Aprendizado de Máquina: Lista 04

Raissa Camelo Salhab

[1]Departamento de Computação – Universidade Federal Rural Pernambucana (UFRPE)
Bacharelado em Ciência da Computação

srtacamelo@gmail.com

Resumo. *Neste artigo foi realizado um estudo comparativo entre diferentes algoritmos classificadores e seu desempenho para classificar figuras geométricas contidas em imagens no formato '.jpg'.*

1. Introdução

O problema de detecção de formas geométricas em imagens computadorizadas já foi extensamente explorado no estado da arte. [Chauhan and Dhingra 2012]. Consiste em conseguir extrair a melhor combinação de características possíveis afim de distinguir uma forma geométrica da outra com algoritmos classificadores. [Caselles et al. 1993].
Este projeto serviu como forma de introdução ao problema, as abordagens existentes de processamento de imagem para manipulação e extração de características. Através deste trabalho é buscada a comparação entre diferentes técnicas de aprendizado de máquina e de algoritmos classificadores.

2. Materiais

O conjunto de dados utilizado foi construído manualmente pelos alunos da disciplina Aprendizado de máquina, consistindo de um conjunto de 1072 imagens de nove categorias: círculos, elipses, hexágonos, linhas, retângulos, losangos, trapézios e triângulos, cada categoria com 120 imagens no formato ".jpg". As imagens foram criadas a partir de desenhos feitos em sala de aula, desenhados a mão. Cada aluno desenhou as diversas formas geométricas e fotografou os desenhos depois que foram formatados para '.jpg' posteriormente. As imagens consistem em figuras geométricas dos nove tipos descritos acima, desenhadas na cor preta em um fundo branco. Existem várias bases de dados de figuras geométricas disponíveis para download, como a do Kaggle, ICS-Forth e T2SSD, porem o objetivo dessa atividade foi fazer com que os alunos se engajassem na construção de uma base de dados própria, afim de testar os diferentes métodos de classificação em uma base adquirida em sala de aula.
Por ter sido feita a mão a base contem algumas falhas, nem todos os desenhos são formas geométricas perfeitas, muitos com traços incontínuos, inclinações acidentais, traços tronchos, etc. Essas falhas ocasionam ma interpretação das funções de extração de características, levando os classificadores a resultados errôneos, por conta de uma inclinação acidental o algoritmo pode considerar que a figura possui mais vértices que o normal, classificando-a erroneamente. Devido a essa característica o pré-processamento foi fundamental para diminuir as falhas e tornar a base mais legível, diminuindo o invesamento na extração de características e otimizando a classificação final.
A seguir um exemplo das imagens contidas no dataset, um conjunto de quatro imagens aleatórias de qualidade satisfatória, contendo poucas inclinações, acidentes e aberturas. E outras quatro imagens aleatória, apresentando as falhas citadas.

Figura 1. Figuras de qualidade satisfatória

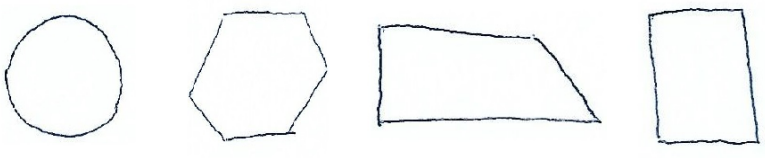
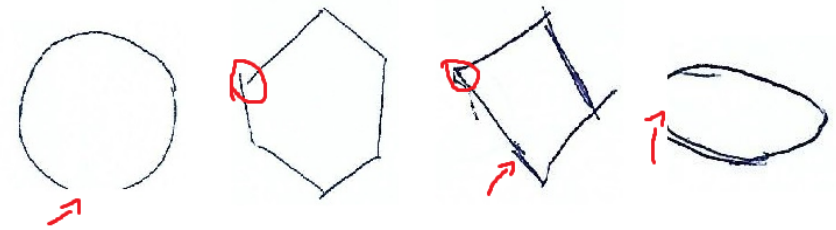


