

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

Diogo C. T. Batista<sup>1</sup>

<sup>1</sup>Universidade Federal do Paraná (UFPR)  
Curitiba – Paraná – Brasil

diogo@diogocezar.com

## 1. Representação

Esta atividade laboratorial tem como objetivo a investigação exploratória de um algoritmo para a classificação de imagens. As imagens analisadas são representações manuscritas de dígitos de 10 categorias diferentes, que representam os números 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  $10 \times 20$ . 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 algoritmo 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 muito 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, empiricamente os valores de  $x, y$  como  $[20, 40]$  e  $[30, 60]$ .

X_Source	Y_Source	X	Y	File Size	Execution Time (s)
20	10	20	10	2180534	1.57
40	20	40	20	9377970	4.84
60	30	60	30	22978779	11.04
$med\_x$	$med\_y$	36	46	20962221	10.54
$max\_x$	$max\_y$	99	81	110050116	46.63

**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:

```

1  [
2    {
3      "data": "features_20_10",
4      "x": 20,
5      "y": 10
6    },
7    ...
8  ]

```

**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: [euclidian, 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 algoritmo, podendo variar:

```
1  [  
2    {  
3      "data": "features_20_10",  
4      "normalized": 0,  
5      "distance": "euclidean",  
6      "k": 1  
7    },  
8    ...  
9  ]
```

**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 (99x81). Isso acontece pois, no momento em que se realiza a compressão das imagens, características são perdidas. O melhor resultado teve a *Acurácia* de 0,928 e o *F1Score* de 0,929 demonstrando 2.743 segundos para execução completa.

Já para nos piores resultados, nota-se que foram obtidos na variação de  $K$ :  $K = 13$ ,  $K = 12$ ,  $K = 10$ . 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 36x46), mas ficaram abaixo dos resultados da estratégia empírica que utiliza imagens 60x30. O destaque é para o tempo de execução em *features\_60\_30*, que conseguiu-se uma Acurácia de 0,925 e o F1Score de 0,926 demorando apenas 203 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_99_81	0	euclidean	1	0,928	0,929	2.743
features_99_81	0	manhattan	1	0,928	0,929	3.161
features_99_81	1	euclidean	1	0,928	0,929	3.550
features_99_81	1	manhattan	1	0,928	0,929	3.961
features_60_30	0	euclidean	1	0,925	0,926	203
features_60_30	0	manhattan	1	0,925	0,926	293
features_60_30	1	euclidean	1	0,925	0,926	380
features_60_30	1	manhattan	1	0,925	0,926	471
features_36_46	0	euclidean	1	0,923	0,924	558
features_36_46	0	manhattan	1	0,923	0,924	641
features_36_46	1	euclidean	1	0,923	0,924	720
features_36_46	1	manhattan	1	0,923	0,924	804
features_36_46	0	euclidean	3	0,919	0,921	565
features_36_46	0	manhattan	3	0,919	0,921	647
features_36_46	1	euclidean	3	0,919	0,921	727
features_36_46	1	manhattan	3	0,919	0,921	811
features_99_81	0	euclidean	3	0,919	0,921	2.778
features_99_81	0	manhattan	3	0,919	0,921	3.193
features_99_81	1	euclidean	3	0,919	0,921	3.584
features_99_81	1	manhattan	3	0,919	0,921	3.992

**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 diminuíram consideravelmente. Como por exemplo, no pior caso, a classificação [1,4] foi classificada erroneamente 12 vezes, enquanto que no melhor caso, o erro ocorreu em apenas 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 (s)
features_20_10	1	manhattan	13	0,871	0,873	40
features_20_10	1	euclidean	13	0,871	0,873	30
features_20_10	0	manhattan	13	0,871	0,873	20
features_20_10	0	euclidean	13	0,871	0,873	10
features_20_10	1	manhattan	12	0,874	0,876	39
features_20_10	1	euclidean	12	0,874	0,876	29
features_20_10	0	manhattan	12	0,874	0,876	19
features_20_10	0	euclidean	12	0,874	0,876	9
features_99_81	1	manhattan	12	0,876	0,879	4.294
features_99_81	1	euclidean	12	0,876	0,879	3.894
features_99_81	0	manhattan	12	0,876	0,879	3.485
features_99_81	0	euclidean	12	0,876	0,879	3.090
features_20_10	1	manhattan	10	0,879	0,881	37
features_20_10	1	euclidean	10	0,879	0,881	27
features_20_10	0	manhattan	10	0,879	0,881	17
features_20_10	0	euclidean	10	0,879	0,881	7
features_99_81	1	manhattan	13	0,880	0,883	4.328
features_99_81	1	euclidean	13	0,880	0,883	3.928
features_99_81	0	manhattan	13	0,880	0,883	3.517
features_99_81	0	euclidean	13	0,880	0,883	3.126

**Tabela 3. Piores Resultados**

95	0	0	0	0	1	1	0	0	0
0	93	0	0	0	1	0	0	0	1
0	1	103	1	0	0	0	4	1	1
0	1	0	98	0	1	0	0	3	0
0	8	0	0	83	1	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	2	0	0	9	0	100

**Tabela 4. Matriz de Confusão - Melhor Caso**

95	1	0	0	0	1	0	0	0	0
0	95	0	0	0	0	0	0	0	0
3	9	89	1	0	0	3	6	0	0
1	1	1	98	0	0	0	2	0	0
0	12	0	0	78	0	1	0	0	4
0	1	0	7	0	88	1	0	0	0
1	6	0	0	0	0	99	0	0	0
0	11	0	0	3	0	0	79	0	4
0	5	0	3	1	6	0	5	65	2
0	3	0	0	7	0	0	17	0	85

**Tabela 5. Matriz de Confusão - Pior Caso**