notMinist Data Set

Inteligência Computacional

2021/2022

Projeto - Fase I

Índice

• Cap. 1: Descrição do caso de estudo e objetivos do problema;

• Cap. 2: Descrição da Implementação dos algoritmos;

• Cap. 3: Análise de Resultados;

• Cap. 4: Conclusões;

• Referências

# Cap. 1: Descrição do caso de estudo e objetivos do problema

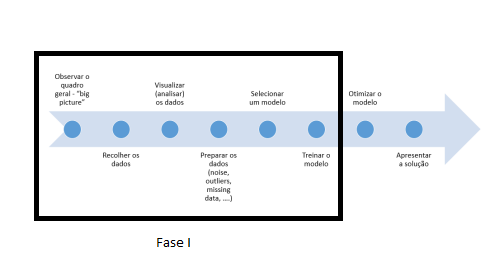


Fig. 1 – Etapas de um processo de Machine learning

Quanto à observação do quadro geral “big picture”.

O caso de estudo em questão tem por um base o notMnist Data set, um dataset que é uma alternativa ao dataset tradicional Mnist para reconhecimento das letras do alfabeto, sendo mais complexo.

Este mesmo dataset foi desenvolvido por Yaroslav Bulatov.

Sendo que para o trabalho usamos a sua versão mais curta que contém as suas especificações:

o Número de exemplos: 18724

o Número de atributos: Imagem 28x28 (754)

o Número de classes: 10 classes (letras A-J)

Este dataset é nos disponibilizado através de um diretório raiz chamado notMNIST\_small que contem várias pastas com o nome de cada letra sendo que em cada uma destas pastas se encontram os exemplos de cada tipo de letra correspondente.

# Cap. 2: Descrição da Implementação dos algoritmos

Fizemos a implementação deste algoritmo em matlab.

Para isso tivemos que usar uma função em python que convertia o dataset inicialmente disponibilizado para um ficheiro .mat, sendo assim possível a sua utilização em matlab. Esta função python encontra-se no ficheiro de submissão.

A arquitetura da rede neuronal utilizada foi **MLP**

**MLP-Redes de múltipla camada**

As redes de múltipla camada resolvem problemas não linearmente separáveis. Este algoritmo permite modificar os pesos de qualquer rede de modo que, sendo-lhe apresentado um padrão, ele tenha como output o padrão pretendido.

**DEFINIÇÃO DAS ENTRADAS E SAÍDAS DA REDE**

Os dados para o treino e de teste usámos o dataset notMNIST\_small. Cada imagem tem uma única letra de tamanho 28x28, para o treino temos dois conjuntos em que o primeiro é uma matriz 784x18724 com todas as imagens (em grayscale) e outra matriz 10x18724 que corresponde às 10 classes, onde cada classe é representada por um valor de 0 a 9 que corresponde uma letra de A-Z respetivamente.

Para a saída temos valores de 1 a 10 que corresponde a cada letra.

# Cap. 3: Análise de Resultados

Para analisar resultados partimos de um teste default, em que os seus parâmetros são os que o próprio matlab define por default para um tipo de rede patternet, sendo por isso o seu resultado bastante bom.

Os restantes testes são testes que tem o teste default por base sendo que são alterados certos parâmetros.

Os resultados destes mesmos testes estão indicados na página seguinte.

Teste Default

Uma imagem com texto, relógio

Descrição gerada automaticamente

Arquitetura da rede

Coeficiente de Aprendizagem=0.01

Épocas= 1000

Função de treino : traincg

Tipo de rede: patternet

Uma imagem com mesa

Descrição gerada automaticamente

Matriz de confusão

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Classes/Métricas | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Precisão | 0.90 | 0.80 | 0.90 | 0.87 | 0.82 | 0.87 | 0.85 | 0.85 | 0.80 | 0.85 |
| Sensibilidade | 0.56 | 0.91 | 0.90 | 0.93 | 0.91 | 0.92 | 0.90 | 0.94 | 0.90 | 0.86 |
| F-measure | 0.69 | 0.851 | 0.90 | 0.89 | 0.86 | 0.89 | 0.87 | 0.89 | 0.84 | 0.85 |

Teste 1- Alteração da arquitetura

Uma imagem com texto, relógio

Descrição gerada automaticamente

Arquitetura da rede

Coeficiente de Aprendizagem=0.01

Função de treino : traincg

Tipo de rede: patternet

Uma imagem com mesa

Descrição gerada automaticamente

Matriz de confusão

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Classes/Métricas | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Precisão | 0.93 | 0.83 | 0.88 | 0.88 | 0.81 | 0.89 | 0.82 | 0.84 | 0.82 | 0.88 |
| Sensibilidade | 0.57 | 0.93 | 0.90 | 0.93 | 0.92 | 0.91 | 0.91 | 0.94 | 0.90 | 0.90 |
| F-measure | 0.70 | 0.87 | 0.90 | 0.89 | 0.86 | 0.90 | 0.87 | 0.89 | 0.86 | 0.90 |