Figura 2. Figuras de qualidade duvidosa



3. Pré-processamento

3.1. Binarização e Suavização da Imagem

Inicialmente as imagens foram lidas uma por uma utilizando a biblioteca opencv, salvando-as em variáveis python. Em seguida cada imagem foi convertida do padrão RGB para binário, removendo as informações de cor desnecessárias para a extração de características e definição dos contornos. As imagens foram passadas em um filtro Gaussiano, para amenizar e suavizar os contornos, removendo riscos e traços extras, indesejados. Afim de aplicar outros métodos de extração de características do opencv, foram realçadas as bordas das figuras, utilizando o threshold do opencv. Ao converter as imagens para tons de cinza binarizados as cores foram invertidas, trocando a cor do fundo (branco) para preto e as linhas das figuras para branco. Esse passo é realizado afim de atribuir valores maiores para os traços mais importantes da figura (desenho geométrico), as cores mais claras e mais próximas ao branco possuem valores de tom de cinza mais próximos de 255 (branco puro), enquanto cores mais escuras possuem valores mais próximos de 0 (preto). Ao inverter as cores estamos atribuindo valores mais próximos de 255 aos traços os quais desejamos extrair informações. Depois de inverter a imagem ao aplicar ao limiarização, foi utilizado o filtro de Canny, para detecção e apuração de contornos, um passo extra para fortificar as linhas das figuras geométricas, porém ao fim dos testes foi constatado que o filtro de Canny não ajuda no pré-processamento, causando mais falhas nas imagens do que reparando-as, no final optou-se por não utilizá-lo.
A seguir o trecho de código que realiza tais operações:

Figura 3. Pré-processamento I

```
def doTheMagic(img_path,label):
    img = cv2.imread(img_path)

    gray_image = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    blur = cv2.GaussianBlur(gray_image, (3, 3), 0)
    ret, tre = cv2.threshold(blur, 127, 255, cv2.THRESH_BINARY_INV)

    edged = cv2.Canny(tre, 50, 100)

    kernel = np.ones((5, 5), np.uint8)
    dilation = cv2.dilate(edged, kernel, iterations=2)
    erosion = cv2.erode(dilation, kernel, iterations=2)

    extractFeatures(erosion, label)
```

A seguir os resultados da aplicação em uma figura qualquer da base de imagens, na ordem: RGB to Gray, Filtro Gaussiano, Threshold + Inversão, Canny-Edge:

Figura 4. Pré-processamento I

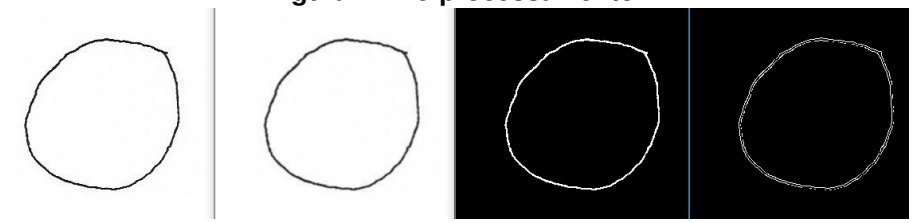
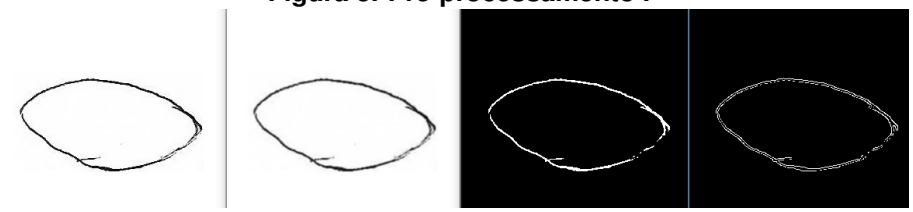


Figura 5. Pré-processamento I



Como pode-se perceber o detector de bordas Canny diminui a concentração de pixels do desenho, dado que seu objetivo é detectar as bordas mais externas, logo, não é de utilidade para esse problema.
Outro fator que pode-se observar é que o passo de inverter as cores da imagem junto ao threshold também diminui a qualidade da imagem e atenua as falhas presentes, o que por si só já prejudica o resultado. Contudo, o passo da limiarização não pode ser pulado.
Para finalizar a primeira etapa de pre-processamento, utilizamos os métodos de dilatação e erosão de imagem, para podermos suavizar as falhas nos contornos e as imagens que contem buracos nas linhas.
A seguir duas imagens demonstrando o procedimento de dilatação e erosão em imagens aleatórias.

