Reconhecimento de caracteres manuscritos utilizando as bibliotecas *scikit-image* e *scikit-learn*

Dennis Felipe Urtubia[1], **Jorge L. F. Rossi**[1]

[1]Departamento de Computação – Universidade Tecnológica Federal do Paraná (UTFPR)
Caixa Postal 071 – 87.301-899 – Campo Mourão – PR – Brasil

{dennisurtubia, jorgerossi}@alunos.utfpr.edu.br

Abstract. *This article describes the image processing/artificial intelligence techniques and methods used in feature extraction and classification of handwritten characters of MNIST dataset. The proposed work was to find the features that better differentiate the characters and then from the obtained data build a dataset, which was posteriorly tested in different classification algorithms.*

Resumo. *Este artigo descreve as técnicas e métodos de processamento de imagens/inteligência artificial utilizados na extração de características e classificação de caracteres manuscritos da base de dados MNIST. O trabalho proposto foi encontrar características que melhor diferenciem os caracteres e a partir dos dados obtidos construir um dataset, o qual posteriormente foi testado em diferentes algoritmos de classificação.*

1. Introdução

Visão computacional é a área da computação que busca desenvolver meios que ajudem o computador a compreender o conteúdo de imagens e vídeos. Nesse sentido, faz-se necessário utilizar técnicas que possibilitem que as características mais importantes dessas imagens sejam identificadas e extraídas. Para tal, foram desenvolvidos diversos algoritmos destinados a extrair determinadas características de imagens, como contornos, cores, contagem de pixels, entre outros. Um exemplo de caso onde a visão computacional pode ser implementada é no reconhecimento de caracteres do alfabeto escritos a mão, problema qual o presente trabalho busca solucionar. A partir dos resultados obtidos pelos algoritmos de extração de características, é possível treinar algoritmos de classificação para que possam reconhecer e diferenciar os caracteres, como o k-NN, SVM , entre outros.

A base de dados escolhida para esse trabalho foi uma base de caracteres do alfabeto escritos a mão fornecida pelo MNIST, contendo mais de 37000 imagens divididas entre 26 classes. O algoritmo de extração de características escolhido foi o HOG (Histogram of Gradients), fornecido pela biblioteca *scikit-image*. Para classificar as imagens, foram utilizados algoritmos de classificação fornecidos pela biblioteca *scikit-learn*.

2. Referêncial Teórico

Na abordagem de reconhecimento de caracteres descrita nesse trabalho foi utilizado um algoritmo de extração de características de imagens e diversos algoritmos de classificação.

2.1. Extração de características

O Histograma de Gradientes Orientados (HGO) é um algoritmo de extração de características popularmente utilizado para detecção de objetos em imagens. A ideia central por trás do HGO é que a aparência e formas locais de uma imagem podem ser descritas pela distribuição de gradientes de intensidade ou direções das bordas. A imagem de entrada é dividida em pequenas células, e para os pixels de cada célula, o histograma de direções do gradiente é computado. O resultado final do algoritmo é a concatenação dos resultados de cada célula [Wikipedia contributors 2019a].

2.2. Algoritmos de classificação

O Aprendizado Supervisionado é um ramo do Aprendizado de Máquina onde os algoritmos aprendem a partir de dados rotulados. Pode ser dividido em duas categorias: Classificação e Regressão. Classificação é uma técnica utilizada para determinar a qual classe algo depende com base em uma ou mais variáveis, conhecidas como características [Shetty 2018]. Nesse trabalho foram usados diversos algoritmos de classificação.

2.2.1. k-NN

O k-NN (k-nearest neighbors) é um algoritmo de classificação bastante utilizado. Sua implementação é uma das mais simples entre os demais algoritmos e pode ser utilizado tanto para classificação quanto regressão. O algoritmo assume que indivíduos similares estão próximos entre si, portanto calcula a distância do indivíduo a ser classificado em relação à todos os indivíduos de classes conhecidas. Após isso, é realizada uma votação entre os k elementos mais próximos para decidir a qual classe o indivíduo provavelmente pertence [Wikipedia contributors 2019b].

2.2.2. SVM

Assim como o k-NN, o SVM (Support Vector Machines) é um algoritmo supervisionado que pode ser utilizado tanto para classificação quanto regressão. O objetivo é encontrar um hiperplano em um espaço n-dimensional que classifique os dados de forma distinta. Hiperplanos são limites de decisão que auxiliam a diferenciar os objetos, pois dados presentes em lados opostos do hiperplano podem ser atribuídos a diferentes classes. A dimensão do hiperplano é dependente do número de características, por exemplo, se o número de características é 2, o hiperplano é uma linha. Se for 3, torna-se um plano bidimensional, e assim por diante [Gandhi 2018]. Para todas as execuções do algoritmo SVM neste trabalho, foi utilizado o kernel linear e diferentes valores para o parâmetro C, comum em todos os tipos de kernels, representando uma margem, o ponto de equilibro entre a maximização da margem e a minimização do erro.

