Reconhecimento de letras manuscritas de A à Z

Barbara Pasqualotto[1], **Clodoaldo Basaglia da Fonseca**[1]

[1]Universidade Técnológica Federal Do Paraná

[2]Departamento Acadêmico de Ciência da Computação

[3]Inteligencia Artificial (BCC35G)

bpasqualotto@alunos.utfpr.edu.br,clodoaldofonseca92@gmail.com

Abstract. *This paper abords the utilization of 3 types of supervised learning algirthms to recognize letters from A to Z from the Mnist database. Here we explain a little about each one of them and present the results.*

Resumo. *Neste artigo abordamos a utlização de 3 tipos de algoritmos de aprendizagem supervisionada para reconhecer letras de A à Z da base de dados Mnist. Aquie explicaremos cada um deles e apresentar os resultados obtidos.*

1. Ambiente dos experimentos

Os algiritmos, bem como a extração das caracteristicas de cada uma das imagens dos grupos de teste e treino, foram realizados em um computador com as seguintes características:

- Windows 10 Pro 64bis
- Intel Core i5-7400 3GHz
- 16Gb de RAM DDR4 2400mhz

A linguagem utilizada, bem como as bibliotecas utilizadas, foram feitas em Python 3.6. Utilizamos o conjunto de bibliotecas a seguir:

- scikit-learn, para os algoritmos de aprendizagem
- Numpy para as estruturas de dados
- Pandas para a manipulação de arquivos .CSV
- Pillow para manipulação das imagens
- OpenCV para extração de caracteristicas

1.1. Scikit-learn

Scikit-learn foi desenvolvida em 2007 como parte de um projeto de verão do Google por David Cournapeau. Ela provem vários algoritmos de aprendizagem tanto superviosonada como não supervisionada através de uma interface concistente em Python.[Brownlee 2014]

1.2. Numpy

Numpy é um pacote fundamenta para computação em Python, contendo vários tipos de arrays N-dimensionais, transformadas mátematicas, ferramentas para integração com outras linguagens, algebra linear,etc. Além do uso cientifico, Numpy pode ser usada em uma grande gama de bases de dados com tipos genéricos de dados ou ainda tipos definidos pelo usuário.[developers 2019]

1.3. Pandas

Pandas é uma biliteca *Open Source* que prove alta performance, estruturas de dados faceis de utilizar, além de ferramentas para analize de dados, tudo para a utilzação com Python. Nesse trabalho foram utilizados as ferramentas para lidarmos com os .csvs criados na extração de dados.

1.4. Pillow

Pillow é um fork do PIL(Python Imaging Library) criado por Alex Clark e uma gama de contribuidores. Tem como objetivo ser mais amigavel que PIL. Por sua vez, Pillow oferece uma gama de funções para a abertura e gravação de imagens, bem como a manipulação e alteração das mesmas[Clark 2019].

1.5. OpenCV

OpenCV ou *Open Source Computer Vision Library* é uma biblioteca de código livre para visão computacional e aprendizagem de máquina. Esta biblioteca conta com mais de 2500 algoritmos que incluem algoritmos classicos, bem como os em estado da arte para visão computacional e aprendizagem de máquina. Além da interface para Python, aprezenda para liguagens C++, Java e Matlab, podendo ser utilizada no Linux, Windows, Android e Mac OS. Pode ser usada ainda em conjunto com o CUDA e OpenCL.

2. Extração de dados

Primeiramente, ao verificarmos que cada imagem dabase tinha um tamanho diferente fazendo com que o número de caracteristicas variasse muito de imagem para imagem, utilizamos a biblioteca para redimensionarmos cada uma delas e depois, separamos em uma pasta diferente para que facilitasse a extração depois, como podemos ver a seguir:

```
def resize(caminho,arq):
    imagem = Image.open(caminho)
    altura, largura= 20,20
    saida = imagem.resize((largura,altura),Image.NEAREST)
    pasta = arq[0:3]
    if not os.path.exists('treinoResize'+pasta):
        os.mkdir('treinoResize'+pasta)
    caminhoSaida = 'treinoResize'+arq
    saida.save(caminhoSaida)
    return caminhoSaida
```

A função acima recebe o nome do arquivo, bem como o caminho até ele, abrindo-o atravez do Image da biblioteca Pillow, redimensionando cada um deles para o tamanho de 20×20 pixels, logo após, salvando a nova imagem no diretorio treinoResize ou testeResize, devolvendo o caminho da imagem salva.