Figura 6. Pré-processamento I

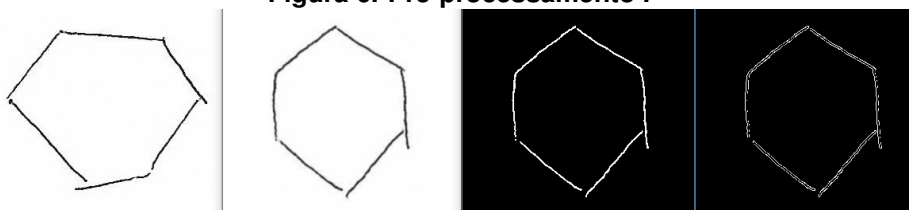
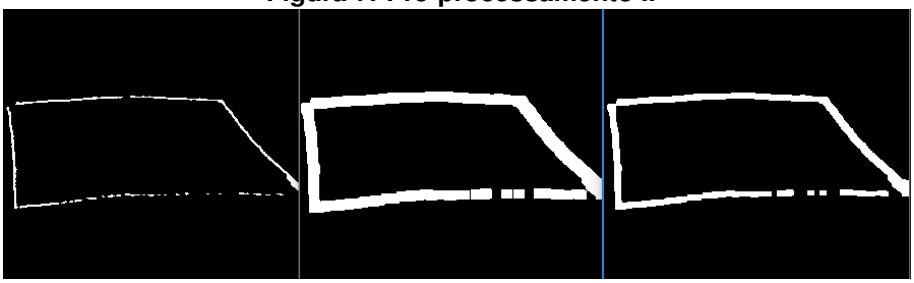


Figura 7. Pré-processamento II



3.2. Extração de Características

Para a extração de características foram retirados os seguintes atributos das imagens da base: Altura da figura, largura da figura, quantidade de vértices, perímetro e área da figura. As mesmas foram extraídas utilizando a biblioteca opencv através dos contornos das figuras.
A seguir imagens contendo a

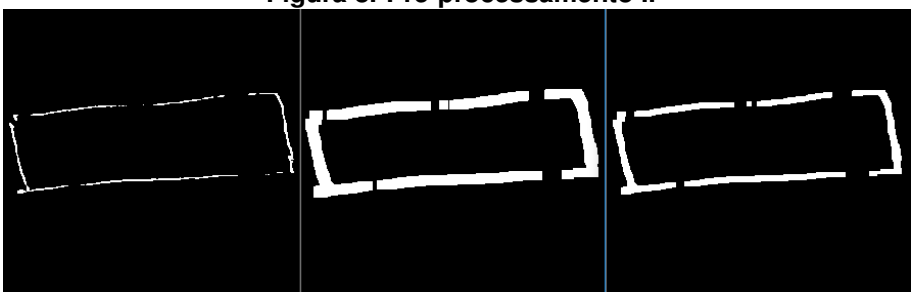
4. Metodologia

Para a avaliação dos resultados o método da validação cruzada foi utilizado sobre os dados, através da função k-fold da biblioteca scikit learn. A tabela gerada durante o pre-processamento foi carregada sobre a forma de um dataframe, da biblioteca pandas, para melhor organizar os dados e fornecer os valores de maneira adequada para os classificadores. Após lidos os valores foram randomizados, usando uma função da própria biblioteca pandas que mistura as linhas dos dataframes sem que os valores das colunas se misturem. Após randomizados, os dados foram divididos entre frequências e classes, dividindo-os em dois dataframes separados. A função de k-fold do sklearn ficou responsável por dividir os dados em 10 folds separados e fazer sua iteração num for. a cada iteração do 10-folds o mesmo conjunto de treino e testes foi usado para todos os algoritmos, gerando uma medida de acurácia para todos eles. O método 10-folds foi repetido 5 vezes diferentes, cada vez randomizando novamente o conjunto de dados, gerando no total 50 valores de acurácia para cada classificador. A média, a mediana e o desvio padrão de cada conjunto de acurácia foi extraída e a hipótese de t-studant foi utilizada para avaliar as diferenças entre abordagens modificadas dos mesmos classificadores.

5. Modelos Apresentados

Dentre os classificadores utilizados para a comparação se encontram a árvore de decisão, naive bayes, SVM, KNN, regressão logística e MLP. A seguir:

Figura 8. Pré-processamento II



5.0.1. Árvore de Decisão

A árvore de decisão é um modelo de aprendizagem que representa uma série de ”regras”para se determinar o pertencimento de um dado a determinada classe. Basicamente, as árvores de decisão traçam caminhos com base nas características de cada dado, inferindo a partir destas características a classe deste dado. O modelo de árvore escolhido usa o algoritmo padrão com base nos valores de entropia de cada característica. Ou seja, a árvore é montada através dos dados treinamento, analisando as características que mais distinguem uma classe de outra primeiro e assim decidindo a que classe o dado pertence.

