Classificação de Caracteres Manuscritos usando HOG para extrair características

Diego Takahashi
Universidade Estadual de Maringá(UEM)
Maringá, Paraná, Brasil
diego.taka@hotmail.com

***Abstract*—Este trabalho teve como principal objetivo a classificação de caracteres manuscritos utilizando as características extraídas via *Histogram of oriented gradients* (HOG). Com os seguintes modelos de Machine Learning (ML): *Support Vector Machine* (SVM) e *K-Nearest Neighbors Classifier* (KNN). Os teste foram feitos utilizando duas técnicas a de *holdout* e a de validação cruzada, onde o maior resultado foi atingido no teste de *holdout* utilizando o modelo SVM com o kernel RBF, obtendo um valor de 0,812 para suas métricas. O menor resultado obtido foi para o K = 3 do KNN usando o *holdout* onde obteve 0,725 para as métricas.**

***Index Terms*—*Support Vector Machine*, *K-Nearest Neighbors Classifier*, *Histogram of oriented gradients*, técnicas a de *holdout*, validação cruzada, classificação de caracteres.**

I. Introdução

Uma das aplicações populares no campo da visão computacional é reconhecimento de caracteres manuscritos [1]. Esse trabalho propõe uma metologia simples para o reconhecimento de caracteres manuscritos utilizando os modelos *Support Vector Machine* (SVM) e *K-Nearest Neighbors Classifier* (KNN) para classifica-los. Onde foi utilizada a base de dados *"English Handwritten Characters"* que contém 55 imagens para cada uma das 62 classes (dígitos e letras maiúsculas e minúsculas), totalizando 3410 imagens de caracteres [2]. As características extraídas para a classificação foram geradas pelo Histogram of oriented gradients (HOG).

II. Metodologia

A. Pré-processamento

As imagens da base de dados estavam todas em uma escala de 900x1200 e não possuíam um padrão para o tamanho do caractere e estavam em posições aleatórias, então foi necessário aplicar algumas técnicas para padroniza-las.

1) Binarização: A primeira etapa foi binarizar as imagens em tons de cinza extraídas da base de dados, onde foi utilizado o método de Otsu que é um algoritmo de limiarização. Que tem como objetivo, a partir de uma imagem em tons de cinza, determinar o valor ideal de um *threshold* que separe os elementos do fundo e da frente da imagem em dois, atribuindo a cor branca ou preta para cada um deles [3]. Um exemplo de umas das imagens binarizada pode ser visualizada na Figura 1.

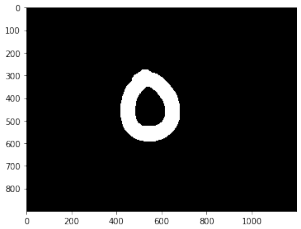

Fig. 1. Imagem binarizada com método otsu.

2) Extração do caractere e redimensionamento: A extração foi feita a partir das imagens binarias, encontrando o caractere e colocando uma caixa delimitadora sobre eles. Como pode ser observado na Figura 2, que é um exemplo de como a extração foi feita.

Fig. 2. Imagem do caractere delimitado.

Um exemplo do caractere extraído da imagem pode ser visualizado na Figura 3.

Após a extração foi feito o redimensionamento das imagens para 28x28, dessa forma garantido que elas estejam na mesma escala e também para diminuir o custo computacional, visto que 900x1200 era um valor muito alto. Pode ser visto na Figura 4 um exemplo de como a imagem fica após o redimensionamento.

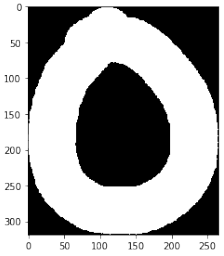

Fig. 3. Imagem do caractere extraído.

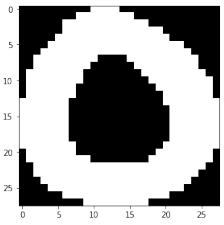

Fig. 4. caractere redimensionado.

B. Extração de características

As características usada para o treinamento foram extraídas utilizando a técnica *Histogram of oriented gradients* (HOG) que é um descritor de características popular para detecção de objetos [4]. O HOG é baseado em gradientes e é descritor mais eficiente para os dígitos manuscritos [5]. O HOG gera histogramas usando a magnitude e orientações do gradiente para as regiões da imagem.

Este método tem como objetivo extrair informações referentes à orientação das arestas existentes em uma imagem, sendo estas arestas calculadas através de métodos de detecção de bordas [6].

A principal ideia é que a aparência e a forma do objeto local dentro de uma imagem possa ser descrita pela distribuição de gradientes de intensidade ou direções de bordas.

