Conhecimento e Raciocínio

2020 - 2021

Redes Neuronais

Identificação de Caracteres Gregos

Ângelo Paiva - 2019129023
José Almeida - 2019129077

Licenciatura de Engenharia Informática
26 de junho de 2021

Conteúdo

1 Introdução

Este trabalho, feito no âmbito da Unidade Curricular de Conhecimento e Raciocínio, tem como objetivo o desenvolvimento e estudo estatístico de redes neuronais para identificação de alguns caracteres gregos. A linguagem e tecnologia usada para esse efeito é o MATLAB. Adicionalmente, o trabalho prático inclui uma aplicação gráfica que permite criar, treinar e simular redes.

2 Conversão e Tratamento das Imagens: Inputs e Targets

Seguindo as indicações sobre arquitetura do projeto, foram adicionados diversos packages de modo a estruturar logicamente o código. Estão organizados da seguinte forma:

O dataset usado no trabalho é constituído por 4 pastas. Destas 4, uma foi criada por nós, com 2 caracteres gregos de cada um dos 10 que devem ser identificáveis pelas redes. As outras 3 foram fornecidas junto com o enunciado, e não tiveram qualquer tratamento prévio fora do MATLAB.

De modo a diminuir os custos de tempo e memória de treino, as imagens, ao serem lidas pelo programa, são redimensionadas para 28x28 (de 3024x3024). De seguida, são colocadas em matrizes binárias e convertidas em uma única coluna, pois, no MATLAB, as redes neuronais recebem os seus *inputs* na forma de colunas de uma matriz.

As imagens, no nosso trabalho, devem ser lidas por ordem alfabética, e em grupos de 10 caracteres diferentes, ou seja, todos os caracteres, para efeitos de treino, devem estar presentes em igual número e ordenados alfabeticamente.

Relativamente ao tratamento dos *targets* da rede neuronal, estes são criados usando matrizes de identidade 10 por 10. Cada coluna da matriz de targets terá 10 linhas, ou seja, uma para cada carater, preenchida com um valor que deve ser 0 ou 1: 1 caso a imagem correspondente contenha o carater que a linha representa, ou 0, em caso contrário.

Analisando, no momento de produção deste relatório, o tratamento feito às imagens, concluímos que devíamos ter aplicado outras técnicas que teriam um impacto positivo no desempenho da rede, tal como fazer o *trim* das imagens, ou seja, remover os espaços em branco à volta das mesmas, visando uniformizar os tamanhos dos caracteres fornecidos à rede.

3 Pastas Individuais

3.1 Pasta 1

Iniciando-se então o processo de desenvolvimento e estudo da rede neuronal, foram testadas várias topologias, funções de ativação e de treino. Todos os valores obtidos foram colocados no excel em anexo.

Comparado com a configuração base de uma camada escondida com 10 neurónios, notamos um impacto positivo, em geral, quando é aumentado o número de camadas escondidas, assim como o número de neurónios por camada, tendo este último um impacto mais significativo no desempenho da rede.

Relativamente ao impacto da função de treino no desempenho da rede, observamos que, pelo menos nos nossos testes, nenhuma das outras funções de treino, neste caso, alcançaram melhores valores do que a configuração base. No entanto, grande parte das mesmas reduz muito a complexidade temporal necessária ao treino.

Por fim, foram testadas várias configurações possíveis para funções de ativação. A partir destes testes, concluímos que usar a função de ativação **purelin** na camada de output tem melhores resultados.

No entanto, os testes efetuados na pasta 1 não são muito significativos, pois até mesmo a configuração base escolhida obtém valores ótimos de desempenho. Isto deve-se ao facto de que as redes, nestes testes, estão a ter acesso e a usar para treino todos os exemplos da pasta. Nesta pasta existe apenas um exemplo por carater gregp possível, tornando-se impossível não usar 100% das imagens para treino.

3.2 Pasta 2

Nesta pasta, a matriz de targets deve ser invertida, pois as imagens encontram-se pela ordem inversa à alfabética, e poderia causar incompatibilidades com as outras pastas caso a mesma não fosse invertida.

Testando, de forma semelhante à pasta 1, o impacto do número e dimensão das camadas escondidas, chegamos aos mesmos resultados: mais camadas e neurónios são benéficos para a rede, pelo menos até certo ponto. No entanto, o aumento do número de camadas e neurónios tem um elevado custo temporal e de memória associado, facto que foi tido em conta ao longo de todo o estudo estatístico.

O mesmo aconteceu quanto ao estudo sobre o desempenho das várias funções de treino: nenhuma delas superou a **trainlm**, presente na configuração base.

Foi testado, de igual forma, o impacto das funções de ativação no desempenho da rede. Obtivemos aqui uma das melhores redes, que reportou um desempenho de **1.8139E-29** (neste caso, quanto mais baixo melhor). No entanto, esta rede, cujas funções de ativação foram ambas configuradas com **purelin**, não obteve os melhores resultados em termos de desempenho de validação e de teste. Mesmo assim, decidimos guardá-la e testá-la um pouco mais, posteriormente.

Devido ao maior tamanho do dataset disponível na pasta 2, já é possível, aqui, dividir os exemplos pelos três conjuntos (de treino, de validação e de teste). De um modo geral, quanto maior for o número de exemplos para treino disponível à rede, melhor o seu desempenho. No entanto, devem existir pelo menos alguns exemplos para validação e teste, pois estes permitem um melhor estudo estatístico das redes: tornam possível a avaliação da rede face a exemplos que não usou durante o treino. A existência de elementos nestes conjuntos permitem à rede autoavaliar-se e, se achar conveniente, terminar mais cedo o treino, o que faz com que a complexidade temporal de treino das redes seja mais reduzido, embora com alguns impactos no desempenho da mesma.

Para finalizar o teste dos diferentes parâmetros da rede, fizemos variar a função de divisão dos conjuntos, onde notamos um melhor desempenho aquando do uso da função **divideblock**. Pensamos que isso se deva à forma como os nossos *inputs* são lidos e fornecidos à rede, em blocos de 10 caracteres.

Por fim, decidimos juntar os melhores parâmetros de cada teste, de modo a analisar se o melhoramento individual dos parâmetros tem um impacto positivo no desempenho da rede. Foi aqui que atingimos os melhores valores de desempenho até então vistos, tendo sidas destacadas 4 redes, com resultados bastante satifatórios em todos os indicadores que considerámos. Todas as 5 redes escolhidas partilham a mesma função de ativação **purelin** para a camada de output, sendo que apenas 1 delas não utiliza a função de divisão **divideblock**.

3.3 Pasta 3

4 Testes Gerais

5 Anexos

Lista de Figuras