5.0.2. Naive Bayes

O naive bayes utiliza os valores de treino para criar um modelo probabilístico, presumindo que cada um dos dados seja independente. Desta forma a polaridade de um documento é definida dada a maior tendencia de classe dos dados. Se a maioria dos termos de um documento tendem a ser positivos, logo o documento inteiro é classificado como positivo.

5.0.3. Support Vector Machine

A maquina de vetores suportes cria um modelo não probabilístico de classificação com base nos valores de treinamento, traçando uma reta imaginaria, que dividiria os dados entre as duas polaridades possíveis. Se representarmos os dados em um plano cartesiano, esta reta imaginaria consiste em nada mais nada menos do que uma função que corta o plano, separando as duas classes. Para os testes realizados utilizamos as variações ”RBF”e ”Linear”da SVM.

5.0.4. K- Nearest Neighbours

O metodo KNN calcula a distancia de cada dado do conjunto de testes para cada dado do conjunto de treino (classificado), os k dados do conjunto de treino mais próximos do dado que se deseja classificar são escolhidos e a classe que mais se repetir entre eles é dada ao novo dado. Esse método possui duas variações distintas que foram testadas, o padrão k-nn e o wKNN ou KNN ponderado, que utiliza uma função de distancia para recalcular as distancias entre os dados, fazendo com que os dados mais próximos se aproximem mais e os mais distantes se afastem. [**?**]

5.0.5. Regressão Logistica

A regressão logística é uma técnica estatística que visa estimar a probabilidade de ocorrência de determinado ”acontecimento”dada a existência de determinadas características independentes, assim criando um modelo, geralmente dicótomo, de classificação com base nessas características. No problema proposto a regressão logística deve classificar cada texto com base nas frequências de cada termo, ou seja, o algoritmo encontrará padrões de associação entre um determinado conjunto de termos e as classes positiva e negativa.

5.1. MLP

O Multi Layer Perceptron consiste em várias camadas de neurônios (perceptrons), que são estruturas matemáticas que simulam neurônios humanos. A MLP é um tipo de rede neural que aplica funções de ativação em um conjunto de dados para determinar a qual classe este pertence.

6. Resultados Esperados

Dada a natureza do problema estima-se que as abordagens probabilísticas como o Naive Bayes e a regressão logística obtenham melhores resultados, . As técnicas de KNN também possuem grandes chances de estarem entre os melhores resultados, pois estas técnicas tendem a encontrar padrões matemáticos com maior facilidade, distinguindo as classes (figuras geométricas) com base em probabilidade (Naive Bayes) e distancia das características (KNN). Também espera-se um bom resultado da MLP que é um dos métodos mais eficases em termos de desempenho e eficiência (tempo)

7. Analise Experimental

A analise experimental consistiu na parte mais importante do projeto, promovendo uma comparação entre as distintas abordagens de classificação fornecidas pela biblioteca sklearn. O objetivo final desta sessão é determinar quais técnicas são mais adequadas para o problema proposto e o motivo das mesmas terem obtido um resultado melhor em detrimento das restantes.

Para a analise ser possível foram utilizadas as métricas de acurácia para medir a taxa de acerto de cada classificador, sendo extraídas logo apos os testes com cada algoritmo. As medidas de média, mediana e desvio padrão foram tiradas sobre as 50 acurácias geradas pelas 5 iterações de 10-folds. A biblioteca sklearn providenciou as funções para extração de tais valores.

Os valores da hipotese de t-studant calculados A analise experimental foi dividida em três partes, primeiramente a média, a mediana e o desvio padrão do conjunto de acurácia de cada classificador foi avaliado, e uma comparação entre os resultados foi realizada. O objetivo da primeira analise é, tendo em vista a média de acurácias, destacar quais foram as melhores abordagens, buscando entender e explanar os motivos de tais resultados. Logo em seguida uma comparação entre cada variação dos algoritmos KNN e SVM foi feita baseada na hipótese de t-studant, gerada entre o conjunto de acurácias das variações. Aqui buscamos determinar se houve realmente alguma diferença significante entre as variações, comparando também o resultado da hipótese com as médias de acurácia e verificando a congruência dos dois dados.