A imagem é dividida em pequenas regiões conectadas chamadas células. Para os pixels dentro de cada célula, um histograma de direções de gradiente é compilado. O descritor é a concatenação desses histogramas, o que traduz a imagem em um vetor de dados [6], assim gerando as características a serem utilizadas pelos classificadores, neste trabalho o HOG gerou um vetor com 441 características. Na Figura 5 é apresentado um exemplo de como a imagem fica após o uso do HOG.

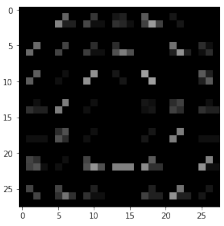

Fig. 5. Imagem do HOG.

III. Modelos de ML

A. Support Vector Machine (SVM)

O método utilizado para a classificação dos dados deste trabalho foi o SVM, que constrói um hiperplano ou conjunto de hiperplanos em um espaço dimensional alto ou infinito, para assim classificar os dados. Intuitivamente, uma boa separação é alcançada pelo hiperplano que possui a maior distância para os pontos de dados de treinamento mais próximos de qualquer classe, pois, em geral, quanto maior a margem, menor o erro de generalização do classificador [7]. A Figura 6 mostra a função de decisão para um problema linearmente separável (i.e., problemas de classificação de padrões que podem ser resolvidos a partir de uma superfície de decisão linear), com três amostras nos limites das margens, chamados de *"support vectors"*.

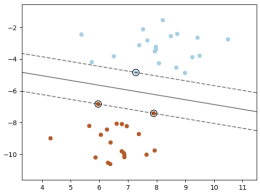

Fig. 6. Função de decisão para um problema linearmente separável.

Onde foram feitos experimentos utilizando o *Suport Vector Classifier* (SVC) da biblioteca do Python *sklearn*, sendo feito testes com os diferentes tipos de *kernel*, sendo eles: rbf, poly, sigmoid e linear.

B. K-Nearest Neighbors Classifier

De acordo com a regra do KNN, sua classificação é baseada em um processo de votação por maioria. Nesse processo, um ponto de consulta é atribuído a classe de dados que tem o maior número de representantes dentro dos vizinhos mais próximos do ponto. Os vizinhos mais próximos são determinados a partir da distância mínima entre o ponto de consulta e as amostras de treinamento [8]. A distância pode, em geral, ser qualquer medida métrica: a distância euclidiana padrão é a escolha mais comum [7]. Na Figura 7, tem-se um exemplo de KNN onde K é o número de vizinhos mais próximos, sendo para K = 3 a amostra pertencerá ao grupo A, enquanto para K = 6 será classificado no grupo B. Neste

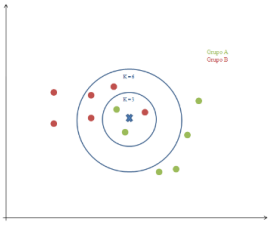

Fig. 7. Exemplo de *KNN* para K = 3 e K = 6 [9].

trabalho o KNN foi testado para quatro cenários para K = 3, 5, 7 e 9.

IV. Divisão do *dataframe* em treino e teste

O conjunto de dados normalmente são divididos entre treino e teste, pois o objetivo é avaliar o modelo para estimar a qualidade da sua generalização de padrão nos dados nos quais o modelo não foi treinado. Essa seção apresentara as técnicas utilizadas para fazer esta divisão.

A. Técnica holdout

Uma forma de dividir o conjunto de dados foi por meio da técnica de *holdout* que consiste em separar dados de treino e dados de teste em uma porcentagem fixa. Neste trabalho foi feita a divisão do *dataframe* em 70% para treino e 30% para teste, também foi definido o parâmetro *random_state* para 0, que garante que a divisão do conjunto de dados seja sempre a mesma, independentemente da execução do código. Isso é útil porque permite que os resultados sejam reproduzidos de maneira confiável, facilitando assim a comparação. Além disso, utilizou-se o parâmetro *stratify* para assim manter o balanceamento do conjunto de dados.

B. Técnica de validação cruzada

A técnica de validação cruzada consiste em particionar os dados em conjuntos, onde um conjunto é utilizado para treino e outro conjunto é utilizado para teste e avaliação do desempenho do modelo. Neste trabalho em especifico a técnica *Stratified K-fold* foi utilizada. Nesta técnica o conjunto de dados é dividido em K partes [10], preservando a porcentagem de amostras para cada classe, assim obtendo resultados diferentes de acordo com cada iteração K do modelo. Neste trabalho K foi definido como 5.