2.2.3. Árvore de decisão

Uma árvore de decisão utiliza uma estrutura como uma árvore para auxiliar na tomada de decisões [Gupta 2017]. É visualizada de cima para baixo, com a raiz no topo. Os nós não folhas representam condições, enquanto os nós folhas representam a decisão final (classe).

2.3. Random Forest

O algoritmo cria várias árvores de decisão para formar uma floresta de forma aleatória. É um algoritmo de aprendizagem supervisionada, utilizada para classificação e também regressão que apresenta bons resultados a maioria das vezes. Uma outra grande vantagem deste algoritmo, é a facilidade para se medir a importância relativa de cada característica. [Donges 2019]

3. Métodos

3.1. Extração de características

Antes de realizar a extração de características das imagens da base de dados, foi executada uma etapa básica de pré-processamento para deixar todas as imagens do mesmo tamanho, para que seja possível gerar vetores de características com tamanhos iguais. Após redimensionar as imagens, as características puderam ser extraídas. Para tal, foi utilizada a função *hog* (Histogram of Oriented Gradients) disponível na biblioteca *scikit-image*. É possível modificar o funcionamento da função por meio dos parâmetros passados na chamada, como orientação, pixels por célula, entre outras.

Os principais parâmetros de configuração que a função aceita são:

- *orientation* - número de orientações
- *pixels_per_cell* - número de pixels em cada célula da imagem
- *cells_per_block* - número de células por bloco
- *bloc_norm* - método de normalização
- *multichannel* - considerar cores

A fim de simplificar o trabalho, apenas o tamanho da imagem e o parâmetro *pixels_per_cell* foram modificados entre as execuções. Os demais parâmetros foram configurados com o valor padrão. Os valores utilizados foram:

- Tamanho imagem = 25x25 pixels e 100x100 pixels
- *orientation* = 9
- *pixels_per_cell*
  - Tamanho imagem = 25x25 - 5x5 e 7x7
  - Tamanho imagem = 50x50 - 10x10 e 15x10
- *cells_per_block* - 3x3
- *bloc_norm* - L2-Hys
- *multichannel* - False

A partir do vetor de características extraídas, foi gerado um arquivo de dataset no formato *.csv* contendo uma linha para cada imagem. O número de colunas varia com o tamanho da imagem e a combinações de parâmetros da função de extração de características. A base de dados de imagens fornecida pelo MNIST é dividida entre imagens de treino e imagens de teste, portanto foram gerados dois arquivos de saída no processo de extração de características.

3.2. Classificação

Tendo sidas as características das imagens extraídas, foram utilizados algoritmos de classificação disponíveis na biblioteca *scikit-learn* para treinar um modelo que seja capaz de identificar a qual classe cada imagem pertence. Assim como na etapa anterior, foram realizados experimentos com os parâmetros das funções, sendo estes:

1. k-NN
    - k = 1, 3, 5

1. SVM
    - *kernel* = linear
    - C = 0.1, 0.5 e 1

1. Árvore de Decisão
    - *max_depth* = 5 e 15

1. Random Forest
    - *n_estimators* = 120

O resultado de cada execução é uma porcentagem de acertos variando entre 0% e 100%. Além disso, é informado o tempo total gasto na execução do algoritmo.

3.2.1. Configuração do sistema

Os testes foram realizados em um computador contendo as seguintes configurações:

- CPU: Intel i7-8750H (12 núcleos)
- Memória: 8192 MB DDR4
- GPU: NVIDIA GeForce GTX 1050 Ti*
- Sistema Operacional: Ubuntu 19.04 (kernel 5.0)

* O *scikit-learn* não utiliza a GPU.

4. Resultados

Cada tabela dos resultados representa uma combinação diferente de tamanho da imagem e pixels por célula utilizada nos testes. As linhas representam os métodos de classificação utilizados, contendo diferentes configurações.

Tamanho imagem: 25x25 pixels Pixels por célula: 5x5		
Algoritmo de classificação	**Porcentagem de acertos**	**Tempo de execução (segundos)**
k-NN, k = 1	86.74%	535
k-NN, k = 3	87.88%	542
k-NN, k = 5	88.22%	544
SVM, kernel = linear, C = 1	91.24%	221
SVM, kernel = linear, C = 0.5	91.54%	232
SVM, kernel = linear, C = 0.1	91.03%	300
Decision Tree, max_depth = 5	38.05%	6
Decision Tree, max_depth = 15	67.45%	15

