Aprendizagem de Máquina (2020/Período Especial) - Impactos da Representação

Diogo C. T. Batista[1]

[1]Universidade Federal do Paraná (UFPR)
Curitiba – Paraná – Brasil

diogo@diogocezar.com

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. Como no exemplo do Código 1.

```
1 data/cdf0361_07_13_0.jpg 0
2 data/cdf1697_45_29_0.jpg 0
3 data/cdf0872_30_6_1.jpg 0
4 data/cdf0371_24_4_0.jpg 0
5 data/cdf1539_20_17_2.jpg 0
```

Código 1. Exemplo dos Rótulos

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 seu valor de intensidade; 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 mediana retornaram 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), os maiores valores encontrados para as dimensões (max_x e max_y) também foram considerados. E ainda, empiramente os valores de x, y como $[20, 40]$ e $[30, 60]$.

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

Para automatização da geração das representações, foram criadas adaptações no código para que fosse possível utilizar como entrada um arquivo do tipo JSON demonstrado no Código 2:

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

Código 2. 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 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]. Foram considerados os K pares, apenas para efeito de comparações, sabendo-se se sua ineficiência por natureza.

O Código 3 mostra as opções de entrada para o algorítmo, podendo variar:

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

Código 3. JSON para experimentos

3. Resultados

Foram realizadas *240* execuções para as variações listadas anteriormente. Os *20* melhores resultados (ordenados por Acurácia e F1Score) estão detalhados na Tabela 2. Já os 20 piores resultados, (ordenados por Acurácia e F1Score) estão detalhados na Tabela 3

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

Já para nos piores resultados, nota-se que foram obtidos na variação de K: $K = 13$, $K = 12$, $K = 11$. Quanto ao K para os melhores resultados, percebe-se que foram obtidos com $K = 1$, seguido de $K = 3$ e $K = 5$. Isso mostra que a distribuição possui uma particularidade interessante, na qual ao se comparar apenas os vizinhos mais próximos se consegue um melhor resultado. Como já esperado, para os resultados nos quais K tinha um valor par, seus resultados foram menos eficientes.

Em outra análise, a estratégia de obter as médias das alturas e larguras também apresentaram resultados significativos (imagens de dimensão $46x36$), mas ficaram empatadas com a estratégia empírica que utiliza imagens $20x40$. O destaque 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 estes já 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.

4. Comparação das Matrizes de Confusão

Para cada execução, criou-se um arquivo de resultados que também armazena as matrizes de confusão.

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. Melhores Resultados

A Tabela 4 demonstra a matriz de confusão para o melhor resultado, enquanto que a Tabela 5 demonstra a matriz de confusão para o pior resultado.

É possível notar que apesar dos erros persistirem na classificação, como por exemplo em: [[0,2],[1,2],[1,3],[1,4],[1,6],[1,7],[1,8],[1,9], eles diminuram consideravelmente. Como por exemplo, no pior caso, a classificação [1,4] foi classificada erroniamente 11 vezes, enquanto que no melhor caso, o erro ocorreu em 8 vezes.

Isso acontece, pois, aumentar o número de *pixels* analisados nas imagens, geram representações mais precisas, porém, como o método é o mesmo, pôde-se observar que os equívocos continuaram existindo nos mesmos lugares, mesmo que em menor número.

5. Código Fonte

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

Experiment	Normalized	Distance	K	Accuracy	F1Score	Execution Time
features_20_40	0	euclidean	13	0,872	0,875	82
features_20_40	0	manhattan	13	0,872	0,875	124
features_20_40	1	euclidean	13	0,872	0,875	166
features_20_40	1	manhattan	13	0,872	0,875	208
features_20_40	0	euclidean	11	0,874	0,876	75
features_20_40	0	manhattan	11	0,874	0,876	117
features_20_40	1	euclidean	11	0,874	0,876	159
features_20_40	1	manhattan	11	0,874	0,876	201
features_20_40	0	euclidean	12	0,875	0,878	79
features_20_40	0	manhattan	12	0,875	0,878	120
features_20_40	1	euclidean	12	0,875	0,878	162
features_20_40	1	manhattan	12	0,875	0,878	204
features_46_36	0	euclidean	13	0,877	0,880	679
features_46_36	0	manhattan	13	0,877	0,880	764
features_46_36	1	euclidean	13	0,877	0,880	852
features_46_36	1	manhattan	13	0,877	0,880	938
features_10_20	0	euclidean	13	0,877	0,880	10
features_10_20	0	manhattan	13	0,877	0,880	20
features_10_20	1	euclidean	13	0,877	0,880	30
features_10_20	1	manhattan	13	0,877	0,880	40

Tabela 3. Piores Resultados

94	1	0	0	0	1	1	0	0	0
0	93	0	0	0	1	0	0	0	1
1	1	101	1	0	0	0	4	1	2
0	1	0	99	0	1	0	0	2	0
0	8	0	0	84	0	1	0	0	2
2	0	0	5	0	89	1	0	0	0
1	5	0	0	0	0	100	0	0	0
0	3	1	0	1	0	0	88	0	4
0	3	0	1	1	2	0	1	79	0
0	1	0	0	1	0	0	10	0	100

Tabela 4. Matriz de Confusão - Melhor Caso

94	1	0	0	0	2	0	0	0	0
0	94	0	1	0	0	0	0	0	0
2	8	92	1	1	0	3	4	0	0
0	1	1	98	0	0	0	2	1	0
0	11	1	0	79	0	1	1	0	2
2	1	0	7	0	86	1	0	0	0
1	9	0	0	0	0	96	0	0	0
0	11	0	0	1	0	0	81	0	4
0	7	0	6	0	3	0	3	67	1
0	4	0	0	4	0	0	14	0	90

Tabela 5. Matriz de Confusão - Pior Caso