V. Métricas de avaliação dos métodos de ML

A avaliação de um modelo de classificação é feita a partir da comparação entre as classes preditas pelo modelo e as classes verdadeiras de cada exemplo. Todas as métricas de classificação têm como objetivo comum medir quão distante o modelo está da classificação perfeita, porém fazem isto de formas diferentes. As classificações de classe que o modelo gera são: falso positivo (FP), falso negativo (FN), verdadeiro positivo (VP) e verdadeiro negativo (VN).

A. Acurácia

A acurácia (A), conforme a Fórmula (1), indica uma performance geral do modelo, dentre todas as classificações, quantas amostras o modelo classificou corretamente.

$$A = \frac{VP + VN}{VP + VN + FP + FN} \tag{1}$$

B. precisão

A precisão (P), conforme a Fórmula (2), busca determinar dentre todas as classificações de classe positivo que o modelo fez, quantas estão corretas.

$$P = \frac{VP}{VP + FP} \tag{2}$$

Como neste trabalho é um problema de multi classes é necessário usar a micro precisão média que na hora do cálculo ele faz a soma de todos o VPs e FPs obtidos por cada classe.

C. Revocação

A revocação (R), conforme a Fórmula (3), busca determinar dentre todas classificações de classe positivo que o modelo gerou, quantas foram classificadas corretamente como positivos

$$R = \frac{VP}{VP + FN} \tag{3}$$

Similar a precisão, na revocação também é necessário usar a micro média, onde a micro revocação média, faz a soma dos VPs e FPs de cada classe.

VI. Resultados e Comparações

A. Testes com holdout

A Tabela I apresenta os resultados utilizando a técnica de *holdout* ao fazer uso do SVM para os seus diferentes kernels. Ao analisa-la é observado que dentre os tipos de kernel testados o RBF foi o que obteve o maior resultado e o Linear o menor. Além disso o número de VPs, FPs e FNs foi o mesmo para todas as 62 classes, visto que a micro precisão média e micro revocação média foram as iguais.

TABLE I
Resultados SVM com *holdout*.

Kernel	Acurácia	Micro Precisão	Micro Revocação
RBF	**0,812**	**0,812**	**0,812**
Poly	0,79	0,79	0,79
Sigmoid	0,791	0,791	0,791
Linear	0,781	0,781	0,781

Nas Tabelas II e III pode-se observar os resultados obtidos do SVM utilizando o kernel RBF com *holdout*, onde cada caractere (classe) tem sua precisão, revocação quantidade utilizada nesse teste, ao examinar os resultados, temos que houve mais casos de classes precisamente classificadas que o contrário.

TABLE II
Relatório de classificação do SVM.

Caractere	Precisão	Revocação	Quantidade
0	0,52	0,71	17
1	0,70	0,41	17
2	1	0,81	16
3	0,94	1	17
4	0,8	1	16
5	0,83	0,94	16
6	1	0,88	17
7	1	0,94	16
8	1	0,94	17
9	0,71	0,88	17
A	0,94	0,94	16
B	1	1	16
C	0,59	0,94	17
D	1	1	17
E	1	0,94	16
F	1	0,94	16
G	0,94	1	17
H	0,94	1	17
I	0,42	0,50	16
J	0,86	0,71	17
K	0,83	0,88	17
L	1	0,76	17

TABLE III
RELATÓRIO DE CLASSIFICAÇÃO DO SVM(CONT).

Caractere	Precisão	Revocação	Quantidade
M	1	0,88	16
N	1	1	17
O	0,69	0,53	17
P	0,76	0,76	17
Q	1	0,88	16
R	0,94	1	16
S	0,82	0,56	16
T	1	1	16
U	0,87	0,81	16
V	0,67	0,88	16
W	0,82	0,88	16
X	0,80	0,75	16
Y	0,94	0,94	16
Z	0,83	0,94	16
a	0,84	0,94	17
b	0,94	0,94	17
c	0,75	0,35	17
d	0,94	0,94	17
e	0,92	0,75	16
f	0,93	0,8	16
g	0,57	0,75	16
h	0,78	0,88	16
i	0,43	0,71	17
j	0,64	0,88	16
k	0,79	0,88	17
l	0,61	0,65	17
m	0,75	0,75	16
n	0,67	0,71	17
o	0,71	0,59	17
p	0,63	0,75	16
q	1	0,69	16
r	0,86	0,71	17
s	0,71	0,71	17
t	0,89	1	17
u	0,93	0,88	16
v	0,75	0,53	17
w	0,71	0,62	16
x	0,71	0,71	17
y	0,86	0,71	17
z	0,83	0,62	16

Podemos observar na Tabela IV os resultado obtidos ao utilizar o KNN como nosso classificador, onde novamente os valores de micro precisão média e micro revocação média foram as mesmas, além disso a maior métrica obtida foi ao usar o K do KNN como 7 e a menor foi ao utilizar o K = 3.