Tamanho imagem: 25x25 pixels Pixels por célula: 7x7		
Algoritmo de classificação	**Porcentagem de acertos**	**Tempo de execução (segundos)**
k-NN, k = 1	77.84%	73
k-NN, k = 3	79.22%	78
k-NN, k = 5	79.90%	79
SVM, kernel = linear, C = 1	79.35%	40
SVM, kernel = linear, C = 0.5	78.96%	46
SVM, kernel = linear, C = 0.1	75.94%	72
Decision Tree, max_depth = 5	34.77%	1
Decision Tree, max_depth = 15	60.28%	2

Tamanho imagem: 50x50 pixels Pixels por célula: 10x10		
Algoritmo de classificação	**Porcentagem de acertos**	**Tempo de execução (segundos)**
k-NN, k = 1	89.79%	507
k-NN, k = 3	90.62%	512
k-NN, k = 5	90.64%	511
SVM, kernel = linear, C = 1	92.78%	181
SVM, kernel = linear, C = 0.5	93.25%	192
SVM, kernel = linear, C = 0.1	92.72%	280
Decision Tree, max_depth = 5	42.09%	9
Decision Tree, max_depth = 15	70.56%	22
Random Forest, n_estimators = 120	90.16%	65

Tamanho imagem: 50x50 pixels Pixels por célula: 15x15		
Algoritmo de classificação	**Porcentagem de acertos**	**Tempo de execução (segundos)**
k-NN, k = 1	85.92%	58
k-NN, k = 3	87.42%	65
k-NN, k = 5	87.99%	70
SVM, kernel = linear, C = 1	86.81%	33
SVM, kernel = linear, C = 0.5	86.28%	39
SVM, kernel = linear, C = 0.1	83.42%	65
Decision Tree, max_depth = 5	36.65%	1
Decision Tree, max_depth = 15	67.49%	3

5. Conclusão

A maior porcentagem de acertos obtida na execução dos testes foi de 93.25%, redimensionando a imagem para 50x50 pixels, utilizando 10 pixels por célula no cálculo do Histograma, juntamente com o algoritmo de classificação SVM com kernel linear e C = 0.5.

5.1. Comparando parâmetros da função de extração de características

- Tamanho da imagem: 25x25 pixels
  Ao utilizar 5x5 pixels por célula, a porcentagem de acertos do k-NN e do SVM ficaram próximas de 90%. Os valores obtidos para a Árvore Decisão variaram entre aproximadamente 38% e 67%. Aumentando a quantidade de pixels por célula para 7x7, houve uma pequena redução na porcentagem de acertos (aproximadamente 10%), porém, o tempo de execução diminuiu drasticamente (cerca de 7 vezes).
- Tamanho da imagem: 50x50 pixels
  Os resultados obtidos redimensionando as imagens para 50x50 pixels foram um pouco maiores do que utilizando 25x25 pixels. A maior porcentagem de acertos obtida foi de 93.25% para o SVM com C = 0.5 e a menor foi de 42.09% para a Árvore de Decisão com *max_depth* = 5. Ao analisar os resultados obtidos utilizando 15x15 pixels por célula, observou-se um padrão parecido com o teste com imagens de menor dimensão, visto que a porcentagem de acertos foi reduzida juntamente com o tempo de execução. Porém, essa a redução na porcentagem de acertos foi bem sutil, enquanto o tempo de execução diminuiu cerca de 8 vezes.

5.2. Comparando classificadores

- Tamanho da imagem: 25x25 pixels, pixels por célula: 5x5
  O algoritmo de classificação que melhor reconheceu as imagens foi o SVM, tanto em porcentagem de acertos como em tempo de execução. Comparando com o k-NN, a melhora na acurácia não foi tão significante, porém o tempo de execução foi reduzido aproximadamente pela metade. Utilizando a Árvore de Decisão com o parâmetro *max_depth* = 15, observou-se uma porcentagem de acertos de 67.45% (diferença considerável de 24 pontos percentuais em relação ao SVM), porém, o tempo de execução foi reduzido aproximadamente 15 vezes.
- Tamanho da imagem: 25x25 pixels, pixels por célula: 7x7 Os valores obtidos pelo k-NN e SVM foram muito próximos, porém o tempo de execução do SVM foi aproximadamente a metade. A Árvore de Decisão foi capaz de acertar 60.28% das imagens em apenas 2 segundos.
- Tamanho da imagem: 50x50 pixels, pixels por célula: 10x10 As porcentagens de acerto obtidas pelo SVM foram cerca de 3 pontos percentuais maiores que o k-NN, demorando aproximadamente metade do tempo para executar. A melhor porcentagem obtida pela Árvore de Decisão foi 70.56%.
- Tamanho da imagem: 50x50 pixels, pixels por célula: 15x15 Os resultados foram mistos para o k-NN e o SVM. Apesar de quase 20 pontos percentuais de diferença entre o melhor algoritmo dessa execução (k-NN com k = 5) e a Árvore de Decisão com *max_depth* = 15, o ultimo demorou aproximadamente 23 vezes menos tempo para executar.