Figura 9. Resultados

Média, Desvio Padrão e Mediana da Acurácia			
Classificador	Média	Desvio Padrão	Mediana
Árvore	0.3559	0.053	0.35
Naive Bayes	0.39	0.0484	0.3953
KNN (n=1)	0.8022	0.0367	0.8037
KNN (n=3)	0.6656	0.0445	0.6697
KNN (n=5)	0.6561	0.0449	0.6511
WKNN (n=1)	0.8022	0.0367	0.8037
WKNN (n=3)	0.7992	0.0400	0.7943
WKNN (n=5)	0.8041	0.0425	0.8037
SVM-LINEAR	0.4554	0.0434	0.4651
SVM-RBF	0.5960	0.0401	0.5934
REG-LOG	0.4216	0.0452	0.4279
MLP	0.2801	0.0535	0.2897

Como esperado, o KNN e o WKNN obtiveram bons resultados, obtendo uma acurácia de 80% tanto para k = 1 e aproximadamente 60% para os outros valores de k. Ambas as SVMs (RBF e Linear) e a regressão logística obtiveram marcas inferiores.
Surpreendentemente a MLP obteve o pior resultado, provavelmente por conta do conjunto de características escolhidas e a arquitetura escolhida (camadas, neurônios etc).
Outra surpresa foi o algoritmo da árvore de decisão, o qual se esperava resultados melhores. A árvore de decisão consegue abstrair quais atributos possuem um grau maior de ”separação”entre classes, criando assim uma série de regras que em uma ordem especifica conseguem distinguir um dado de uma classe de outra, contudo o resultado obtido foi bem menor do que os do KNN e da regressão logística, esperava-se que fosse mais próximo.

A seguir encontra-se a tabela com a analise do tempo de execução de cada algoritmo:

Figura 10. Resultados II

Média, Desvio Padrão e Mediana do Tempo de Execução			
Classificador	Média	Desvio Padrão	Mediana
Árvore	0.0059	0.0017	0.006
Naive Bayes	0.0027	0.0011	0.003
KNN (n=1)	0.0026	0.0007	0.003
KNN (n=3)	0.0029	0.0007	0.003
KNN (n=5)	0.0031	0.0008	0.003
WKNN (n=1)	0.0031	0.0015	0.003
WKNN (n=3)	0.0030	0.0009	0.003
WKNN (n=5)	0.0032	0.0006	0.003
SVM-LINEAR	0.0364	0.0049	0.035
SVM-RBF	0.0333	0.0039	0.032
REG-LOG	0.0284	0.0034	0.028
MLP	9.519	0.9002	94.579

Observa-se que a MLP estranhamente obteve o maior tempo de execução chegando a uma média de aproximadamente 9 segundos. Tal marca se obteve em detrimento de uma característica da biblioteca Keras, que printa os resultados de cada época e iteração da rede neural. Por conta da impressão dos resultados na tela o calculo do tempo de execução ficou enviesado, obtendo um longo tempo.
A árvore de decisão, o Naive Bayes, o KNN e o WKNN obtiveram marcas aproximadas, enquanto as SVMs demoraram 10 vezes mais que as de cima, devido ao alocamento de memória necessário para o algoritmo.
A seguir encontra-se uma analise entre as variações de KNN e SVM, dada a hipotese de t-studant fornecida na tabela abaixo:

Figura 11. Comparação das variações

KNN e SVM variações		
Comparação	t-value	p-value
KNN-1 e KNN-3	167.222	1.830
KNN-1 e KNN-5	17.781	19.513
KNN-3 e KNN-5	1.06287	0.2904
WKNN-1 e WKNN-3	0.3896	0.6976
WKNN-1 e WKNN-5	-0.232	0.816
WKNN-3 e WKNN-5	-0.5867	0.5587
SVM-LIN e SVM-RBF	-168.142	1.2267

Como se pode perceber, as variações entre SVM-LIN e SVM-RBF são praticamente nulas

de acordo com a hipótese estatística, não possuindo nenhuma diferença significativa. O restante dos algoritmos também apresentam valores de p-value inferiores à 5%, demonstrando que não houve grandes ganhos entre um método ou outro.

8. Conclusões

O pré-processamento utilizado é de extrema importância para a obtenção dos resultados, dependendo dos métodos utilizados as imagens podem sofrer perdas de informações, o que pode dificultar em sua classificação.

As características utilizadas para a extração também são de extrema importância, dado que algumas características distinguem melhor uma forma geométrica da outra. Futuros testes com novas características podem apontar melhores resultados.

As configurações de layers e neurônios de uma MLP também devem ser levadas em consideração quando se inicializa um experimento de classificação. Dependendo da configuração de arquitetura utilizada pode se obter um resultado muito bom ou muito ruim.

Referências

Caselles, V., Catté, F., Coll, T., and Dibos, F. (1993). A geometric model for active contours in image processing. *Numerische mathematik*, 66(1):1–31.

Chauhan, S. and Dhingra, S. (2012). Pattern recognition system using mlp neural networks. *Pattern Recognition*, 4(9):43–46.