TABLE IV
RESULTADO KNN COM *holdout*.

K	Acurácia	Micro Precisão	Micro Revocação
3	0,725	0,725	0,725
5	0,736	0,736	0,736
7	**0,741**	**0,741**	**0,741**
9	0,728	0,728	0,728

B. Testes com validação cruzada

A Tabela V apresenta os resultados utilizando a técnica de *K-fold* ao fazer uso do SVM para os seus diferentes kernels. Ao analisa-la, pode-se observar que as métricas obtiveram resultados iguais e que dentre os tipos de kernel testados o RBF foi o que obteve o maior resultado e o Linear o menor.

TABLE V
RESULTADOS SVM COM K-FOLD

Kernel	Acurácia	Micro Precisão	Micro Revocação
RBF	**0,801**	**0,801**	**0,801**
Poly	0,78	0,78	0,78
Sigmoid	0,785	0,785	0,785
Linear	0,772	0,772	0,772

Analisando a Tabela VI, pode-se observar que os resultados das métricas foram semelhantes. Ademais o K que obteve os maiores resultados foi o K = 9, já o K = 3 e 5 geraram os menores resultados ao utilizar a validação cruzada para o KNN.

TABLE VI
RESULTADOS KNN COM *K-fold*

K	Acurácia	Micro Precisão	Micro Revocação
3	0,73	0,73	0,73
5	0,73	0,73	0,73
7	0,731	0,731	0,731
9	**0,733**	**0,733**	**0,733**

C. Comparação dos resultados

No geral os resultados fazendo uso do SVM foram maiores que os fazendo uso do KNN, além disso entre todos os resultado obtidos as métricas avaliadas obtiveram os mesmo resultados, isso provavelmente acontece pois o conjunto de dados está balanceado.

O maior resultado obtido foi no teste de *holdout* utilizando o modelo SVM com o kernel RBF sendo ele 0,812, já o menor resultado obtido foi para o K = 3 do KNN usando o *holdout* onde obteve 0,725.

VII. CONCLUSÃO

Neste trabalho foram avaliados os modelos SVM e KNN para classificação de caracteres, utilizando características extraídas a partir do HOG das imagens. A partir dos testes realizados, conclui-se que o uso de caraterísticas extraídos do HOG para os modelos de ML são promissores no reconhecimento de caracteres manuscritos, porém mais testes devem ser feitos com relação ao pré processamento das imagens e parâmetros dos modelos.

REFERENCES

[1] R. Ebrahimzadeh and M. Jampour, "Efficient handwritten digit recognition based on histogram of oriented gradients and svm," *International Journal of Computer Applications*, vol. 104, no. 9, 2014.

[2] D. Dave, "English handwritten characters," https://www.kaggle.com/datasets/dhruvildave/english-handwritten-characters-dataset, 2021, [Acessado em: 24 de fevereiro de 2023].

[3] L. Torok, "Método de otsu," *Instituto de Computação (UFF)*, 2016.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. Ieee, 2005, pp. 886–893.

[5] K. Greeshma and J. V. Gripsy, "Image classification using hog and lbp feature descriptors with svm and cnn," *Int J Eng Res Technol*, vol. 8, no. 4, pp. 1–4, 2020.

[6] A. Almeida, C. Borges, and I. Paula, “Aplicaçao do descritor hog e classificador svm no reconhecimento de poses humanas em imagens de profundidade,” in *Anais da IV Escola Regional de Informática do Piauí*. SBC, 2018, pp. 119–124.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in python,” Disponível em: https://scikit-learn.org/stable/. Acesso em: 13 de março 2022, 2011.

[8] H. Jaafar, N. H. Ramli, and A. S. A. Nasir, “An improvement to the k-nearest neighbor classifier for ecg database,” in *IOP Conference Series: Materials Science and Engineering*, vol. 318, no. 1. IOP Publishing, 2018, p. 012046.

[9] T. J. R. GUIMARÃRES *et al.*, “Identificação de doenças cardíacas a partir de eletrocardiogramas utilizando machine learning.” 2019.

[10] J. P. Z. Cunha, “Um estudo comparativo das técnicas de validação cruzada aplicadas a modelos mistos,” Ph.D. dissertation, Universidade de São Paulo, 2019.