Já em mãos das imagens redimencionadas, extraimos as caracteristicas através da função a seguir, utilizando-se da biblioteca OpenCV:

```
def extrairCaracteristicas(img):
    imagem = cv2.imread(img)
    blur = cv2.blur(imagem,(5,5))
    edge = cv2.Canny(blur,0,100,3)
    return edge.flatten()
```

A imagem é lida do disco, passa por filtros low-pass filters(LPF) e high-pass filters(HPF) através da função blur para que sejam removidos os ruídos utilizando-se de um *kernel* de 25 pixels.

Utilizamos a função Canny para detecção de bordas, com parametros de 0 e 100 representando a faixa de detecção dos pixels, sendo 3 o tamanho do *kernel* utilizado. A utilização do Canny da-se por ter uma baixa taxa de erro, localiza bem a distancia da borda real e do pixel e devolve apenas um detector por borda. Por fim, a matriz resultante é transoformada em um *Numpy array* através da função .flatten(). Esse método retorna o total de 400 caracteristicas variando seus valores de 0 a 255. As caracteristicas, bem como as classes que representão, são armazenadas em um arquivo *.csv* que depois é utilizado pelos algoritmos de aprendizagem[team 2019].

3. Os algoritmos

3.1. K-Nearest Neighbor

KNN ou K-Nearest Neighbor (K vizinhos mais próximos) é um algoritmo de aprendizagem supervisionada muito usado em aprendizagem de máquina. Nele usamos um conjunto de treinamento, um parâmetro de quantos vizinhos serão utilizados para medir a distância, e definimos uma forma de calcular a distância entre eles. Quando um dado novo é inserido, ou seja, um dado não classificado, ele não é comparado com com os dados classificados. Na verdade é feito um cálculo para medir a distância entre os dados, para assim classificá-lo.

3.2. SVM

SVM ou Support Vector Machine (Máquina de vetores de suporte) é também um algoritmo de aprendizagem supervisionada, um classificador linear binário não probabilístico. É um método que procura separar diferentes classes de dados. É gerado um hiperplano, são traçadas retas para separar os dados, e são usados os vetores de suporte, que são gerados durante o treinamento. No treinamento são usados conjuntos de exemplos, um para cada categoria.

3.3. Decision Trees

Decision Trees ou Árvores de Decisão é um dos algoritmos de aprendizagem mais simples e mais usado. São feitos testes para chegarem em cada decisão. Na árvore existem vários nós e ramos. Cada nó se refere a um estado e cada ramo aos resultados possíveis de cada estado.

3.4. Resultados

O dataset utilizado nesse experimento compreende-se de 37440 instancias para o treino dos algoritmos, sendo que o dataset para o teste tem 11941 instancias. Os dois datasets foram gerados extraindo as caracteristicas das imagens da base de dados Mnist.

Com a utilização do algoritmo Knn chegamos aos resulados expressados na Tabela

K	Acerto(%)
3	52%
5	51%
7	50%

Tabela 1. k escolhidos por serem os mais recorrentes na literatura

Na Decision Tree com max depth = 11, como pode ser visto na configuração no código abaixo:

```
dtree = DecisionTreeClassifier(max_depth=11)
```

Chegamos a um nível de acerto de 24,9%, sendo esse o menor nível atingido neste artigo.

Já com o algortitmo Support Vector Machine, a configuração utilizada foi:

```
SVC(kernel='poly',C=1 ,gamma='auto')
```

O kernel utililizado foi o polinomial já que esse, além de olhar as caracteristicas das amostras para determinar a sua semelhança, ele também combina as mesmas de forma regressiva. C é o parametro de penalidade de erro, mantido em 1. Já gama é o coeficiente do kernel, calculado através da divisão de 1 pelo número de amostras.

Resultando em uma taxa de acerto de 77,2% sendo está, a maior desse trabalho.

Referências

Brownlee, J. (2014). A gentle introduction to scikit-learn: A python machine learning library. *Machine Learning Mastery!*

Clark, A. (2019). Pillow. *Read The Docs*.

developers, N. (2019). About numpy. *NumPy.org*.

team, O. (2019). Opencv.