5.3. Conclusões finais

Foi possível notar que ao aumentar o tamanho da imagem houve uma pequena melhora na porcentagem de acertos. Além disso, aumentando o número de pixels por célula resulta em uma sutil diminuição da acurácia, porém sai em vantagem pela grande redução no tempo de execução. O melhor algoritmo de classificação observado foi o SVM, saindo em vantagem tanto na porcentagem de acertos quanto no tempo de execução. Apesar de ter sido o algoritmo com menor porcentagem de acertos, a Árvore de Decisão teve o tempo de execução drasticamente menor em todos os casos.

```
[[437   0   0   1   0   1   0   7   1   1   1   0   0   0   1   0   1   2   0   0   1   0   0   5   0   0]
 [  4 410   0   3   2   0   9   1   0   0   0   0   0   0   1   0   0   3   2   0   0   0   0   0   0   0]
 [  0   0 503   0   0   0   5   0   0   0   0   6   0   0   1   2   0   0   1   0   0   0   0   0   0   0]
 [  0   1   0 375   0   0   0   0   0   1   0   0   0   0  12   6   0   1   0   0   0   0   0   0   0   0]
 [  0   2   5   0 344   6   5   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   2]
 [  1   0   0   0   2 378   1   0   2   0   0   0   0   0   0  22   0   0   0  10   0   0   0   0   0   3]
 [  4   3   6   0   0   2 357   1   0   0   0   1   0   0   6   0   7   0   1   0   0   0   0   0   1   0]
 [  7   0   0   1   0   0   0 383   0   1   2   0   2   1   0   0   0   0   0   1   2   0   0   0   2   0]
 [  0   1   4   1   2   2   0   3 744  24   1   8   0   0   0   0   0   0   2  11   0   0   0   5   3   4]
 [  0   1   0   1   0   0   0   0   3 405   0   0   0   0   0   0   0   0  11   4   1   0   0   0   0   0]
 [  2   0   4   0   3   3   1   3   0   0 336   4   0   0   0   0   0  13   0   0   1   0   0   6   1   0]
 [  0   0  19   0   0   0   0   0   0   0   0 475   0   0   0   0   0   0   0   0   0   1   0   1   0   0]
 [  5   0   0   0   0   0   0  23   0   0   0   0 397  19   0   0   0   1   0   0   4   5   1   1   4   0]
 [  0   0   0   0   0   1   0   0   0   0   1   0   2 425   0   0   0   1   0   0   2   2   3   2   0   0]
 [  0   1   2  46   0   0   0   0   0   0   0   0   0   0 404   0   4   0   0   0   2   0   0   0   0   0]
 [  2   0   0   5   0   9   0   0   1   1   0   0   0   0   2 444   0   1   0   0   0   1   0   0   1   0]
 [  2   3   4   4   0   0   3   0   0   0   0   0   0   0  11   1 414   1   0   0   9   0   0   0   0   0]
 [  4   4   5   3   3   3   1   1   2   0  18   2   0   0   0   4   3 385   0   0   1   0   0   2   0   5]
 [  0   1   1   3   1   2   1   0   0  10   0   0   0   0   0   0   1   0 425   0   0   0   0   0   0   0]
 [  1   0   0   0   0   2   0   0   2   5   0   0   0   0   0   2   0   0   0 453   0   0   0   0   4   0]
 [  0   0   0   1   0   0   0   0   0   1   0   1   2   1   2   0   0   0   0   0 423  23   3   0   1   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0  13 464   0   0   3   0]
 [  0   0   0   1   0   0   0   1   0   0   0   1   3   6   0   0   1   0   0   0   8   7 447   0   0   0]
 [  0   0   0   0   0   0   0   0   0   1  13   0   1   1   0   0   1   0   0   1   2   2   0 448   2   0]
 [  1   0   1   0   1   3   0   0   1   1   2   0   0   0   0   5   0   0   0   2   1  18   0   6 411   0]
 [  0   2   0   1  12   0   0   0   1   0   0   3   0   0   0   0   0   0   0   1   0   0   0   1   1 445]]
```

Figura 1. Matriz de Confusão do melhor caso

Referências

Donges, N. (2019). A complete guide to the random forest algorithm.

Gandhi, R. (2018). Support Vector Machine—Introduction to Machine Learning Algorithms.

Gupta, P. (2017). Decision Trees in Machine Learning.

Shetty, B. (2018). Supervised Machine Learning: Classification.

Wikipedia contributors (2019a). Histogram of oriented gradients — Wikipedia, the free encyclopedia.

Wikipedia contributors (2019b). K-nearest neighbors algorithm — Wikipedia, the free encyclopedia.