguir, falaremos da motivação e objetivos deste trabalho, assim como a estrutura deste relatório e dos arquivos.

Motivação

As redes neurais neurais estão se tornando cada vez mais populares, sendo utilizadas nos mais variados campos, desde visão computacional[2] até a astronomia [3]. Por que não tentar utilizar essa tecnologia para ajudar na questão de saúde publica que vivemos atualmente?

Alem disso, esta tarefa se mostra uma excelente oportunidade para colocarmos em pratica os conhecimentos obtidos na disciplina MAC0425 - Inteligencia artificial.

Objetivos

Esperamos criar um conjunto de redes neurais com acurácia minimamente satisfatória e que, a partir de um conjunto de exames realizados (não relacionados a COVID-19) possam prever o resultado dos exames de igG, igM e PCR de um determinado paciente.

Estrutura deste relatório

O restante do relatório é dividido em 4 partes, são elas:

- Metodologia, onde explicamos como foi realizado o pré-processamento dos dados, a arquitetura da rede neural e a descrição dos experimentos realizados para decisão do modelo.
- Resultados, onde apresentamos a qualidade obtida pela nossa rede.
- Discussão, onde expomos os possíveis usos da rede.
- Bibliografia, fontes das citações aqui presentes.

Estrutura dos diretórios

O diretório principal deste trabalho contem 3 subdiretórios, são eles:

- `NN/`: Contem programas relacionados exclusivamente ao treinamento, uso e avaliação da rede neural. Além disso, os artefatos da rede gerada ficam armazenados em `/trained_nn/`.
- `PRE/`: Contem as planilhas brutas de dados, além disso, os scripts responsáveis por realizar a preparação inicial ficam em `/scripts/`.
- `REL/`: Contem o relatório.

Metodologia

Aqui descrevemos o pré-processamento dos dados, a arquitetura da rede neural e fazemos a descrição dos experimentos.

Pré-processamento

Para o treinamento de uma boa rede neural é necessário um bom numero de dados consistentes. Por isso, utilizamos os *datasets* disponibilizados pelo Grupo Fleury [4] (mais de 120000 pacientes) e pelo Hospital Sírio-Libanês [5] (mais 2700 pacientes). Ambos *datasets* se resumem a dois arquivos, planilhas que contem informações dos pacientes e dos resultados de vários exames que os pacientes realizam.

Primeiramente, corrigimos os caracteres que estavam errados por causa da conversão de formatação, usando o *script* `fix_utf.py` e um pouco da ajuda de nosso editor de texto. Com o *script* `get_unique.py` pegamos todas as entradas únicas das colunas *DE_ANALITO* da tabela de exames do Fleury e do HSL e que estão presentes em ambas tabelas, a lista completa destes exames utilizados pode ser vista em `used_analitos.csv`. Em seguida tivemos que refinar esta lista de exames usados, retirando da lista exames que tivesses resultados de difícil mapeamento numérico e também todos os exames como frequência de resultado menores que 1000.

O *script* `generate.py` é responsável por criar o *dataset* final a ser utilizado. Nele é criado um dicionario que aceita como chave os *ids* relacionados a cada paciente, assim, cada entrada retorna uma lista que corresponde aos exames realizados por um dado paciente. Para lidar com o fato de que um paciente pode realizar o mesmo exame diversas vezes em um dado período de tempo usamos o conceito de *batch*, em particular, processamos todos os exames realizados em um certo dia, atualizando as respectivas colunas de um dado paciente, finalmente, após o processamento de um dia, todos os pacientes que tiveram algum exame modificado tem sua lista de exames escrita no arquivo `data.csv`.

No *dataset* final `data.csv`, elementos com valores igual a zero significam resultados de exames que não estavam disponíveis na base de dados. Alem disso, logo antes dos dados serem usados pelas redes, normalizamos os dados, colocando-os no intervalo [0,1], e fazemos *over-sampling* para garantir que as classes fiquem balanceadas, isto é, termos o mesmo numero de pacientes com resultados positivos e negativos para um dos 3 exames principais. Essa fase final de adequação é feita pelo modulo `data.py`.

Por ultimo, dois outros *scripts* foram responsáveis por nos ajudar no pré-processamento, são eles o *script* `get_histogram.py` que nos informa a frequência de cada exame no *dataset* final e o *script* `get_reduced.py` que cria uma instancia reduzida do dataset final, o que ajudou a depurar os códigos acerca da rede neural.

Arquitetura da rede neural

Como os 3 exames possuem resultados independentes, optamos por utilizar 3 redes distintas, assim, cada uma ficaria responsável por prever um tipo de exame. Além disso, cada rede recebe como entrada 196 *features*, que são os resultados dos exames listados em `used_analitos.csv`.

