1 Representação

Esta atividade laboratorial tem como objetivo a investigação exploratória de um algorítmo para a classificação de imagens. As imagens analisadas são representações manuscritas de dígitos de 10 categorias diferentes, que representam os número decimais $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$. A figura 1 demonstra alguns exemplos das imagens que devem ser classificadas.

Figura 1: Dígitos a serem classificados

Além das figuras, um arquivo de texto rotula cada uma das imagens, classificando-a em alguma categoria.

A atividade se iniciou com base na implementação do programa *digits.py* que extrai uma representação bem simples de cada uma das imagens. Para cada imagem, se gera uma nova imagem no tamanho de $10x20$. E para cada pixel verifica-se o valor de intensidade do pixel e se esse valor for maior que 128, a característica é igual a 1, caso contrário 0.

Neste ponto, notou-se a possibilidade da primeira exploração. Além do tamanho inicial proposto, uma adaptação no algorítmo realizou uma análise em toda a base de imagens, obtendo outras dimensões para verificação. As estratégias adotadas foram:

1. Obter a média entre todas alturas e larguras;
2. Obter a mediana entre todas alturas e larguras;
3. Obter os valores máximos entre alturas e larguras;

Após análise dos resultados, notou-se que a média e a media possuam resultados muitos similares, por isso, optou-se apenas pela utilização da mediana. A Tabela 1 mostra as coleções utilizadas nos experimentos. Além da médiana (med_x, med_y) e dos maiores valores(max_x e max_y) também foram considerados empiramente os valores de x, y como $[20, 40]$ e $[30, 60]$.

Para automatização da geração das representações, adaptações no código foram realizada para que fosse possível utilizar como entrada um arquivo do tipo JSON demonstrado no Código 1:

```
[
  {
    "data": "features_10_20",
    "x": 10,
    "y": 20
  },
  ...
]
```

Código 1: JSON para representações

Neste json, *data* é o nome da coleção de representações, x e y podem ser valores fixos ou terem os valores para obtenção dinâmica de: avg_x, avg_y, med_x, med_y, max_x, max_y.

2 Experimentos

Assim como na adaptação do primeiro algoritmo, um sistema de entrada JSON possibilitou a variação de combinações para análise de diferentes resultados.

O Código 2 mostra as opções de entrada para o algorítmo, podendo variar:

- O arquivo de representação fonte; as variações testadas foram: [features_10_20, features_20_40, features_30_60, features_46_36, features_81_99]
- Ativar ou desativar a normalização; as variações testadas foram: [0,1]
- Alterar o tipo de distância; as variações testadas foram: [eucledian, manhattan]
- Alterar o K; as variações testadas foram: [1,3,4,5,6,7,8,9,10,11,12,13]

```
[
  {
    "data": "features_10_20",
    "normalized": 0,
    "distance": "euclidean",
    "k": 1
  },
  ...
]
```

Código 2: JSON para experimentos

3 Resultados

Foram realizados 240 execuções para as variações listadas anteriormente. Os 20 melhores resultados de ordenados por acurácia e F1Score estão detalhados na Tabela 2.

Pode-se observar que os melhores resultados foram obtidos a partir das representações que consideraram as maiores alturas e larguras das imagens (81x99). Isso pode ter acontecido pois, no momento em que se realiza a compressão das imagens, características podem ser perdidas. O melhor resultado teve a Acurácia de $0,927$ e o F1Score de $0,928$ demorando 962 segundos para execução completa.

Na sequência, a estratégia de obter as médias das alturas e larguras também apresentaram resultado significativo (imagens 46x36), mas ficaram empatadas com a estratégia empírica que utiliza imagens 20x40. O destaque aqui é para o tempo de execução. Com a *features_20_40* conseguiu-se uma Acurácia de $0,925$ e o F1Score de $0,926$ demorando apenas 43 segundos para execução completa.

As normalizações não parecem fazer muita diferença nos resultados pois eles se encontram bem distribuídos.

Em geral a alteração da estratégia de medição das distâncias (euclidean e manhattan) não influenciaram em uma melhoria da Acurácia ou do F1Score. O que se pode notar, foi que nos testes executados a estratégia manhattan impacta negativamente no tempo de execução.

Quanto ao K os melhores resultados foram obtidos com $K = 1$, seguido de $K = 3$ e $K = 5$.

X_Source	Y_Source	X	Y	File Size	Execution Time (s)
10	20	10	20	2180534	1.69
20	40	20	40	9377970	5.44
30	60	30	60	22978779	12.25
med_x	med_y	46	36	20962221	11.23
max_x	max_y	81	99	110050116	52.76

Tabela 1: Estratégias de representação

Experiment	Normalized	Distance	K	Accuracy	F1Score	Execution Time (s)
features_81_99	0	euclidean	1	0,927	0,928	962
features_81_99	0	manhattan	1	0,927	0,928	1.386
features_81_99	1	euclidean	1	0,927	0,928	1.772
features_81_99	1	manhattan	1	0,927	0,928	2.495
features_20_40	0	euclidean	1	0,925	0,926	43
features_20_40	0	manhattan	1	0,925	0,926	86
features_20_40	1	euclidean	1	0,925	0,926	127
features_20_40	1	manhattan	1	0,925	0,926	170
features_46_36	0	euclidean	1	0,925	0,926	597
features_46_36	0	manhattan	1	0,925	0,926	686
features_46_36	1	euclidean	1	0,925	0,926	771
features_46_36	1	manhattan	1	0,925	0,926	859
features_30_60	0	euclidean	1	0,924	0,924	214
features_30_60	0	manhattan	1	0,924	0,924	314
features_30_60	1	euclidean	1	0,924	0,924	406
features_30_60	1	manhattan	1	0,924	0,924	504
features_20_40	0	euclidean	3	0,920	0,921	46
features_20_40	0	manhattan	3	0,920	0,921	89
features_20_40	1	euclidean	3	0,920	0,921	131
features_20_40	1	manhattan	3	0,920	0,921	173

Tabela 2: Resultados

4 Código Fonte

Os códigos preparados podem ser analisados através do repositório: https://github.com/diogocezar/machine-learning/tree/master/lab1/src