Teste 2- Alteração do coeficiente de aprendizagem

Uma imagem com texto, relógio

Descrição gerada automaticamente

Arquitetura da rede

Coeficiente de Aprendizagem=0.1

Função de treino : traincg

Tipo de rede: patternet

Uma imagem com mesa

Descrição gerada automaticamente

Matriz de confusão

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Classes/Métricas | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Precisão | 0.95 | 0.88 | 0.92 | 0.91 | 0.87 | 0.92 | 0.88 | 0.89 | 0.86 | 0.91 |
| Sensibilidade | 0.66 | 0.95 | 0.93 | 0.95 | 0.93 | 0.95 | 0.94 | 0.96 | 0.92 | 0.93 |
| F-measure | 0.78 | 0.91 | 0.92 | 0.93 | 0.93 | 0.90 | 0.92 | 0.89 | 0.89 | 0.92 |

Teste 3-Alteração da função de treino

Uma imagem com texto, relógio

Descrição gerada automaticamente

Arquitetura da rede

Coeficiente de Aprendizagem=0.01

Função de treino : traingd

Tipo de rede: patternet

Uma imagem com mesa

Descrição gerada automaticamente

Matriz de confusão

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Classes/Métricas | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Precisão | 0.94 | 0.65 | 0.31 | 0.76 | 0.16 | 0.68 | 0.78 | 0.22 | 0.50 | 0.03 |
| Sensibilidade | 0.16 | 0.96 | 0.99 | 0.97 | 0.97 | 0.93 | 0.84 | 1 | 0.94 | 0.84 |
| F-measure | 0.28 | 0.77 | 0.47 | 0.85 | 0.25 | 0.78 | 0.81 | 0.36 | 0.65 | 0.05 |

Teste 4-Alteração da função de treino com mais iterações

Uma imagem com texto, relógio

Descrição gerada automaticamente

Arquitetura da rede

Épocas=10000

Coeficiente de Aprendizagem=0.01

Função de treino : traingd

Tipo de rede: patternet

Uma imagem com mesa

Descrição gerada automaticamente

Matriz de confusão

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Classes/Métricas | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Precisão | 0.96 | 0.74 | 0.84 | 0.85 | 0.73 | 0.86 | 0.80 | 0.81 | 0.75 | 0.75 |
| Sensibilidade | 0.36 | 0.98 | 0.94 | 0.97 | 0.95 | 0.95 | 0.96 | 0.94 | 0.93 | 0.94 |
| F-measure | 0.55 | 0.84 | 0.88 | 0.89 | 0.83 | 0.90 | 0.87 | 0.87 | 0.84 | 0.83 |

# Cap. 4: Conclusões

Como já foi referido anteriormente o teste default serve como um “benchmark” teste para poder comparar os outros testes com este mesmo teste.

Em relação ao teste 1, foi alterada a arquitetura da rede neuronal passando agora a contar com duas hidden layers [10 50] em vez de uma [10].Através dos resultados obtidos podemos concluir que a performance da rede subiu ligeiramente, concluindo assim também que o aumentar do número de hidden layers ajuda ao aumento da performance neste caso de estudo.

Em relação ao teste 2, foi alterado o coeficiente de aprendizagem de 0.01 para 0.1. Sendo que os resultados passaram de uma média de 85 para 90.Através dos resultados obtidos podemos concluir que a performance da rede subiu significativamente, concluindo assim também que o aumentar do número do coeficiente de aprendizagem ajuda ao aumento da performance neste caso de estudo.

Em relação ao teste 3, foi alterada a função de treino de traincg para traingd. Sendo que os resultados passaram de uma média de 85 para 50. Através dos resultados obtidos podemos concluir que a performance da rede desceu muito significativamente, sendo que por isso a função traingd não é das melhores funções de treino tendo em conta este caso de uso.

Por fim em relação ao teste 4, foi alterada a função de treino de traincg para traingd e aumentado o número de épocas de 1000 para 10000. Sendo que os resultados passaram de uma média de 85 para 81. Através dos resultados obtidos podemos concluir que a função de treino traincg realmente não é mesmo das melhores funções de treino para usar neste caso de estudo pois mesmo após 10x mais épocas os resultados ainda são piores.

# Referências

<http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html>

https://www.kaggle.com/lubaroli/notmnist