A rede neural utilizada para os 3 casos possuia 3 camadas ocultas, com tamanhos 1024, 512 e 256, respectivamente. Optamos por usar `0.001` como taxa de aprendizado, esta taxa, apesar de pequena, se mostrou mais eficiente em comparação com outros valores. Além disso, utilizamos o otimizador Adam, que lida muito bem com a convergência quando temos um grande *dataset*. Finalmente, usamos a *binary cross entropy loss function* (BCELoss), pois é uma ótima função de custo para nosso caso de 3 redes distintas.

Descrição dos experimentos

Para definir a melhor arquitetura para a rede neural optamos por testar 3 modelos diferentes. Cada um dos modelos possuem 3 camadas ocultas e foi chamado de NNX. As camadas de NN1 tem tamanho `512, 256, 128`, as de NN2 `1024, 512, 256` e as de NN3 `256, 128, 64`. Os experimentos para decidir qual rede usar foram feitos a partir no arquivo `experiment.py`. Nele são testadas 3 redes neurais usando *stratified k-fold cross validation* com k = 5 (para diminuir um pouco o longo tempo de execução, que foi para aproximadamente 2 horas e 30 minutos na máquina 'neozil' da Rede Linux). Abaixo temos a acurácia media de cada arquitetura para cada exame nos *cross validation*:

	igG	**igM**	**PCR**
Acurácia modelo NN1	Validação: 65% Treinamento: 66%	Validação: 67% Treinamento: 67%	Validação: 68% Treinamento: 72%
Acurácia modelo NN2	Validação: 66% Treinamento: 67%	Validação: 68% Treinamento: 68%	Validação: 68% Treinamento: 72%
Acurácia modelo NN3	Validação: 65% Treinamento: 67%	Validação: 66% Treinamento: 67%	Validação: 68% Treinamento: 71%

A partir dos dados obtidos com os testes foi possível concluir que a melhor rede é a NN2.

Resultados

O resultado de acurácia por *fold* e por rede/exame está em `NN/result_experiment.txt`. Por eles é possível notar que a melhor rede é a NN2 com acurácia no grupo de validação de 66.86% ,68.12%, 68.56% no igG, igM e PCR respectivamente.

Assim, finalmente, a rede NN2 foi treinada com um dataset completo, usando 10% do mesmo como conjunto de validação. O relatório completo do treinamento da rede final pode ser visto no arquivo `NN/result_main.txt`. Em suma, as redes obterem as seguintes acurácias:

	igG	**igM**	**PCR**
Acurácia no conjunto de validação	66%	68%	69%
Acurácia no conjunto de testes	68%	68%	72%

Além disso, temos o utilitário `use.py`, que, através da rede treinada realiza a predição para novos dados, veja o arquivo para mais detalhes. Ele utiliza `min_max.csv` que contém o menor e o maior valor para cada

exame utilizado para que se possa normalizar os dados fornecidos da pessoa.

Discussão

Acreditamos que é difícil mensurarmos a qualidade real da rede neural criada e **não recomendamos, de forma alguma, o seu uso para o diagnostico de pacientes**. Porém, apreciamos muito a oportunidade de trabalhar com dois tópicos tão atuais e relevantes, apredemos muito sobre a COVID-19 e as ferramentas existentes para sua detecção, além disso, o uso de *frameworks* como o PyTorch se mostrou incrivelmente proveitoso.

Bibliografia

- [1] "Testes para Covid-19: perguntas e respostas". ANVISA. Disponível em: http://portal.anvisa.gov.br/documents/219201/4340788/Perguntas+e+respostas+-+testes+para+Covid-19.pdf. Acesso em: 07 de Julho de 2020.
- [2] LEE, Yuchun. "Handwritten Digit Recognition Using K Nearest-Neighbor, Radial-Basis Function, and Backpropagation Neural Networks". Disponível em: https://www.mitpressjournals.org/doi/abs/10.1162/neco.1991.3.3.440. Acesso em: 05 de Julho de 2020.
- [3] NUNES, Jorge; LLACER, Jorge. "Astronomical image segmentation by self-organizing neural networks and wavelets". Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S089360800300011X. Acesso em: 05 de Julho de 2020.
- [4] "Dados COVID Grupo Fleury". FLEURY, Grupo. Disponível em: https://repositoriodatasharingfapesp.uspdigital.usp.br/handle/item/99. Acesso em: 01 de Julho de 2020.
- [5] "Dados COVID Hospital Sírio-Libanês". SIRIO-LIBANES, Hospital. Disponível em: https://repositoriodatasharingfapesp.uspdigital.usp.br/handle/item/97. Acesso em: 01 de Julho de 2020.